When Opportunity Proceeds from Autonomy: A Tour-Based Architecture for Disconnected Mobile Sensors*

Michel Charpentier, Radim Bartoš and Swapnil Bhatia
Department of Computer Science
University of New Hampshire
Durham, NH 03824
{charpov,rbartos,sbhatia}@cs.unh.edu

Abstract

We consider the case of sparse mobile sensors deployed to implement missions in challenging environments. This paper explores a notion of tour networks *that is well suited to circumstances in which autonomous sensing agents cannot rely on standard networking abstractions and must create their own opportunities for communication and interaction. Tours are high-level building blocks that combine motion, communication and sensing and can be assembled to implement a broad class of autonomous sensing missions. They are supported by an architecture designed to deliver performance and robustness without compromising design abstraction.*

1 Background and paper overview

Mobility has the potential to dramatically broaden the range of sensing applications, but it also presents an array of unfamiliar obstacles to the networking task. Moreover, sensor mobility often coexists with environmental challenges that make it even more difficult to implement standard networking abstractions reliably and efficiently. We have started to explore strategies that forgo all-purpose networking in favor of mission-aware designs that can better use communication links opportunistically. We refer to such systems, in which mission-driven agent interaction is not expressed in terms of standard general-purpose networking abstractions, as *disconnected mobile sensors*.

Much work has been done to implement standard networking functions and protocols as efficiently and comprehensively as possible in mobile networks with the goal of attaining a general purpose networking abstraction that can support a broad range of missions. Mobility, sparseness of the network, and environmental challenges must be circumvented in order to provide users with the functionality found in more structured networks. Major progress has been made in the recent past towards this goal, for instance through elaborate routing algorithms that perform satisfactorily in spite of network disruptions, including in topologies that are possibly never fully connected [10, 4, 9].

Nevertheless, there remains circumstances in which a general purpose network is undesirable, if at all possible. Reasons that prevent the establishment of a general purpose network vary and may result from design, the sensitivity of the mission to delays, limited resources or characteristics of the environment. An illustrative example is the case of Autonomous Underwater Vehicles (AUV) deployed in vast areas and limited to comparatively short-range acoustic communication [7, 8]. Other cases include stealth missions, in which short-range communication is used to avoid enemy detection, or the use of energy efficient point-to-point directional communication devices, which make it difficult to maintain a connected network when nodes are mobile. In situations like these, agents are able to (or choose to) communicate in particular configurations of the agents only, for instance by having AUVs move underwater to within communication range of each other or surface in order to use radio links, or by having autonomous drones fly in close parallel lines so they can rely on low-power communication links difficult to intercept by their adversary.

In contrast to the work mentioned above, a key characteristic of our approach to such networks is that it does not attempt to hide the fact that the system is disconnected. In particular, this shift in perspective means that agents do not rely on a seemingly stable communication layer implemented on top of dynamic links but rather acknowledge the fact that application-level interactions can only happen in specific system configurations. Recently, computation models have emerged for such disconnected mobile agents, in which participants opportunistically form groups in order

*This research was supported in part by grant N00014-05-1-0666 from the U.S. Office of Naval Research.

978-1-4244-4439-7/09/$25.00

to interact [5, 1]. These models have been used to characterize classes of computable functions and to show that meaningful computations can be performed under fairly weak assumptions regarding which groups of agents actually get to interact as the system evolves.

These models take the extreme view that the environment controls the interaction among agents and only minimal constraints are specified regarding what this environment might do. In reality, many networks consist of autonomous agents, which have more control over their mobility than this adversarial model allows for. This controlled mobility can be taken into account to refine the models so that more precise evaluations can be performed, including performance-related results. Work on data ferrying [11] exploits controlled mobility to minimize communication delay. In this paper, we focus on a model of opportunistic computation induced by controlled mobility, which we call a *tour-network*. Our model retains the notion of group-based, joint operations and the absence of a standard network abstraction. However, it is defined in a context of controlled mobility in which agents are responsible for reaching system configurations that enable the desired interactions.

Controlled mobility does not mean that we assume environments to be benign. We still envision autonomous sensors deployed in challenging surroundings and therefore do not expect that they can always reach the desired system configurations successfully. Accordingly, the proposed tour-network model must be supported by an architecture that makes it possible for designers to rely on tour-based designs in spite of failures, delays and other environmental disturbances. This paper proposes such an architecture.

Since mobile sensor networks are being tasked with increasingly complex missions, designers will benefit from new abstractions that let them focus on the mission objectives. We propose that missions for disconnected mobile sensors be designed in terms of a tour-based abstraction, for which we provide a supporting architecture. This architecture facilitates the design of complex missions by combining motion, sensing and computation within a unified tour mechanism. We focus on a class of missions characterized by periodic sampling accompanied by an ongoing in-network computation of the mission's objectives by the sensors in a distributed way. Examples of such missions include complex event detection, intrusion perception and tracking, or monitoring of physical phenomena.

2 Tours and tour-networks

At a conceptual level, each agent in the system is charged with a sensing task (to repeatedly sample its share of the area of interest) and a computing task (to collaborate with other agents in order to perform a distributed computation based on the data acquired by sensing). This data is used, individually or cooperatively, in an ongoing mission-related computation. The systems implemented by these agents are thus reactive systems, characterized by a continuing interaction with their environment. Agents are mobile and they clearly use their motion capabilities in their sensing task. Moreover, since we assume a mission context in which it is impossible or undesirable to maintain connectivity among participants in the system, agents also need to rely on mobility to arrive at system configurations in which interaction is possible and to form groups that perform collaborative computation steps. Tradeoffs and design complexity stem from the fact that motion, which takes time and consumes resources like energy, is used for both the sensing task and the interaction with other agents. Agents must be programmed so they all visit their share of the area of interest *and* they form groups often enough to carry out a distributed computation from the acquired data. We propose to facilitate the implementation of such networks by designing them in terms of an abstraction that combines motion, sensing and computation, and by developing an architecture that supports this abstraction with efficiency and reliability.

The primary component of the proposed abstraction is a *tour*. A tour represents a period of activity of an agent, including motion, sensing and computation. It is defined by an area, a collection of meeting points and a schedule. The tour area represents the agent's share of the overall area of interest. Meeting points are those configurations of the system in which agent interaction is possible. (In its simplest form, a meeting point is a location where agents gather to communicate, hence its name.) The tour schedule defines the times at which these meetings occur. Each agent implements its tour with a trajectory that repeatedly visits meeting points according to the schedule (thus allowing interaction and collaboration with other agents) while scanning and sensing its assigned area and while maintaining mission-related constraints such as maximum speed, energy consumption or sensing-rate.

The notion of a tour provides system designers with building blocks they can assemble into solutions to particular missions, allowing them to shift low-level motion, sensing and communication operations into the tour-implementing trajectory. Given a specification, a solution is designed as a collection of tours (one per agent) assembled into a network (of tours). In particular, the algorithmic steps necessary to solve the mission are designed as computation and communication steps that agents perform during meetings.

Formally, a tour network can be represented as a graph. The vertices of this graph are the tours and the edges are the meeting points. The meaning of an edge is that the agents in charge of the tours connected by the edge will meet regularly at this meeting point to exchange information and to perform joint computations. In contrast to typical graph rep-

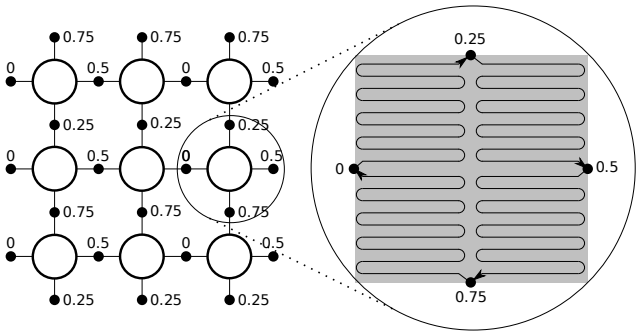

Figure 1. A rectangular grid network of tours

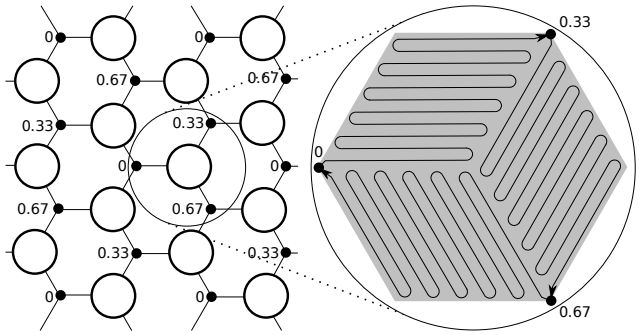

Figure 2. A hexagonal grid network of tours

resentations of networks, it is *not* that the agents share a link through which they may communicate at any time. We consider graphs with both multi-edges (the same agents may meet in more than one location) and hyper-edges (meetings may involve more than two agents). Moreover, these graphs are enriched with node and edge annotations that represent the geometry of the area covered by a tour and the locations and schedules of meeting points.

As an example, fig. 1 shows a section of a fully labeled tour network in which tours are assembled into a rectangular grid. Each tour area is a rectangle to be monitored by an agent and is associated with four meeting points, located on the four borders of this area. Meeting points are labeled with relative times within the tour cycle, chosen from the set $\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. This results in a schedule that maximizes the shortest time between two meetings (to a quarter of the cycle). The idea is to give agents as much time as possible between any two consecutive meetings to scan the region during trips from one meeting point to the next. The right-hand side of the figure shows a possible implementing trajectory for one of the tours. Such a trajectory visits the meeting points at the specified times while scanning exactly a quarter of the area when traveling from a meeting point to the next. Fig. 2 shows a different network, in which tours are assembled into a hexagonal grid. In this network, each agent interacts with six neighbors and must attend three meetings per period at relative times $\{0, \frac{1}{3}, \frac{2}{3}\}$. In contrast to the pairwise meetings of the previous example, each meeting in this network involves three agents and represents a tripartite computational step.

3 Tour-supporting architecture

In tour-based designs, agents must attend meetings according to an agreed-upon schedule in order to communicate and collaborate with other agents. Each agent is responsible for building a trajectory that allows it to visit meeting points at the scheduled times while scanning its area for data acquisition. Relying on tour networks thus requires agents to interact in a loosely choreographed way so the necessary encounters are likely to occur. This makes the realization of a tour-based design a nontrivial task, especially in challenging environments: Agents may miss meetings because of failures or delays, and this can have adverse effects on the agents' mission. If tour networks are expected to facilitate mission design, they must be supported by an architecture that implements them reliably and efficiently in spite of environmental challenges.

We propose an architecture capable of realizing tours and tour networks in spite of challenges such as agent loss (transient or permanent) or delay. This architecture allows agents to monitor their surroundings and to adapt both the tour network itself (in a distributed, cooperative way) and their individual tour implementing trajectories based on changes in the environment. This capacity of the architecture to reconfigure tours as the mission progresses results in an implementation of tour-based designs that is adaptive, efficient and reliable. The architecture is built on top of standard system control interfaces, which are used for the low-level implementation of the motion, sensing and communication necessary in each tour.

The proposed layered architecture is outlined in figs. 3 and 4. It starts with a *mission planner*, which is an offline component in charge of designing a tour-based solution to a particular mission, given resource and environment constraints. This design may or may not be computer aided, making use for instance of optimization techniques, but it is essentially an offline activity. It results in a tour network that represents the stable state of the system—which parts of the area are monitored by which agents, where the meeting points are located and when meetings are scheduled. These initial tour specifications are uploaded into the agents before they are deployed, after which the tour control layers handle the online activities required to evolve and implement this statically designed tour network.

The *Tour network transformation layer* is handed over the tour network designed by the mission planner as well as

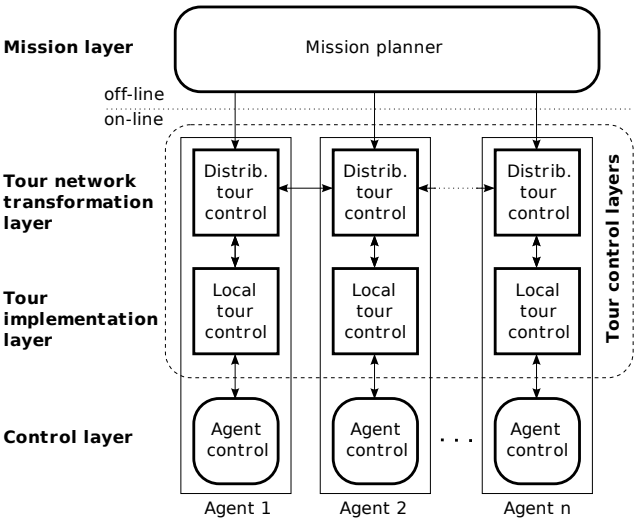

Figure 3. Tour-supporting architecture

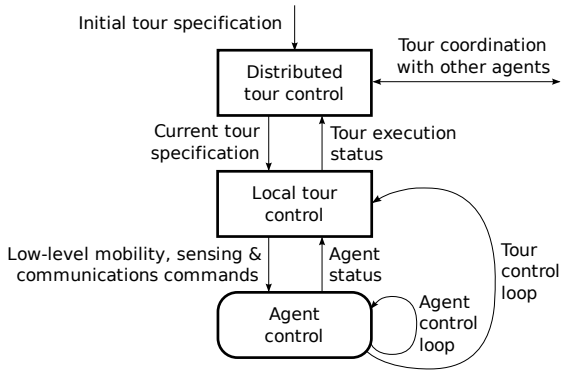

Figure 4. Inter-layer information flow

a set of desirable mission properties enjoyed by the network in its stable state, like resilience or adaptiveness or performance. These properties involve quantities such as the size (number of participants) of meetings, the duration between two consecutive meetings on a given trajectory, or the lowest sensing rate acceptable for the mission. The role of the tour network transformation layer is to adjust the network to take into account changes in the environment, including the loss of agents, while striving to maintain mission related properties of tour networks, as specified by mission designers. Some desired properties can be guaranteed to endure, for instance that no agent is attending more than N meetings per cycle, or that no meeting involves more than M agents. Other properties, however, are only used as guidelines and are handled in a best-effort fashion. For example, it is not possible to guarantee a minimum sensing rate for each point in the area if there is no known upper bound on the number of agent failures.

The tour network transformation layer relies on a collection of distributed protocols used by agents to autonomously and cooperatively reconfigure a tour network in reaction to changes in their environment, for instance by adding or removing meeting points, by changing the location and/or time of a meeting point, or by repartitioning an area among a group of agents. Most (but not all) of these network transformations are coordinated, requiring that agents interact to put them into effect, but they are essentially local (no central coordination) and have limited impact on the rest of the network. Some transformations are triggered by single-agent detection of a change in the environment, after which an agreed upon modification to the network is performed. As an example of such an uncoordinated transformation, consider an agent that interacts with four other agents using four meeting points for pairwise meetings. If this agent is lost and hence stops attending its meetings, all four agents can detect this loss individually when they realize that the fifth agent has not attended a meeting and then implement a transformation that the four of them had agreed upon beforehand. Sect. 4 illustrates possible tour network transformations and the mission-related properties they result in.

The next layer in our architecture is the *Tour implementation layer*, in charge of implementing and monitoring tours at the level of one agent. The task of this layer is to implement, by building a suitable trajectory, the current tour of an agent, as handed down by the tour network transformation layer following network evolution. In the upward direction, the tour implementation layer is used to monitor tours and report relevant information to the tour network transformation layer. For instance, agent loss can be detected by missed meetings and reported above so the network can be reconfigured accordingly. Various information regarding the current trajectory and sensing data can be forwarded to the tour network transformation layer to be used as guidelines when network transformations are being considered and decisions that affect the tour implementation layer need to be made, such as the location and schedule of meeting points.

Finally, the tour implementation layer relies on a *control layer* to physically drive the corresponding agent and gather information from its environment. This layer is responsible for localization, sensing, the implementation of low-level motion and communication commands and elementary distributed computing operations, like leader election or atomic commitment. It handles variations in the environment, both in terms of motion primitives (using standard control system) and communication primitives (using standard networking techniques), and informs the tour implementation layer of conditions that this layer cannot handle directly.

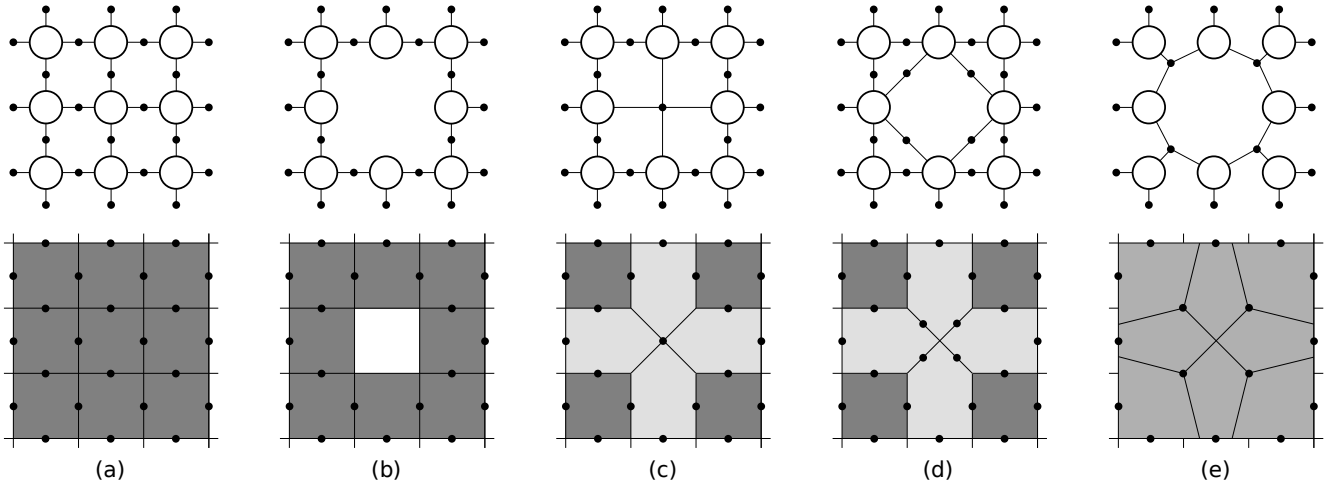

Figure 5. An example of topological transformations after a node failure and corresponding geometric transformations

4 Tour-network transformations

At a fundamental level, the role of the tour network transformation layer in the architecture described above is to maintain the desirable topological (graph-theoretic) and geometrical properties of a network of tours in spite of changes in the environment. For instance, when agents meet, they can decide to move the location and schedule of the meeting point for the next cycle if they jointly decide that the new time and location will result in better trajectories. In contrast to such transformations involving only meeting times and tour area geometry, more elaborate transformations can modify the tour network topology by adding, removing or merging meeting points, or by modifying the membership of one or more meetings.

Such topological and geometrical transformations may be interdependent: A topological transformation may trigger geometrical adjustments and vice-versa. For instance, a topological step that removes a meeting point from a network can be followed by geometrical steps that adjust tour areas and trajectories now that there is no need for agents to visit the meeting point that was eliminated. Conversely, changes in agent tour areas and in the locations of their meeting points may result in a suboptimal topology in which agents geometrically close do not interact directly, thus triggering a topological step to create a new meeting point.

To illustrate the changes that take place in the tour transformation layer, the external events that trigger them and the mission-related metrics that guide them, consider the following scenario. A regular grid of tours is deployed for a sensing mission during which, at one point, an agent is permanently lost. Fig. 5 represents a possible series of steps that agents can take to adapt the network after such a failure. The top row outlines the topological transformations undertaken by the network; the bottom row describes the evolution of tour areas, meeting point locations and average scanning rates. Black dots represent meeting points and large white circles represent tours, as before (the meeting times associated with each meeting point are omitted for clarity). Each tour area is drawn in the bottom part of the picture, using gray levels that represent scanning rates (the darker the gray, the higher the rate).

In (a), the network is in its stable state: each agent monitors a unit square using a trajectory that visits four meeting points per cycle (up, down, left and right) and the entire area is sampled with high rate. In (b), the agent in charge of the central tour fails. The corresponding area is then not monitored at all (null sensing rate). In (c), the four neighbors of the missing agent detect (at different times) that the agent is missing, based on the fact that it is not attending its scheduled meetings. They unilaterally switch to an agreed upon new pattern in which the lost agent is replaced with a new meeting point attended by all of its four neighbors. At the same time, the tour area of the missing agent is split into four triangles used by all four neighbors to extend their own tours. Since the tour areas of these agents get larger, they are rendered using a lighter gray to represent the decrease in scanning rate. So far, the agents of the four corner tours have not been involved. They continue to scan their areas with the same rate and they attend the same meetings as before. In stage (d), the four agents already involved in the operation decide to split the new meeting point into four pairwise meetings, presumably because a meeting of four agents was deemed undesirable by the mission planner. Finally, in (e), the agents responsible for the four corner tours are notified of the failure as asked to adjust to the new network. Four sets of three pairwise meetings are merged

into four three-agent meetings and tour area boundaries and meeting point locations are recalculated to balance the sensing workload. This results in a new network fragment that is uniformly scanned with a sensing rate not quite as high as what it is in the rest of the network, since an area originally monitored by nine agents is now handled by eight. Moreover, each agent now attends three meetings, presumably because the mission planner regarded this as more desirable than having some agents attend five meetings.

For each stage of the scenario above, the tour transformation layer implements a simple distributed protocol that guarantees that the transformation is successfully implemented. The first operation (replacement of a missing agent with a meeting point) uses no communication among the participants, but requires an agreed-upon location and time for the new meeting. The second operation is implemented in a centralized way, since all four agents attend a meeting in which they all participate. The last operation, however, requires a true distributed protocol because the three agents involved in each merge form a clique of pairwise meetings but there is no single meeting of all three agents.

In terms of network properties maintained, it can be shown easily that this way of handling agent loss will always result in a connected graph of tours, even in case of multiple failures (assuming each transformation is atomic and can be implemented entirely between failures). These operations also have a clear impact on the diameter of the graph and on the number of meetings per agents and of agents per meeting, and they can be used to increase or decrease these metrics in accordance to the goals of a mission planner. Furthermore, the last stage can easily be implemented in such a way that all eight areas have equal size in the end, thus guaranteeing, as a geometrical property, that the operation maximizes the lowest scanning rate among all agents.

5 Current and future work

This paper proposes an architecture to support tour-based mission design abstraction for mobile sensors operating in a sparse and challenging environment where establishment of traditional networking infrastructure is either impractical or undesirable. As the next step in the evolution of the architecture, we are currently designing tour network transformation operations that maintain desirable topological and geometrical properties of the tour network. We are studying optimization of and trade-offs in tour-based mission designs [6] and evaluating metrics to guide the definition of tour network transformations. We are also devising protocols, to be run within the proposed architecture, so that agents can implement these transformations efficiently and reliably. As a part of our ongoing partnership with the Autonomous Undersea Systems Institute (AUSI) [2], we will define the interfaces between the layers and build a reference implementation together with supporting simulation and emulation tools. We intend to port the reference architecture to the AUSI Solar-powered AUV and test it in the context of real-world missions. On the theoretical side, we are studying the fundamental properties of formal models of disconnected mobile sensors [3]. We are exploring how properties of the joint computation executed in meetings can inform the design of a tour network and its evolution through transformations.

References

[1] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[2] R. Bartoš, S. G. Chappell, R. J. Komerska, M. M. Haag, S. Mupparapu, E. Agu, and I. Katz. Development of routing protocols for the solar-powered autonomous underwater vehicle (SAUV) platform. *Wireless Communications and Mobile Computing*, 8(8):1075–1088, 2008.

[3] S. Bhatia and R. Bartoš. Self-similar functions and population protocols: a characterization and a comparison. In *Proc. of the 10th International Conference on Distributed Computing and Networking (ICDCN 2009)*, pages 263–274, 2009.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, Apr. 2006.

[5] K. M. Chandy and M. Charpentier. Self-similar algorithms for dynamic distributed systems. In *27th International Conference on Distributed Computing Systems (ICDCS'2007)*, June 2007.

[6] M. Charpentier, R. Bartoš, and S. Bhatia. A mechanism to structure mission-aware interaction in mobile sensor networks. In *Proc. of the 10th International Conference on Distributed Computing and Networking (ICDCN 2009)*, pages 425–436, 2009.

[7] J. Partan, J. Kurose, and B. N. Levine. A Survey of Practical Issues in Underwater Networks. In *WUWNet '06: Proceedings of the 1st ACM International Workshop on Underwater Networks*, pages 17–24, Los Angeles, CA, 2006.

[8] D. Pompili and I. F. Akyildiz. Overview of networking protocols for underwater communications. *IEEE Communications Magazine*, 47(1):97–103, Jan. 2009.

[9] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. *IEEE/ACM Trans. on Networking*, 16(1), Feb. 2008.

[10] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys and Tutorials*, Jan. 2006.

[11] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay tolerant network. In *IEEE INFOCOM*, Apr. 2005.