Localized Distance-sensitive Service Discovery in Wireless Sensor and Actor Networks

Xu Li, Nicola Santoro, and Ivan Stojmenovic, *Fellow, IEEE*

Abstract—We formalize the *distance-sensitive service discovery* problem in wireless sensor and actor networks, and propose a novel localized algorithm, iMesh. Unlike existing solutions, iMesh uses no global computation and generates constant per node storage load. In iMesh, new service providers (i.e., actors) publish their location information in four directions, updating an *information mesh*. Information propagation for relatively remote services is restricted by a blocking rule, which also updates the mesh structure. Based on an extension rule, nodes along mesh edges may further advertise newly arrived relatively near service by backward distance-limited transmissions, replacing previously closer service location. The final information mesh is a planar structure constituted by the information propagation paths. It stores locations of all the service providers and serves as service directory. Service consumers (i.e., sensors) conduct a lookup process restricted within their home mesh cells to discover nearby services. We analytically study the properties of iMesh including construction cost and distance sensitivity over a static network model. We evaluate its performance in static/dynamic network scenarios through extensive simulation. Simulation results verify our theoretical findings and show that iMesh guarantees nearby (closest) service selection with very high probability $> 99\%$ (resp., $> 95\%$).

Index Terms—Service discovery, Distance sensitivity, Localized algorithms, Sensor networks, Wireless networks

1 Introduction

A wireless sensor network is a collection of micro-sized and resource-constrained wireless sensing devices, *sensors*, deployed in a region of interest for surveillance purpose. Traditionally, it is a data gathering network where nodes are stationary and responsible only for sampling their surroundings and reporting to predefined data sinks. As hardware technology advances, it is now evolving toward service-oriented wireless sensor and actor network (WSAN) [1]. In WSAN, actor nodes are able to interact with the physical world. They may be static or mobile. Examples of static actors are sprinklers attached on the ceiling of an office room and sound-sensitive lights installed in a dark hallway.

In this work, we focus on WSAN with mobile actors (e.g., mobile robots, and unmanned aerial vehicles). Actors offer movement-assisted services to sensor nodes and/or to their monitored environment. They remain static and move only upon request. Because each actor serves only one sensor at a time, it does not response to any service request while delivering service. For example, when a sensor is about to deplete its battery power, it discovers an actor carrying battery charger and requests for battery recharging service. After receiving the request, the actor relocates to the position of the sensor and recharges its battery. Another example is in a WSAN for fire detection and fire fighting in a woody area. After a fire occurs, surrounding sensors detect it and collaborate among themselves to aggregate data, and one of them reports the fire to a firefighter actor, which then moves to extinguish the fire.

Service discovery is a crucial component of any service-oriented network. Discovery criteria depend on the underlying network and the application. In movement-assisted service delivery cases in WSAN, delivery distance is a primary concern for energy saving and timely response. This heralds the emergence of *distance-sensitive* service discovery. Here, distance sensitivity implies closest or nearby service selection. Informally speaking, *closest service selection* means that each service consumer discovers the closest service provider, while *nearby service selection* means that each service consumer discovers a service provider that is at most twice as far as the closest. In addition to the distance-sensitivity requirement, service discovery must be conducted in an efficient way, i.e., with constant per node storage load and with no global computation, for the resource constraints of WSAN.

Generic service discovery algorithms, e.g., [6], [8], [10], [11], and adoptable techniques e.g., [4], [12], [16], [18], [19], have been proposed, each however with major weaknesses in resource usage. None of them were developed with distance-sensitivity in mind. They are not a good option for distance-sensitive service discovery in WSAN. Specialized and efficient solutions are needed.

1.1 Intuition

Intuitively, if we construct a *Voronoi diagram* [2] using service providers (i.e., actors) as creating nodes and let each of them distribute its location along the perimeter of its Voronoi polygon, then the Voronoi diagram

- *X. Li is with SITE, University of Ottawa, Ontario, Canada. Email: xuli@site.uottawa.ca. Part of this work was done when he was at SCS, Carleton University.*
- *N. Santoro is with SCS, Carleton University, Ontario, Canada. Email: santoro@scs.carleton.ca*
- *I. Stojmenovic is with SITE, University of Ottawa, Ontario, Canada and EECE, University of Birmingham, UK. Email: stojmenovic@storm.ca*

becomes a distributed service directory with bounded per-node storage load. In this case, distance-sensitive service lookup becomes localized (restricted in a Voronoi polygon). That is, a service consumer (i.e., a sensor) queries in an arbitrary direction, and will find closest service provider once it hits the perimeter of its home Voronoi polygon. This intuitive solution possesses all the properties that we are looking for; but it requires *global computation*. Hence, to make it practical, as service directory we must substitute the Voronoi diagram with a localized planar structure with good proximity property.

A naive idea of improvement is to replace Voronoi diagram with *square mesh* constructed by the well-known quorum technique (Quorum for short) [19]. That is, service providers propagate their location information in four geographic directions, i.e., north, east, south, and west, across the entire network; the propagation paths form a mesh structure as service directory. Although this method requires only local computation, it can generate inconstant storage load on network nodes if service providers are all placed in a line, and it also makes the localized in-cell lookup no longer able to provide closest/nearby service selection guarantee because the mesh structure bears *no proximity* property.

As we show in this article, it is however possible to modify Quorum to obtain a planar structure that (as the mesh but unlike the Voronoi diagram) can be constructed in a purely localized manner and (as the Voronoi diagram but unlike the mesh) possesses our required proximity property. The needed modification is the use of distance-based blocking, and has been informally suggested in [20] for content location problem and formally rediscovered by us in [14]. Unlike our work presented here, previous work [20] did not contain theoretical analysis of the resulting information structure, and it did not consider the use, in addition to information blocking, of an important extension rule that we show leads to major improvements in the performance.

1.2 Contributions

We formalize the *distance-sensitive service discovery* problem and propose a novel localized solution, iMesh. iMesh is grounded on a planar structure, which we refer to as *information mesh*, created by the use of the information blocking rule [20] in the traditional quorum-based mesh construction, and the use of our newly proposed information extension rule. We present, analyze and evaluate iMesh in the context of singular-service WSAN. Its natural extension to multi-service scenarios is discussed near the end of the article.

In iMesh, service providers publish their location in the four geographic directions: north, east, south, and west. During transmission, information collinearly or orthogonally blocks each other by the blocking rule: a node receiving information from multiple service providers forwards only the information of the closest one. It may however also be extended to other directions by the extension rule: a node where information x orthogonally blocks information y transmits x along the backward transmission path of y for a limited distance. Information propagation paths together form the information mesh that distributedly stores the locations of all the service providers. To discover nearby services, service consumers conduct a simple cross lookup process within their home mesh cells.

We present a thorough analytical study on iMesh over a static grid network model. Our analysis focuses on the properties (construction cost and distance-sensitivity) of the information mesh constructed by the basic iMesh (denoted by iMesh-A) [20] using the blocking rule only, and by the complete iMesh (denoted by iMesh-B) containing both the blocking rule and the extension rule. These results are confirmed by extensive simulation.

Our experimental results show that iMesh-A and iMesh-B guarantee closest service selection respectively with probability $> 95\%$ and $> 97\%$, and nearby service selection both with probability $> 99\%$. They indicate that iMesh-B uses negligible extra communication in exchange for noticeably improved distance sensitivity. Through simulation we as well comparatively evaluate the construction cost of the information structures of iMesh and Quorum both in static and in dynamic networks. It is observed that iMesh generates significantly lower message overhead than Quorum.

We review related work in Sec. 2 and define network model in Sec. 3. We describe iMesh-A and analyze its properties respectively in Sec. 4 and 5, and present iMesh-B afterwards in Sec. 6. We give the implementation detail of iMesh in Sec. 7 and provide comparative simulation study of iMesh and Quorum in Sec. 8. We show the practical significance of iMesh by describing an application example in Sec. 9. We extend iMesh to multi-service scenarios in Sec. 10. Finally, we discuss some open problems in Sec. 11.

2 Related Work

Many service discovery algorithms have been proposed for wireless ad hoc networks. These algorithms can be categorized either as *directory-based approach* or as *directory-less approach*. The former, e.g. [10], [11], use a well structured service directory to store service provider information and to facilitate service lookup. They usually require global communication/computation such as clustering and dominating set formation for service directory construction and maintenance. The latter, e.g. [6], [8], do not maintain any special component but rely on periodical service advertisement and multicasting-/anycasting- based service lookup. Their execution often involves (limited) flooding operations. Because these existing algorithms may generate large message overhead and/or inconstant per node storage load, they are not suitable for resource-constrained WSAN. A survey of service discovery algorithms can be found in [9], [15].

In addition to specialized service discovery algorithms, there exist other techniques, e.g., data-centric

storage schemes [4], [16], [18] and location services [12], [19], [20], which can be adopted to solve the service discovery problem. In the following we will review some of these related works.

Fang et al. [4] presented a landmark-based data storage and retrieval scheme. This scheme constructs a number of globalized shortest path trees of predefined landmark nodes, each rooted at a landmark node. A data producer hashes data (according to data type) to a certain landmark node, and distributes the data using the shortest path tree rooted at that landmark node. A data consumer queries along the same shortest path tree; it gets the data when it hits the storage path or reaches the hash node. This scheme generates storage hot spots and involves many expensive global operations. It provides no energy-efficiency and does not scale.

Ratnasamy et al. [16] presented a Geographic Hash Table (GHT) scheme for data-centric storage. A node hashes data to a unique location by data type and routes the data to that location by a combined greedy-face routing protocol. The nodes that enclose the harsh location in a planar graph store the data; other nodes may get the data from any of these nodes. The main drawback of this scheme is the undesired non-locality-aware data query, that is that a node near the data source may have to travel a long distance to retrieve the data. Also, it may induce bottleneck spots when some types of data are frequently generated or requested.

Li et al. [12] presented a Grid Location Service (GLS). This algorithm partitions the sensory field into grids and constructs a quad-tree structure over the grids. It uses a hash function, designed on basis of the quad tree, to match each node (by ID) to a unique subset of nodes called location servers. Every node updates all its location servers with its current location. A node can find the location of any other node by querying one of the location servers of that node. This protocol requires pre-knowledge of the sensory field for grid partition. It may generate large message overhead since location updates and location queries travel along zigzag lines.

Stojmenovic et al. [19] presented a quorum-based location service (Quorum). Each node distributes its current position along a "column" in the network. When a node wants to discover the location of another node, it searches along a "row" in the network. This row intersects the columns of all the other nodes, thus ensuring discovery. The weakness of this protocol is that location update and search has to cross the entire network, and network boundary has to be included to guarantee intersection. In addition, if all the nodes are collinear along a column, every node has to store every other's location, suffering from large storage load.

Tchakarov and Vaidya [20] presented a Geography-based Content Location Protocol (GCLP). Content servers advertise their locations in four directions on a periodic basis. Nodes receiving location advertisements become content location server. "If a content location server receives multiple advertisements for a particular resource, it will only forward updates from the content server closest to it". This forwarding policy is an informal definition of the blocking rule rediscovered by us for information mesh construction in the preliminary version [14] of this article. Due to its periodic location advertisement, GCLP generates large message overhead. In this article, we propose an important information extension rule, which, when used together with the blocking rule, leads to significant improvement in the performance on distance sensitivity.

Summarizing, all these algorithms (but GCLP [20]) have some, if not all, of the following weaknesses: requirement for pre-knowledge of the network, frequent global computation, inconstant per node storage load, communication bottleneck spots, and non-locality-aware service lookup. On the contrary, the algorithm iMesh proposed in this article has none of those drawbacks and may lead to efficient use of WSAN.

3 Model and Definitions

We first consider a static WSAN where actors do not move. The network can be viewed as a snapshot of a dynamic WSAN with mobile actors. We assume nodes are placed exactly at the intersection points of a square grid structure and provide analytical study in Sec. 5. The reasons for choosing to study the grid network model are that it has already been adopted in literature [5], [21] to facilitate performance analysis of static wireless ad hoc networks, and that we want to emphasize on the theoretical aspects of iMesh. We verify our theoretical findings through simulation in Sec. 8.3 and 8.4.

Static-network-based study provides good insights into the performance, especially the distance sensitivity, of iMesh. But it does not reflect the maintenance cost induced by actor mobility. In Sec. 8.5, we experimentally study the overhead of iMesh in dynamic networks. In our simulation experiments, *actors move only upon request and become unavailable while moving*, and the information structure is updated dynamically to reflect the availability and location change of actors. This mobility model is motivated by applications described in Sec. 1.

In the sequel, actors are called *service providers* (SPs). They are scattered in the network at random. Sensors that require services themselves or on behalf of their monitored physical objects are called *service consumers* (SCs). All the nodes, whether SPs or SCs, have the same communication radius. By convention, we denote by $|ab|$ the Euclidean distance of two points (nodes) a and b and by $|S|$ the size of a set S.

We denote the network graph by $G(V, E)$ (or simply by G), where V and E are the set of nodes and the set of edges in G, respectively. We use S_P and S_C to represent SP set and SC set in G. Let $n = |V|$ and $\nu = |S_P|$. By definition, $\nu \leq n$, $S_P \cap S_C = \phi$, and $S_P \cup S_C \subseteq V$. Let $P(a)$ represent the SP-node discovered by an SC-node a. We formally define closest service selection and nearby service selection as follows:

***Definition** 1 (*Closest service selection):

$$\forall a \in S_C, |aP(a)| = \min_{p \in S_P} \{|ap|\}.$$

***Definition** 2 (*Nearby service selection):

$$\forall a \in S_C, |aP(a)| \leq 2 \min_{p \in S_P} \{|ap|\}.$$

We require nodes to know their own locations. We believe this requirement is reasonable for the surveillance goal of WSAN. We assume the standard restrictions, i.e., total reliability, FIFO communication channel, bidirectional links and finite communication delay, commonly used in distributed computing domain [17].

4 Basic iMesh Protocol

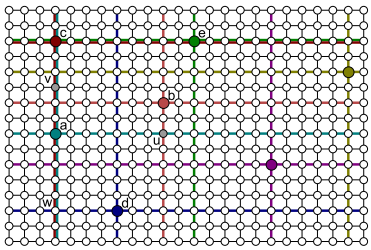
We shall now present the basic version of iMesh, iMesh-A, which is a generalization of GCLP [20]. It uses the blocking rule alone to build the service directory, i.e., *information mesh*. The complete version, iMesh-B, containing both the blocking rule and our newly proposed extension rule will be presented later, in Sec. 6. We describe iMesh in the grid network model for ease of understanding. Its implementation detail for arbitrary network scenarios will be given in Sec. 7.

4.1 Information mesh construction

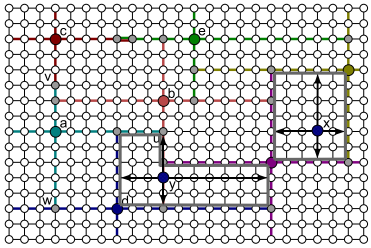
Consider only the residing rows and columns of the SP-nodes in G. They intersect each other and constitute a complete mesh, as illustrated in Fig. 1(a), where SP-nodes are represented by solid big dots, and their residing rows and columns are highlighted by thick lines. If each SP distributes its own location information (by a *registration message*) among the nodes along its residing row and column, this complete mesh distributedly stores the location information of all the SPs and therefore can be used for the purpose of service discovery.

Let us closely examine the complete mesh structure in Fig. 1(a). SP-node c is closer to the area above the midpoint node v between itself and the vertically collinear SP-node a, and thus it has relatively high priority to be discovered by the SC-nodes in that area. In addition, SP-node b might be a better choice for the SC-nodes located in its right-side area than SP-node a. In these cases, a does not need to distribute its location information in those areas. Similar argument can be made for other SP-nodes. By this observation, we define a blocking rule (an informal definition can be found in [20]):

***Rule** 1 (Blocking Rule – A Formalization):* A common node u of the residing rows/columns of two SPs a and b ($a \neq b$) stops the further propagation of the information of a, iff $|ua| > |ub| \vee |ua| = |ub| \wedge colline(a,b) \vee |ua| = |ub| \wedge \neg colline(a,b) \wedge horizon(b)$, where $colline(a,b)$ and $horizon(b)$ denote the case that a and b are (vertically or horizontally) collinear and the case that the involvement of b is along the horizontal direction, respectively. When this blocking happens, we say "b blocks a at u" and denote it by $a \overset{u}{\leftarrow} b$ or $b \overset{u}{\rightarrow} a$.



(a) A complete mesh



(b) An information mesh

Fig. 1. Mesh construction in a grid sensor network

Application of the above blocking rule can lead to the merge of adjacent mesh cells and result in a pruned mesh structure, which is what we call *information mesh*. We denote the information mesh constructed over G by $\mathcal{M}(G)$ (or simply by $\mathcal{M}$). Figure 1(b), where solid small dots represent the nodes at which the blocking rule is applied, shows the information mesh corresponding to the mesh structure in Fig. 1(a). By the blocking rule definition, we have the following corollary.

***Corollary** 1: $\mathcal{M}$* is a planar structure.

In an asynchronous environment, a node c, at which an SP-node b is supposed to block another SP-node a, may wrongly retransmit the registration message of a, because of the late arrival of the registration message of b, violating the blocking rule. This problematic situation is detected by c, as soon as it receives both of the two messages. Then it as initiator starts a revocation process, in which the inconsistent information is erased from $\mathcal{M}$. More specifically, c sends a *revocation message* following the forward path of a's registration message. The revocation message stops at a node where a's registration message stopped propagating. Nodes that receive this revocation message remove a's information from their local repositories. Such a revocation process can lead to chain effect. That is, the registration message of an SP-node previously incorrectly blocked will continue to propagate until the blocking rule is satisfied again.

Information revocation supports actor mobility. Right before an SP node leaves its current position (for service delivery), it starts a revocation process to remove its own information from $\mathcal{M}$; immediately after it becomes available again, it re-registers as a new SP.

4.2 Distance-sensitive service lookup

For an SC-node a, the objective of its service lookup is to identify the location of its target service provider $T(a)$

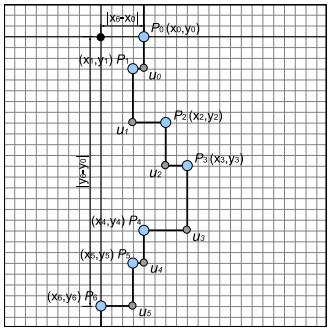


Fig. 2. Blocking chain $p_0 \overset{6}{\Leftarrow} p_6$ ($p_0 \overset{u_0}{\leftarrow} p_1 \overset{u_1}{\leftarrow} \cdots \overset{p_5}{\leftarrow} p_6$)

(see below for definition). Depending on the position of a, there are two possible lookup cases: *in-cell* case and *on-edge* case.

***Definition** 3 (Home Cell):* The home cell $HCell(a)$ of an SC a is the mesh cell where a is located in or the aggregation of the mesh cells which a is adjacent to.

***Definition** 4 (SPV):* The Set of Providers in Vicinity $SPV(a)$ of an SC a is the set of SPs that distribute their information along the perimeter of $HCell(a)$.

***Definition** 5 (Target Service Provider):* The target service provider $T(a)$ of an SC a is a nearest SP in $SPV(a)$.

In the in-cell case, a is located inside a cell of $\mathcal{M}$. When a wants to find $T(a)$, it sends a search message along its residing grid row and column in four directions. Such a search message stops its further transmission as soon as it hits a mesh edge (or the border of G), and then the node at which the message stops replies a with the location of its recorded SP-node closest to a (resp., a failure notice). If there is no SP in the network, what a will receive are all failure notice; otherwise, a can find the location of $T(a)$ simply by a local comparison among its received location data. Because the search paths of a form a cross, we call this method *cross lookup*.

The cross lookup method can also be applied to the on-edge situation, namely, when the SC-node a is riding on an edge of $\mathcal{M}$. In this case, the search message that travels along a residing mesh edge of a stops at the farthest end of the mesh edge on the home cell perimeter. By this means, a reaches all the composing mesh edges of $HCell(a)$ and make a right decision. Figure 1(b) illustrates the cross lookup of an in-cell SC-node x and an on-edge SC-node y. In the figure, the home cells of the two nodes are marked by thick gray lines, and their search paths are highlighted by arrowed black lines.

5 Analysis

We now explore the theoretical aspects of iMesh-A. Our analysis is conducted in a static grid network. In this scenario, once constructed, an information mesh has zero maintenance cost. As we will see, iMesh-A has low message complexity and optimal per node storage load; but it does not always provide perfect nearby service selection guarantee (rare counterexample cases exist).

***Definition** 6 (Chain Blocking):* For two SPs a and b ($a \neq b$), b is said "chain-blocking a" if there is a blocking chain of length k ($k \geq 1$) from b to a, i.e., $a \overset{u_0}{\leftarrow} \cdots \overset{u_{k-1}}{\leftarrow} b$. We denote this chain blocking by $a \overset{k}{\Leftarrow} b$ or $b \overset{k}{\Rightarrow} a$.

***Lemma** 1:* In a blocking chain $a \overset{k}{\Leftarrow} b$ along Y (resp., X) axis, the distance of a and b along X (resp., Y) axis is not longer than their distance along Y (resp., X) axis.

Proof: For illustrative purpose, take as an example the blocking chain $p_0 \overset{6}{\Leftarrow} p_6$ in Fig. 2, where $a = p_0$, $b = p_6$ and $k = 6$. Consider two consecutive SP-nodes p_i and p_{i-1} ($1 \leq i \leq k$) in the blocking chain. $|x_i - x_{i-1}| \leq |y_i - y_{i-1}|$, where (x_i, y_i) and (x_{i-1}, y_{i-1}) are respectively the coordinates of p_i and p_{i-1}. It is because, otherwise, p_i, can not block p_{i-1} in Y-direction. In this case, $|x_k - x_0| = |\sum_{i=1}^{k}(x_i - x_{i-1})| \leq \sum_{i=1}^{k}|x_i - x_{i-1}| \leq \sum_{i=1}^{k}|y_i - y_{i-1}| = |\sum_{i=1}^{k}(y_i - y_{i-1})| = |y_k - y_0|$. □

***Definition** 7 (Extension):* The extension $\eta(\mathcal{M})$ (or η for brevity) of $\mathcal{M}$ is the length sum of the edges in $\mathcal{M}$.

***Lemma** 2:* In a square G, $\eta \in O(Min\{\nu\sqrt{n}, n\})$.

Proof: For a complete mesh that is constructed without applying the block rule, its extension is just the product of $\sqrt{n}$ and the number v of its constituting grid rows and columns of G. Clearly, the maximum value of v is 2ν, for example, in the case that there are no horizontally or vertically collinear SP-nodes. Therefore, the extension of the complete mesh is bounded above $O(\nu\sqrt{n})$. Because $\mathcal{M}$ is the result of edge pruning of the complete mesh structure by the blocking rule, its extension is naturally bounded above $O(\nu\sqrt{n})$ as well. This upper bounder is actually achievable, for example, when SP-nodes are all located on the same line along X-axis (or Y-axis). Note that, when $\nu > \sqrt{n}$, $\nu\sqrt{n}$ can be much larger than n in order of magnitude for large n. Furthermore, since $\mathcal{M}$ is accommodated within G, its extension η obviously never exceeds $|V| = 2n - 2\sqrt{n} = O(n)$, the total number of edges in G. Hence, $\eta \in O(Min\{\nu\sqrt{n}, n\})$. □

***Lemma** 3:* In a square G, $\eta = \Omega(\nu + \sqrt{n})$.

Proof: In $\mathcal{M}$, every SP-node has exactly four incidental edges, each of which is shared by at most two SP-nodes, and thus the number of mesh edges is not less than 2ν. Under this circumstance, because each mesh edge has length at least 1, η is bounded below $O(\nu)$. Now, let us consider a northmost SP-node p_0. If p_0 is not blocked along Y-axis, its entire residing column will be included in $\mathcal{M}$; otherwise, there must exist a blocking chain spanning the entire network along Y-axis. In either case, η is not less than $\sqrt{n}$. By the above analysis, $\eta = \Omega(\nu + \sqrt{n})$. □

Note that, the lower bound indicated by Lemma 3 is achievable, for example, in the scenario shown in Fig. 3. In this example, there are $\nu = a^2 + 12$ SP-nodes: a^2 are densely packed in the middle of G, constituting an $a \times a$ inner grid; 12 are evenly placed around the inner grid at distance $a - 1$, forming a big square that blocks the inner grid expanding. The length summation of the mesh edges is less than 6ν (in fact, it should be $6\nu - 6(a+12)$) inside the big square; on the outside, it is no more than

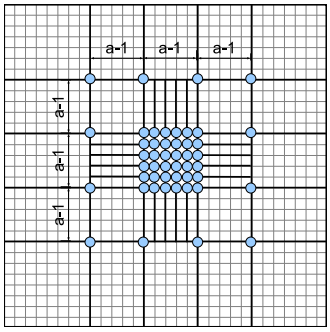


Fig. 3. An information mesh of $\eta = O(\nu + \sqrt{n})$

$8\sqrt{n}$ (in fact, it should be $8\sqrt{n} - 12(a-1)$). Thus in total is $\eta < 8\sqrt{n} + 6\nu = O(\nu + \sqrt{n})$.

Theorem 1: In a square G, the message complexity of information mesh construction is $O(\psi(G))$, where $\nu + \sqrt{n} \leq \psi(G) \leq Min\{\nu\sqrt{n}, n\}$.

Proof: If G is a synchronous environment, the paths that SP-nodes' registration messages travel are exactly the edges of $\mathcal{M}$. In this case, due to the blocking rule, a constant number (1 or 2) of registration messages are transmitted on each communication link in these mesh edges. Specifically, there are two registration messages transmitted on the middle link of two collinear SP-nodes separated by an odd number of hops (as the case with c and e in Fig. 1(b)), and one registration message over all the other links. Hence, the theorem follows immediately from Lemma 2 and 3. If G, otherwise, is an asynchronous environment, because some registration messages may be incorrectly transmitted on the links in $G - \mathcal{M}$, and revocation messages are used for consistency maintenance, the message complexity can not be lower than in a synchronous scenario. On the other hand, because SP-nodes still block messages effectively, there are at most 4 messages, 2 in each direction, transmitted on each link in the complete mesh structure, and as a consequence the message complexity can not be worse than $O(Min\{\nu\sqrt{n}, n\})$. □

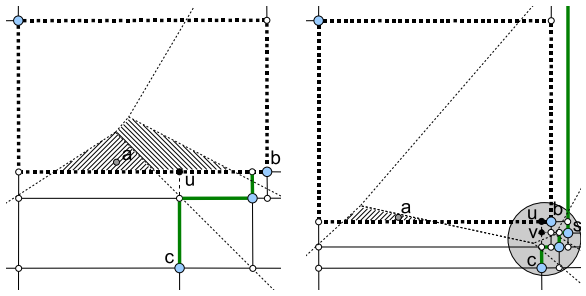
Theorem 2: In a square G, the message complexity of cross lookup is $O(\sqrt{n})$.

Proof: A cross lookup process of an SC-node is restricted within a search cell, i.e., the home cell of the SC-node. In the worst case, for example, when SP-nodes are all located on the same network border, a search cell spans the entire network, and an SC-node in the search cell will inquire all the way along its residing grid row and/or column, generating $O(\sqrt{n})$ search messages. □

Theorem 3: iMesh generates constant storage load on each sensor node.

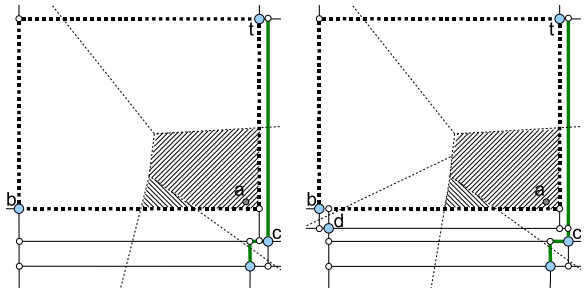
Proof: At any time, each of the sensors that constitute $\mathcal{M}$ records at most one SP-node's information from each of the four geographic directions, i.e., north, south, west, and east, due to the blocking rule. The sensors which are not part of $\mathcal{M}$ do not store any data at all. □

***Definition** 8 (Target over Closest Ratio):* For an SC a, its target over closest ratio $TCR(a)$ is defined as $TCR(a) = \frac{|aT(a)|}{|aC(a)|}$, where $C(a)$ is an SP closest to a.



(a) Barrage Case (b) Clean-Pass Case

Fig. 4. Example situations of $1 < TCR(a) \leq 2$



(a) Dirty-Pass Case (b) Isolation Case

Fig. 5. Example situations of $TCR(a) > 2$

TCR measures the distance sensitivity of iMesh. Ideally, $TCR(a)$ is equal to 1, meaning closest service selection. This happens when $C(a)$ is in $SPV(a)$, i.e., when the residing grid row and/or column of $C(a)$ is part of the perimeter of $HCell(a)$. However, due to randomized distribution of SPs, it may not always be the case. To study the distance sensitivity of iMesh-A, all the possible violation situations where TCR > 1 need to be identified. By an exhaustive search, it is observed that all violations are the variants of the following four general cases:

1) *Barrage case*: $C(a)$ is chain-blocked by an SP in $SPV(a)$, before its blocking chain passes around $HCell(a)$;
2) *Clean-Pass case*: the blocking chain of $C(a)$ passes around $HCell(a)$ at the corner where an SP is located;
3) *Dirty-Pass case*: the blocking chain of $C(a)$ passes around $HCell(a)$ at the corner where no SP exists, and a composing mesh edge of $HCell(a)$ intersects the residing mesh edge of $C(a)$;
4) *Isolation case*: the blocking chain of $C(a)$ passes around $HCell(a)$ at the corner where no SP exists, and no composing mesh edge of $HCell(a)$ intersects the residing mesh edge of $C(a)$.

Figures 4 and 5, where irrelevant SC-nodes are hidden and SP-nodes are represented by solid big dots, illustrate the above four general violation cases. In the two figures, the home cell $HCell(a)$ of SC-node a is emphasized by broken thick lines, and the blocking chain of $c = C(a)$ is highlighted by continuous thick lines; broken thin lines indicate the Voronoi diagram created using SP-nodes, and shaded areas are the places where TCR > 1.

***Lemma** 4:* In Barrage case, $TCR(a) \leq 2$.

Proof: Let b be the SP-node in $SPV(a)$ that chain-blocks c (i.e., $C(a)$). We have $|aT(a)| \leq |ab|$. Without loss of generality, assume that the chain of blocking happens along Y-axis. Let u be the intersection node of the residing grid column of c and the residing grid row of b, as shown in Fig. 4(a). By Lemma 1, $|bu| \leq |cu|$. Observe that angle $\angle cua$ can not be acute in any case. Thus ca is the longest side in triangle Δcua. Namely, $|cu| < |ca|$ and $|ua| < |ca|$. Then $|ab| \leq |bu| + |ua| \leq |cu| + |ua| < |ca| + |ca| = 2|ca|$. Because $|aT(a)| \leq |ab|$, we have $|aT(a)| \leq 2|ca|$. □

***Lemma** 5:* In Clean-Pass case, $TCR(a) \leq 2$.

Proof: Without loss of generality, assume that the blocking chain of c (i.e., $C(a)$) is towards $HCell(a)$ along Y-axis. Let b be the SP-node in $SPV(a)$ where the blocking chain passes around $HCell(a)$. Further, let s be the SP node in the blocking chain that (chain) blocks b along X-axis. Such an s must exist because, otherwise, this is not Clean-Pass case but Barrage case. Denote by u the the intersection node of the residing grid column of c and the residing grid row of b, and by v the intersection node of the residing grid column of c and the residing grid row of s, as shown in Fig. 4(b). By Lemma 1, $|sv| \leq |vc|$. Unambiguously, $|bu| \leq |sv| \leq |cv| \leq |cu|$. From this point, the lemma follows from the same proof as Lemma 4. □

***Lemma** 6:* In both Dirty-Pass case and Isolation case, $TCR(a)$ may be larger than 2 but must not be larger than $\frac{d(G)}{|aC(a)|}$, where $d(G)$ is the spatial diameter of G.

Proof: Examine Fig. 5, where $t = T(a)$, $c = C(a)$, and the blocking chain of c is along Y-axis. By observation, $|at|$ is already greater than $2|ac|$, namely, $TCR(a) > 2$, and there is no restriction on the distance from t to the residing grid row of c. If we move b (together with d in Fig. 5(b)) and t far apart from a while maintaining their blocking relation, then $|at|$ could be way larger than $2|ac|$. On the other hand, because no pair of nodes have their separation larger than $d(G)$, we have $|at| \leq d(G)$ and consequently $TCR(a) = \frac{|at|}{|ac|} \leq \frac{d(G)}{|ac|}$. □

In [20], the authors claimed that the blocking rule allows a node to discover a service that is at a distance no larger than $\sqrt{2}$ times the distance from the closest. Lemma 6 shows that this claim is false.

6 Complete iMesh Protocol

A distance-sensitive service discovery algorithm is expected to guarantee nearby service selection, that is that $TCR(a) \leq 2$ for any SC-node a. By Lemma 6, iMesh-A may not satisfy this expectation in Dirty-pass case and Isolation case. In this section, we will present the complete version of iMesh, iMesh-B, which achieves major improvement on distance sensitivity over, but has the same complexity as, iMesh-A.

Define the *territory* of an arbitrary SP-node c as the area in which c can be discovered by local SCs through the cross lookup method. The larger the territory of c, the higher its probability of being discovered, and thus the better the distance sensitivity of iMesh. However, in iMesh-A, the size of an SP's territory is strictly restricted by the blocking rule for message saving purpose. Figure 6 redraws the Dirty-Pass situation given in Fig. 5(a). In this figure, the territory of SP-node c is represented by the light gray area, which is actually the aggregation of the mesh cells adjacent by the registration paths (marked by arrowed hollow lines) of c.

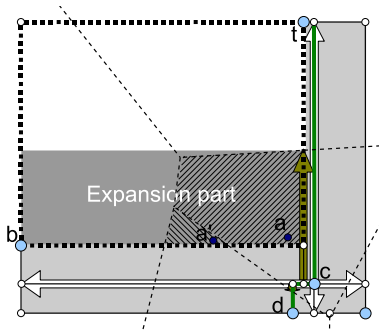


Fig. 6. The effect of the extension rule

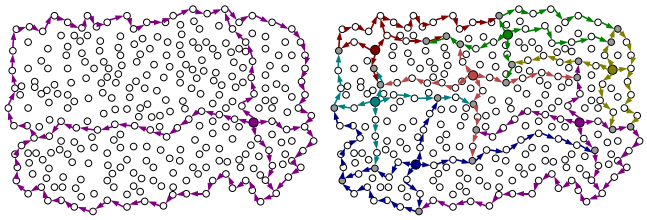
To improve the distance sensitivity of iMesh, territory expansion is necessary. In iMesh-B, information mesh is built not only by the blocking rule but also by an extension rule. The new extension rule enables SPs to expand their territories in the case of orthogonal blocking. The formal definition of the extension rule is given below:

***Rule** 2 (Extension Rule):* A node u at which an SP-node a orthogonally blocks another SP-node b sends the information of a to b along the backward path from which it receives b's information. The information of a does not travel all the way to b but stops at the point where the path intersects the bisector between a and b.

In Fig. 6, the transmission paths of the extension messages of SP-node c are highlighted by arrowed solid lines, and the dark gray area is the expansion part of the territory of c. By observation, c's territory expands into the home cell $HCell(a)$ of SC-node a, and a becomes able to discover c as a result. Consider another SC-node a' that shares the same home cell with a. The closest SP-node $C(a')$ to a' is d in the blocking chain of c. In iMesh-A, $TCR(a')$ could be way greater than 2 (if $HCell(a)$ is very large) according to Lemma 6. On the contrary, in iMesh-B, we have $T(a') = c$ and then $TCR(a') \leq 2$ following a similar proof as Lemma 4.

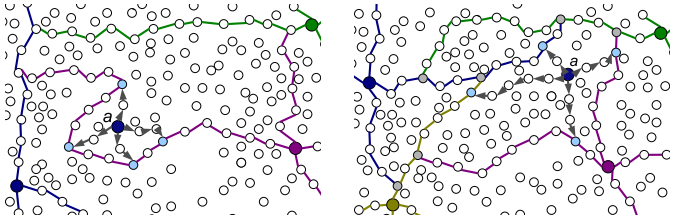
By the above examples, the extension rule eliminates Dirty-Pass case, and thus the negative Lemma 6 only holds partially for iMesh-B. By definition, the extension rule does not either change the structure of, or remove data from, the information mesh. Therefore, Lemma 2, 3, 4 and 5 and Theorem 2 still hold for iMesh-B. In addition, it is not difficult to verify that Theorem 1 and 3 are also applicable to iMesh-B. During an information revocation process of iMesh-B, information should be removed not only along its regular propagation paths but also along its extension paths.

In summary, the extension rule enables iMesh-B



(a) One SP node (b) Seven SP nodes

Fig. 7. Information mesh in an arbitrary sensor network



(a) Curly edge (b) Irregular shape

Fig. 8. Cross lookup in an arbitrary sensor network

to achieve improved overall distance sensitivity over iMesh-A at very low cost. Its effect and cost will be seen clearly later, through simulation in Sec. 8.

7 Implementation Detail

In an arbitrary sensor network, there is no grid structure that we can make use of for information mesh construction and cross service lookup. So we accomplish our goal by using routing protocol GFG [3], [7], which is known for its guaranteed packet delivery and has been used to support quorum formation in Quorum [19].

7.1 Information mesh construction

An arbitrary SP generates four registration messages carrying its location information respectively for the four directions, north, south, west, and east. Then it sends them to the corresponding directional foremost neighbors, namely, the northbound message to the northmost neighbor, and the southbound message to the southmost neighbor, and so on. These registration messages are retransmitted by receiver nodes following GFG. More specifically, upon receiving a registration message, a node retrieves the embedded SP-node information from the message, stores it in local storage, records the message's designated transmission direction, and then greedily forwards the message to its foremost neighbor in that direction. When a registration message reaches a void area, it is switched to the *face routing* mode and passed around the void area in the clockwise (counterclockwise) direction by the left (resp., right) hand rule. Greedy forwarding resumes whenever possible.

If the source is the only SP in the network, due to the absence of network boundary information and the nature of GFG, a registration message will finally stop at the globally foremost node in its transmission direction, and its transmission path will include the entire network boundary, as shown in Fig. 7(a), where the registration paths (i.e., the transmission paths of the registration messages) of the only SP are highlighted by arrowed lines. In the case that there is more than one SP in the network, SPs' registration paths intersect one and another inside the network and/or overlap on the network boundary. For two intersecting registration paths, they will be either in a *node-sharing situation* or in a *link-crossing situation*. In the former case, the two paths intersect at a common node, while in the latter case, they have a pair of crossover links. A link-crossing situation can be locally (without extra message transmission) transformed to a node-sharing situation.

By above analysis, for any two different SPs, their registration paths are guaranteed to have some node in common. The common node applies the blocking rule and the extension rule as in grid networks. An information mesh is established as a result. Figure 7(b) shows an information mesh created by seven SP nodes (colored differently) in an arbitrary sensor network.

7.2 Distance-sensitive service lookup

The implementation of cross lookup is simple. An SC node a sends a query message to its foremost neighbors in the four directions. Each of these messages is retransmitted through protocol GFG and stops at the first receiver node that resides on the information mesh. Then this receiver node sends a a positive reply containing its locally stored SP information. However, if there does not exist any SP in the network, which is possible when all the SP nodes become unavailable, such a query message will reach a boundary node b and then traverse the entire network boundary starting from there (by the property of GFG). In this case, once the query message gets back to b along the network boundary, b sends a a negative reply, indicating the failure of service lookup. If all the replies that it receives are positive, a can easily determine its target service provider $T(a)$; if all of them are negative, it knows that its service lookup fails. In any other case (some failure must occur), a may re-search in the direction from which it received a negative reply so as to ensure the discovery of $T(a)$.

If no void area appears in network topology, greedy forwarding will never fail. As a result, every cell in the information mesh has a rectangular shape, and the cross lookup method always works. In the presence of void areas, messages are routed along their perimeters, causing zigzag-line message transmissions and thus possible cross lookup failures. Figure 8, where arrowed gray lines indicate search paths, shows two counter-examples. In Fig. 8(a), the search messages of SC node a all hit the same curly edge of $HCell(a)$; in Fig. 8(b), $HCell(a)$ is composed of five edges, causing that no search message reaches the northmost edge. Apparently, a fails to find its true target service provider in the two cases.

In such undesired situations, an alternate *perimeter lookup* method can be used. An SC a sends a search

message to an arbitrary direction by protocol GFG. This search message will hit the perimeter of $HCell(a)$ at certain node, called *entry node*, which then retransmits the message along the cell perimeter, e.g., in the clockwise direction. The search message picks up the information of the closest SP that it sees during perimeter traversal. After it returns back to the entry node, it has found the target service provider $T(a)$ of a. Therefore, upon receiving the search message back, the entry node returns the message to a as reply. A special case is that an SC is riding on the information mesh. In this scenario, the SC performs perimeter lookup in its every sub home cell. Since it is already on the cell perimeter, it can start perimeter traversal directly.

7.3 Tolerating node failures

Node failure may lead to loss of SP information. Similar to the fault-tolerance approach employed in Quorum [19], iMesh uses *thick registration paths* to increase information redundancy and consequently its fault-tolerance capability. Specifically, during the information mesh construction process, SP nodes' registration messages are transmitted along paths of certain thickness. For thickness 1, all the nodes that overhear a registration message store the embedded SP location information; for thickness k, these overhearing nodes are required to broadcast the registration message to $k-1$ hops.

Another impact from node failure is loss of control messages for service lookup. To tolerate such message loss, iMesh uses a simple yet effective fault-tolerance approach, *transmission retrial*. During a service look up process, if the SC node does not get any reply to its lookup message, it backs off for a while and then retries.

It is possible that an SP node suddenly fails without notification. To handle SP failure, iMesh requires that the neighbors of each SP monitor the SP node's aliveness, e.g., by listening to a periodic beacon message. Once they find the SP node fails, they immediately start a revocation process (refer to Sec. 4.1) to remove the SP's information from the information mesh.

7.4 Keeping storage load constant

By the information mesh construction method, the storage load of a node is subject to the number of its incidental mesh edges. The number of nodal incidental mesh edges is bounded above node degree, which is an inconstant value (equal to $n-1$ in the worst case). To keep storage load constant, an node where the blocking rule applies does not store the location information that it blocks but adds a mark (nearly at no extra storage cost) to the neighbor from which it receives the blocked information, such that it can later find the blocked information without actually storing it.

8 Performance Evaluation

As summarized in Sec. 2, existing service discovery algorithms and adoptable techniques usually rely on global computation and therefore generate large message overhead, and they may in addition impose inconstant storage load on network nodes and/or induce bottleneck problem in the network. Our proposed algorithm iMesh however has obvious advantages in all these aspects. It aims to yield optimal (constant) per node storage load and avoid long service registration/lookup paths while providing satisfactory distance-sensitivity.

For a fair comparative study, we excluded centralized, distributed, or semi-distributed algorithms including globalized-structure-based algorithms (e.g., [10], [11]), flooding-based algorithms (e.g., [6], [8]), data-centric storage based algorithms (e.g., [4], [16]), and hashing-based or hashing-quorum-based location service algorithms (e.g., [12], [18]). We considered only competing localized algorithms. This left us with quorum approaches; thus we chose the representative algorithm Quorum [19].

Because Quorum and iMesh share the same routing protocol GFG [3], [7] at their core for message transmission, their message overhead has similar relative trends both in dense networks and in sparse networks. Quorum always guarantees closest service selection at the cost of messages; whereas, iMesh has degraded performance in distance sensitivity in sparse networks in exchange for reduced message overhead. It is because, as node density decreases, information mesh structure tends to be increasingly irregular and gradually loses its proximity property. However, in this article we focus on the theoretical aspects of iMesh in general case rather than the extreme boundary condition study.

For the above reasons, we simulated Quorum, iMesh-A (a generalization of GCLP [20]) and iMesh-B in grid networks. We believe our simulation support fair comparative evaluation of the two algorithms. Indeed, grid networks have already been adopted in literature [5], [21] for effective performance analysis of wireless ad hoc networks. Importantly, our simulation also provides valid verification of our theoretical findings. As we will see in the following, iMesh has considerably low message overhead in general when compared with Quorum, and iMesh-B guarantees closest service selection with high probability, larger than 97%, and nearby service selection with very high probability, larger than 99%, significantly improving the distance sensitivity of iMesh-A at negligible communication cost.

8.1 Evaluation metrics

We study the message overhead of iMesh and Quorum using the following metrics:

- Total Number of Construction Messages (TNCM): the total number of messages transmitted in the network for information mesh construction;
- Number of Construction Messages per SP (NCMSP): the average number of messages generated by an SP for the purpose of information mesh construction;
- Number of Search Messages per SC (NSMSC): the average number of lookup messages generated by an arbitrary SC (reply messages are not counted);

As Quorum guarantees closest service selection (i.e., TCR $= 1$), the following metrics are for iMesh only:

- Average TCR and Peak TCR: the average TCR and the peak TCR of all the possible SCs in the network.
- PTCR1, PTCR2, and PTCR3: the probabilities of TCR $= 1$, $1 <$ TCR ≤ 2, and TCR > 2.

8.2 Simulation setup

We simulated Quorum, iMesh within a custom network simulator. we have limited our simulation to grid networks of size $32 \times 32 (\approx 1,000)$, $45 \times 45 (\approx 2,000)$, $\cdots$, $100 \times 100 (= 10,000)$. SPs are randomly scattered in the network. We chose the settings of the Percentage of SPs (PSP) in the network varying from 0.1% to 1.5%. For each setting, we executed the protocols over 100 randomly generated SP distribution scenarios to get average results. We run two streams of simulation for different objectives.

In the first stream, we use *static* networks where actors do not move, and we aim to verify our theoretical findings including mesh construction cost and distance sensitivity. Since a static network can be viewed as a snapshot of a dynamic network, the distance sensitivity study will be valid also for *dynamic* networks. Note that physically available SPs might not be informatively available due to information propagation delay and vice versa. This is a common problem for any service discovery algorithm. Hence, our distance sensitivity evaluation counts only for SPs available both physically and informatively. We run two sets of experiments. In the first set, the network is set to be a synchronous environment with simultaneous execution and unified link delay; in the second set, the network is configured to be an asynchronous environment where SP-nodes start the protocols maximally 30 simulated time units off each other, and each communication link has transmission delay of 10 simulated time units at most.

In the second stream, we comparatively study the construction cost of the information structures of these protocols. Because Quorum is irrelevant to network synchrony, we use asynchronous networks in simulation. We consider both *static* networks and *dynamic* networks and investigate the impact of actor mobility. In our simulated dynamic networks, there are 50 randomly scattered SCs requesting services over a period of 1000 simulated time units. Scenarios with different number of SCs are also tested; the results are similar and thus not presented here. According to [23], data travels at least an order of magnitude faster than typical mobile nodes. In our simulation, SPs move at a speed $10-50$ times slower than the transmission speed of a data packet. Before moving, SPs delete their own information from the network; after delivering service, they stay where they are and redistribute their information in the network.

8.3 Message overhead of iMesh

Below we study the communication cost of the two versions of iMesh in a synchronous environment and in an asynchronous environment with $10,000$ nodes. In Sec. 8.5, we will analyze the message overhead of iMesh in other sized networks, in comparison with Quorum [19].

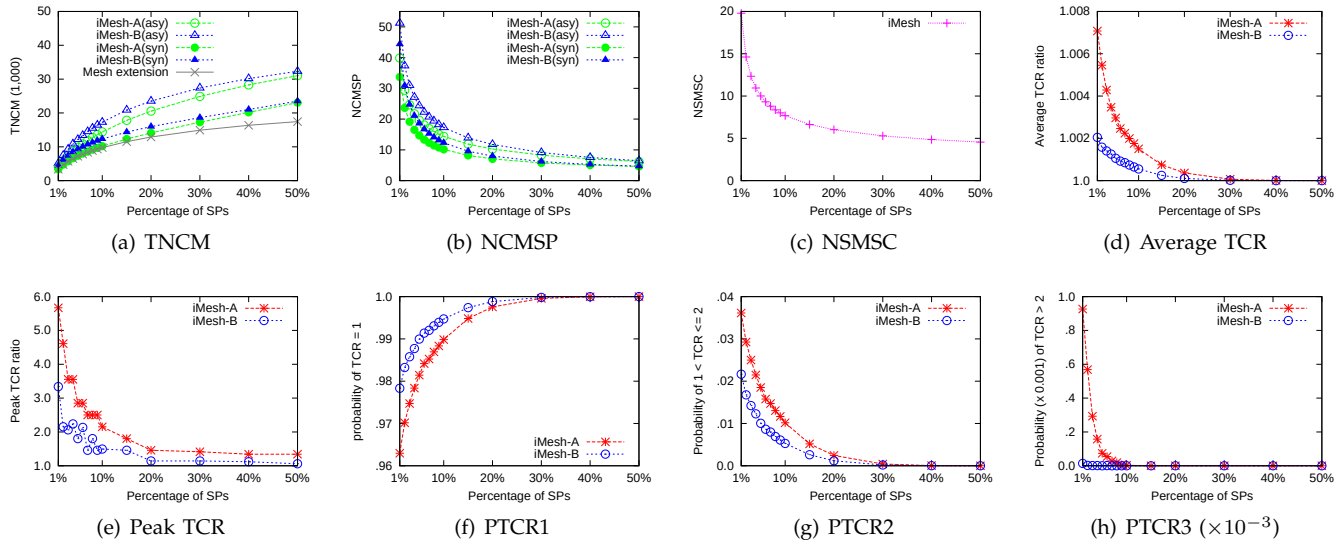
Figure 9(a) show the TNCM of iMesh in relation with PSP. For reference, mesh *extension* (Definition 7), which has relation (See Theorem 1) to TNCM, is also drawn in the figure. As PSP grows, the information mesh has a more and more complex structure and is therefore expected to exhibit an increasing extension and a growing construction message overhead. The expectation is confirmed by the ascending trend of the curves in the figure. The small gap between the TNCM curves for iMesh-A and iMesh-B in either environment indicates that the overhead of the *extension rule* (Rule 2) is minor. And, from the figure we can also see that TNCM will never exceed some constant times mesh extension. This observation verifies Theorem 1.

Examine again Fig. 9(a) and pay attention to the difference of TNCM in the two environments. It is observed that TNCM is always higher in the asynchronous environment than in the synchronous environment. This is due to the extra messages used for eliminating the information inconsistency caused by asynchrony. Further, as PSP grows in either environment, TNCM curves deviate more and more from the curve of mesh extension, and the TNCM of iMesh-B approaches to that of iMesh-A closer and closer. It is because, when there are more SP-nodes, the situation that two collinear SP-nodes are an odd number of hops away happens more often, causing more overlapping registration messages on mesh edges, and the mesh cell has smaller size, leading to the reduction of the travel distance of extension messages.

Figure 9(b) displays the NCMSP of iMesh as a function of PSP. We can see that NCMSP drops and approaches to 4 as PSP goes up. It is because, when SP density increases, an SP-node's registration message travels a decreased hop-distance (on average) in each direction before being blocked, and the travel distance can be as low as 1-hop, resulting in merely 4 registration messages in the extreme case. As shown in the figure, each SP-node uses slightly more construction messages in the asynchronous environment than in the synchronous environment due to the cost of information consistency maintenance; iMesh-B generates slightly larger NCMSP than iMesh-A in both environments, which again implies the negligible message cost of the extension rule.

Figure 9(c) depicts the NSMSC of iMesh, which is irrelevant to synchrony and to the application of the extension rule, as a result of PSP. It is observed that NSMSC drops and approaches to 4 as PSP climbs. It is because, when SP density increases, an SC-node's search message travels a decreased hop-distance (on average) in each direction before finding an SP, and the travel distance can be as low as 1-hop, resulting in merely 4 search messages in the extreme case.

To sum up, the results given in Fig. 9(a) – 9(c) clearly indicate that iMesh use a considerably small, compared with network size, number of messages for service regis-



(a) TNCM (b) NCMSP (c) NSMSC (d) Average TCR

(e) Peak TCR (f) PTCR1 (g) PTCR2 (h) PTCR3 ($\times 10^{-3}$)

Fig. 9. iMesh in a network of $10,000$ nodes. Abbreviation "asy" ("syn") indicates the asynchronous (resp., synchronous) nature of the network. The absence of such an abbreviation implies that a metric is irrelevant to network synchrony.

tration and service lookup. At a detailed level, iMesh-B generates slightly larger message overhead than iMesh-A; but the difference is actually negligible.

8.4 Distance sensitivity of iMesh

We now study the distance sensitivity of iMesh, which is regardless of the (synchronous or asynchronous) nature of the execution environment. In our simulation, iMesh exhibits consistent distance sensitivity in different sized networks. Due to space limitation, we present the results from a network of size $10,000$ only.

Figures 9(d) and 9(e) respectively show the average TCR and the peak TCR in relation with PSP. From Fig. 9(d) we can see that the average TCR is nearly equal to 1 in all the PSP cases. This is because of the low probability of $TCR > 1$. In both of the two figures, the curves decline and approach to 1 closer and closer as PSP increases. This phenomenon is due to the decreasing probability of TCR > 1. According to the two figures, iMesh-B always has better distance sensitivity than iMesh-A. It is because the extension rule effectively eliminates Dirty-pass case (see Sec. 5).

Figures 9(f) – 9(h) depict PTCR1, PTCR2 and PTCR3 as a function of PSP. By Fig. 9(f), both iMesh-A and iMesh-B provide closest service selection with high probability, respectively larger than 96% and 97%. By Fig. 9(g) and 9(h), both PTCR2 and PTCR3 quickly drop down nearly to 0 as soon as PSP increases to 10%. The three figures together indicate that iMesh guarantees nearby service selection with very high probability, larger than 99%, in all the PSP cases, and they also confirm our analysis about TCR value in the previous paragraph.

Figures 9(f) – 9(h) also imply that iMesh-B always has better distance sensitivity than iMesh-A. It is because iMesh-B eliminates the Dirty-Pass case by the extension rule. Examine the part for PSP in range $1\% - 10\%$ in Fig. 9(h). The PTRC3 of iMesh-A and iMesh-B are both extremely low, smaller than $O(10^{-3})$. In particular, due to the extension rule, iMesh-B's PTCR3 is significantly lower, in order of magnitude, than that of iMesh-A.

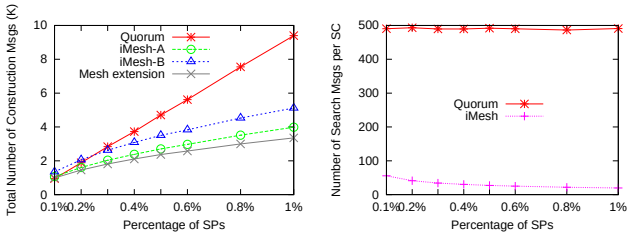
The experimental results shown in Fig. 9(d) – 9(h) indicate that iMesh has satisfactory distance sensitivity. Compared with iMesh-A, iMesh-B performs much better both in closest service selection and in nearby service selection, and has lower probability of undesired distant service selection. By Sec. 8.3, iMesh-B achieves these advantages over iMesh-A at negligible message cost.

8.5 iMesh v.s. Quorum

As Quorum is irrelevant to network synchrony and guarantees closest service selection, we will only study the difference of iMesh and Quorum in message overhead (measured by TNCM/NCMSP and NSMSC) in asynchronous environments. Let us first examine Fig. 10 and 11, which show iMesh v.s. Quorum with varying PSP in a network of size $10,000$ and one of size $1,000$. In the two figures, NCMSP is not shown since it is equivalent to TNCM.

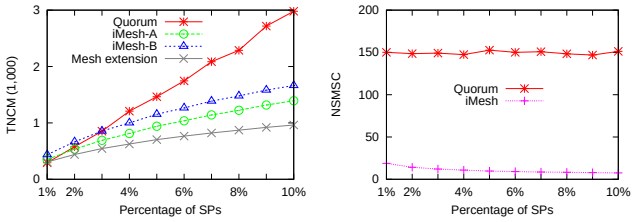
Figures 10(a) and 11(a) depict TNCM as a function of PSP. It is observed that, as PSP increases, TNCM climbs quickly in Quorum but at a very slow speed in iMesh, almost starting from the same point. It is because, an SP's registration message always propagate across the entire network in Quorum; but, as discussed in Sec. 8.3, it travels a shorter and shorter distance due to message blocking in iMesh when PSP ascends.

Figures 10(b) and 11(b) display NSMSC in relation with PSP. It is seen that Quorum generates almost constant but dramatically larger NSMSC, regardless of PSP, when compared with iMesh. This phenomenon is reasonable because an SC in Quorum has to search across the entire network and along the whole outer boundary



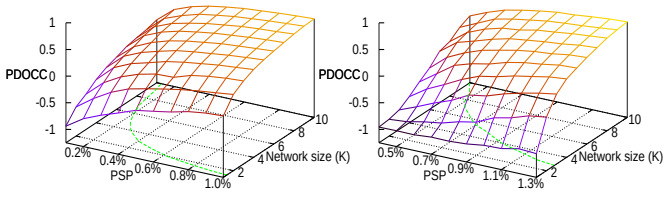
(a) TNCM (b) NSMSC

Fig. 10. iMesh v.s. Quorum in a network of $10,000$ nodes

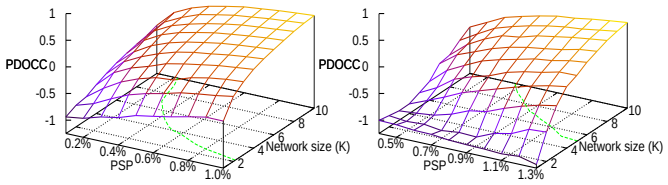


(a) TNCM (b) NSMSC

Fig. 11. iMesh v.s. Quorum in a network of $1,000$ nodes



(a) iMesh-A in static networks (b) iMesh-A in dynamic networks



(c) iMesh-B in static networks (d) iMesh-B in dynamic networks

Fig. 12. PDOCC between iMesh and Quorum

for a closest SP; while in iMesh, an SC does not query along the outer boundary of the network, and its service lookup operation is restricted within a search cell, whose size generally decreases as PSP increases.

The similarity in the trend of the curves in Fig. 10 and 11 implies the performance consistency of iMesh in different-sized networks. Let us now turn our attention to their difference, which is actually more of our interest. We will comparatively study their message overhead in static networks and in dynamic networks. Note that, whether the network is static or dynamic, Quorum has outer boundary traversal included in every service lookup process and thus always yields larger NSMSC than iMesh. Under this circumstance, we are interested only in their difference in TNCM.

In Fig. 10, Quorum outperforms iMesh in TNCM (i.e., message overhead for service registration) for very small PSP ($< 0.15\%$ in iMesh-A, $\approx 0.25\%$ in iMesh-B.) The reason is obvious: when message blocking rarely happens, four-direction service registration of iMesh leads to more messages in total than two-direction service registration of Quorum. Similar crossover is observed in Fig. 11, but in a different PSP range ($< 0.2\%$ in Mesh-A, $< 0.3\%$ in iMesh-B.) This difference is important in that it tells us the conditions under which iMesh can be used to replace Quorum at best.

To ease our study on crossover point of iMesh and Quorum in service registration overhead, we introduce *Proportional Difference of Construction Cost (PDOCC)*:

$$\text{PDOCC} = \frac{\text{TNCM(Quorum)} - \text{TNCM(iMesh)}}{\text{TNCM(Quorum)}}$$

By definition, the larger PDOCC, the more advantageous iMesh; if PDOCC is negative, then iMesh may not be as efficient as Quorum. A crossover point of iMesh and Quorum is a point where PDOCC $= 0$. Figures 10 and 11 suggest that, PDOCC starts to exhibit a positive value with a deferred PSP threshold (in other words, iMesh gradually becomes less and less advantageous than Quorum) as network size descends, and that the degradation would be very slow compared with the decreasing speed of the network size. This change is expected both in static networks and in dynamic networks.

Our simulation confirms the above expectation about PDOCC, as shown by the 3D surfaces in Fig. 12, where the X, Y and Z axises respectively represent PSP, network size and PDOCC. The crossover points (the points where PDOCC $= 0$) of iMesh and Quorum are plotted on the X-Y plan. They indicate the boundary beyond which an X-Y combination would yield a positive PDOCC. For example, by Fig. 12(a) and 12(c), in a static network of $6,000$ nodes, iMesh-A produces positive PDOCC when PSP $> 0.2\%$, while iMesh-B does not until PSP $> 0.35\%$. This slight different between iMesh-A and iMesh-B is obviously due to the cost of the extension rule.

In dynamic networks, SPs remove their location information from the network before moving and redistribute it (latest) after service delivery. Frequent information deletion and re-distribution will amplify the negative impact of four-direction service registration on message overhead in iMesh, and cause deferred crossover points. This is confirmed by the results shown Fig. 12(b) and 12(d). Let us again focus on network size 6000. We find that, in our dynamic-network-based simulation, iMesh-A produces positive PDOCC when PSP $> 0.65\%$, while iMesh-B does not until PSP $> 1.1\%$. To have an overall view, we just compare the positions of the curves on the XY plane with their counterparts in Fig. 12(a) and 12(c). Notice that the difference of iMesh-A and iMesh-B is also slightly enlarged in dynamic networks.

9 An Application Example of iMesh

Sensor relocation [13], [14], [22] is an import research topic in robotic sensor networks. It deals with timely

patching of sensing holes through autonomous node movement, and can be used as a fault-tolerance approach to prevent coverage loss caused by node failures. Generally speaking, a solution algorithm fulfills two tasks: (1) *replacement discovery*: finding a predetermined redundant sensor as the replacement of a failed sensor; (2) *replacement migration*: migrating the discovered replacement to the position of the failed sensor.

To minimize migration distance and save energy, replacement node should be a redundant sensor closest to the failed node. In this sense, *replacement discovery is a distance-sensitive service discovery problem*, where redundant sensors are service providers, and thus can be accomplished by our proposed algorithm iMesh. Because the sensor relocation problem is out of scope of this article, below we will briefly show how to use iMesh to build a sensor relocation algorithm in principle.

For brevity, we denote redundant sensors by *R-nodes* and non-redundant (i.e., active) sensors by *A-nodes*. R-nodes are randomly scattered in the network. At initiation, each R-node spontaneously takes a nearest A-node as proxy; then they fall "asleep" to save energy. Proxy nodes execute algorithm iMesh on the behalf of their delegated R-nodes to construct an information mesh. Upon an ordinary (i.e., non-proxy) A-node failure, the A-nodes neighboring the failed node cooperate to discover a replacement by executing iMesh. The replacement is defined as the nearest delegated R-node of the target replacement proxy (i.e., target service provider in iMesh) of the failed node.

For replacement discovery, the two lookup methods, i.e., cross lookup and perimeter lookup, may be selectively used. In cross lookup, the northmost, the southmost, the eastmost, and the westmost neighbors of the failed A-node, as *servers*, send four search messages respectively to the north, the south, the east, and the west. After getting replies, they exchange their discovery results by underlying routing protocol to find the target replacement proxy. In perimeter lookup, only one neighbor, say the northmost, acts as the server of the failed A-node and initiates the lookup process. The server closest to the target replacement proxy communicates with that proxy node and triggers subsequent replacement migration process. For efficient replacement migration method, one may refer to [14].

10 Multi-service Scenarios

In previous sections, iMesh was presented in the context of single-service networks, which is however not a common setting in practice. When there is more than one type of service provided in the network, a *multi-layer* information mesh can be constructed to support service discovery. That is, the same type of service providers together constitute a mesh layer, and different layers correspond to different types of service. For a network with $k \geq 1$ service types, the height, i.e., the number of layers, of the information mesh is equal to k. Note that a node offering multiple types of services will appear in more than one layer of the information mesh.

With cares, the message complexity of constructing a multi-layer information mesh can be made less than the summation of the message complexity of building every single mesh layer separately. For instance, an SP-node shared by t number of mesh layers does not necessarily distribute its location information t times. Instead, it attaches t bits to the information to indicate its offered services. The information is virtually blocked in one layer by flipping the corresponding bit; it physically stops propagating when all the attached bits are flipped or when it reaches the network border. By this means, the SP-node fulfills its construction duty in all its residing layers simultaneously, thus saving a considerable number of messages. An obvious coarse upper bound of the message complexity is $O(\nu\sqrt{n})$. The study of precise message complexity is not included in this work.

With the multi-layer information mesh, when a node wants to discover a particular type of service, it just needs to perform service lookup in the corresponding layer as if it was in a single-service network. In this way, the distance sensitivity and the service lookup message overhead of iMesh naturally stay unchanged. Because a node shared by t ($1 \leq t \leq k$) mesh layers has to store a constant amount of information for each of its residing layer (by Theorem 3), it has $O(t)$ storage load in total. Apparently, $O(t) \leq O(k)$. Since k is usually a known (small) value at the network deployment time, iMesh still yields constant per node storage load.

11 Conclusions

In the future, we will simulate iMesh in sensor and actor networks with different node densities and in the presence of holes in the region. We wish to study the boundary conditions for the superiality of iMesh over Quorum [19]. We will also study the effectiveness of iMesh in solving the sensor relocation problem, by evaluating the iMesh-based sensor relocation protocol [14] in comparison with existing competing solutions.

iMesh assumes that actors (i.e., service providers) have the same communication radius as sensors (i.e., service consumers). Since actors are usually resource-rich nodes in practice, they may however have higher energy level and thus larger (possibly adjustable) communication range. It will be an interesting subject for future research to modify iMesh so as to take full advantage of the high capacity of actors. In iMesh, undesired distant service selection is still possible, even though with very low probability. To achieve perfect nearby service selection guarantee, we may consider to use, for example, triangular mesh, instead of square mesh as service directory. Further improvement of distance sensitivity of iMesh is another possible direction of future work.

In iMesh, actors become unavailable while moving. This is justified by the fact that an actor may physically serve only one sensor at a time. However, it may be

desired to allow moving actors to change their service targets so as to increase service availability in emergency cases. One simple solution is to keep a moving actor's information in the information mesh and maintain a chain of pointers along the actor's trajectory. When a service request reaches the position where the actor joins the information mesh, it is redirected to the actor along the pointer chain. This solution however has two main drawbacks. Firstly, it degrades the distance sensitivity of iMesh, as the information mesh no long reflects actors' latest positions; secondly, it fails if no node exists along actor trajectory (for example, when an actor is passing through a void area). This warrants further investigation.

ACKNOWLEDGMENTS

This work was partially supported by NSERC Strategic Grant STPSC356913-2007B and UK Royal Society Research Merit Award. The authors would like to thank anonymous reviewers for their valuable criticism.

REFERENCES

[1] I. F. Akyildiz and I. H. Kasimoglu. "Wireless sensor and actor networks: research challenges". *Ad Hoc Networks*, 2(4):351–367, 2004.
[2] F. Aurenhammer and R. Klein. "Voronoi Diagrams". http://www.pi6.fernuni-hagen.de/publ/tr198.pdf.
[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks". *Proc. ACM DIALM*, pp. 48–55, 1999.
[4] Q. Fang, J. Gao, and L. J. Guibas. "Landmark-Based Information Storage and Retrieval in Sensor Networks". *Proc. IEEE INFOCOM*, pp. 286–297, 2006.
[5] G. Ferrari and O. K. Tonguz. "MAC protocols and transpan capacity in ad hoc wireless networks: Aloha versus PR-CSMA". *Proc. IEEE MILCOM*, pp. 1311–1318, 2003.
[6] C. Frank and H. Karl. "Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks". *Proc. ACM MSWiM*, pp. 105–114, 2004.
[7] H. Frey and I. Stojmenovic. "On Delivery Guarantees of Face and Combined Greedy-Face Routing Algorithms in Ad Hoc and Sensor Networks". *Proc. ACM MobiCom*, pp. 390–401, 2006.
[8] Z. Gao, L. Wang, M. Yang, and X. Yang. "CNPGSDP: An Efficient Group-based Service Discovery Protocol for MANETs". *Computer Networks*, 50(16):3165–3182, 2006.
[9] Z. Gao, Y. Yang, J. Zhao, J. Cui, and X. Li. "Service Discovery Protocols for MANETs: A Survey". *Proc. MSN*, LNCS 4325, pp. 232–243, 2006.
[10] M. Klein and B. K. Ries. "Multi-layer Clusters in Ad Hoc Networks - an Approach to Service Discovery". *Proc. Networking Workshops on Web Engineering and Peer-to-Peer Computing*, pp. 99–110, 2002.
[11] U. C. Kozat and L. Tassiulas. "Service Discovery in Mobile Ad Hoc Networks: an Overall Perspective on Architectural Choices and Network Layer Support Issues". *Ad Hoc Networks*, 2(1):23–44, 2004.
[12] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. "A Scalable Location Service for Geographic Ad Hoc Routing". *Proc. ACM MobiCom*, pp. 120–130, 2000.
[13] X. Li and N. Santoro. "ZONER: A ZONE-based Sensor Relocation Protocol for Mobile Sensor Networks". *Proc. IEEE WLN*, pp. 923–930, 2006.
[14] X. Li, N. Santoro, and I. Stojmenovic. "Mesh-based Sensor Relocation for Coverage Maintenance in Mobile Sensor Networks". *Proc. UIC*, LNCS 4611, pp. 696–708, 2007.
[15] A. N. Mian, R. Beraldi, and R. Baldoni. "Survey of Service Discovery Protocols in Mobile Ad Hoc Networks". TR 4/06, Universit degli Studi di Roma La Sapienza, Italy, 2006.
[16] S. Ratnasamy, B. Karp, L. Yin, and F. Yu. "GHT: A Geographic Hash Table for Data-Centric Storage". *Proc. ACM WSNA*, pp. 78–87, 2002.
[17] N. Santoro. *Design and Analysis of Distributed Algorithms*. Wiley, 2007.
[18] R. Sarkar, X. Zhu, and J. Gao. "Double Rulings for Information Brokerage in Sensor Networks". *Proc. ACM MobiCom*, pp. 286–297, 2006.
[19] I. Stojmenovic, D. Liu, and X. Jia. "A scalable quorum based location service in ad hoc and sensor networks". *Int'l J. of Communication Networks and Distributed Systems*, 1(1): 71–94, 2007.
[20] J. B. Tchakarov and N. H. Vaidya. "Efficient Content Location in Wireless Ad Hoc Networks". *Proc. IEEE MDM*, pp. 74–85, 2004.
[21] O. K. Tonguz and G. Fermri. "A Communication-Theoretic Framework for Ad Hoc Wireless Networks". Technical Report TR-043-2003, ECE, Camegie Mellon University, U.S., 2003.
[22] G. Wang, G. Cao, T. L. Porta, and W. Zhang. "Sensor Relocation in Mobile Sensor Networks". *Proc. IEEE INFOCOM*, pp. 2302–2312, 2005.
[23] G. Xing, T. Wang, Z. Xie, and W. Jia. "Rendezvous Planning in Mobility-assisted Wireless Sensor Networks". *Proc. IEEE RTSS*, pp. 311–320, 2007.

Xu Li received his Ph.D. degree from Carleton University, Canada (2008), his M.C.S. degree from the University of Ottawa, Canada (2005), and his B.Sc. degree from Jilin University, China (1998). In 2004, he held a visiting researcher position at National Research Council Canada. He is currently a postdoctoral fellow at SITE, University of Ottawa and at CNRS/INRIA, France. His current research interests are wireless ad hoc, sensor, and actuator networks, mobile robots, distributed and localized algorithms, and wireless security. He was/is involved in many scholarly activities including ACM FOWANC'09, IMAGINE'09, AdHoc-Now'08&09, LOCALGOS'09, WWASN'09, IFIP WSAN'08, IEEE MASS'07, etc.

Nicola Santoro received his Laurea degree in Scienze dell'Informazione from the University of Pisa and his Ph.D. degree in Computer Science from the University of Waterloo. He is Professor of Computer Science at Carleton University. His research activities have focused mostly on the design and analysis of algorithms. He has been involved in distributed computing from the beginning of the field, and has contributed extensively on the algorithmic aspects. His current research interest is on the computational and complexity issues arising in systems of distributed mobile entities: agents, robots, and sensors

Ivan Stojmenovic received Ph.D. degree in mathematics. He held positions in Serbia, Japan, USA, Canada, France, Mexico, Spain and UK (as Chair in Applied Computing at the University of Birmingham), and is Full Professor of the University of Ottawa, Canada. He published over 250 different papers, and edited four books on wireless, ad hoc and sensor networks and applied algorithms with Wiley/IEEE. He is currently editor of over dozen journals, 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 Networks, An International Journal). He is in the top 0.56% most cited authors in Computer Science (Citeseer 2006). One of his articles was recognized as the Fast Breaking Paper, for October 2003 (as the only one for all of computer science), by Thomson ISI Essential Science Indicators. He is recipient of the Royal Society Research Merit Award, UK. He is elected to IEEE Fellow status (Communications Society, class 2008). He chaired and/or organized > 50 workshops and conferences, and served in over 100 program committees. Among others, he was/is program co/vice-chair at IEEE PIMRC-08, IEEE AINA-07, IEEE MASS-04&07, EUC-05&08, WONS-05, MSN-05&06, ISPA-05&07, founded workshop series at IEEE MASS, IEEE ICDCS and IEEE DCOSS, and Workshop Chair at IEEE MASS-09, ACM Mobicom/Mobihoc-07 and Mobihoc-08. He presented over dozen tutorials.