Force-Based Navigation in Wireless Sensornets

Igor Talzi, Massimo Monti, Thomas Meyer and Christian Tschudin
Mathematics and Computer Science Department, University of Basel
CH-4056 Basel, Switzerland
Email: {igor.talzi,massimo.monti,th.meyer,christian.tschudin}@unibas.ch

***Abstract*—Certain scenario-oriented network tasks require a non-traditional approach to routing. This happens when routing should become aware of the node's state defined beyond the network layer (e.g., sensed value, current physical location, etc.). In this paper we present an approach that helps software elements to navigate through the network based on the metaphors of fields and forces known from physics. The elements encapsulate logic to make a decision on the next move in the dynamic environment that is formed by overlapping, interacting fields and represent a physical value (e.g., fire) or a role (e.g., sink node). As a result, the force is induced on a node which pushes the element in a certain direction, towards or even *away* from the field source. We illustrate the feasibility of our solution using an emergency evacuation scenario for Wireless Sensor Networks (WSN). We show that our system is simple yet provides more flexibility and functionality than existing counterparts.**

***Index Terms*—Artificial Physics, Gradient Routing, Mobile Code, Wireless Sensor Networks**

I. Introduction

Transmission of data or code in computer networks is usually associated with an underlying routing layer. The objective of classical end-to-end routing algorithms (e.g., OSPF [1]) is to compute the shortest path to a given destination using various criteria, such as link quality, traffic conditions, etc. This goal rests on two assumptions: First, each node in the network has a global unique address, and second, a certain destination must be specified at the time of transmission.

In some types of networks, especially in Wireless Sensor Networks (WSN), these basic assumptions do not necessarily hold. Often, we want to address nodes by their roles and not by global addresses. For example, values measured and gathered by a sensor field should be sent to any node that fulfills the role of a base station. In such a scenario, sensor nodes only need to know the direction, which the collected measurements must be forwarded to. Another common task in WSN is to deploy jobs to nodes in a certain region. In all these cases a region-oriented, fuzzy routing would be preferable to traditional point-to-point schemes.

Traditionally, the question of region-oriented navigation is addressed using a middleware layer, which provides necessary abstraction primitives. Underneath, it still uses complex routing (multi-hop, multi-sink) and addressing schemes. Many such examples can be found in the WSN domain, for data collection [2], [3] and code dissemination tasks [4].

Several network-layer approaches for WSN have been proposed which are based on landmark forwarding. Position-based routing (see surveys in [5], [6]) lets an intermediate node forward packets based on its own, its neighbors' and the destination's geographic position, for example using the greedy GEographic DIstance Routing (GEDIR) method [7]. If the exact geographical position is unknown, landmark routing [8] or gradient-based routing schemes [9], [10] are able to derive a virtual position from the node's distance to landmark points, the positions of which are known. Similar to our approach, [11] proposes a gradient routing scheme inspired by physical fields.

While middleware requires big message complexity and lacks flexibility, the low-level routing schemes do not provide enough abstraction, which is needed to easily describe scenario-oriented network-wide tasks. Let us consider a "fire escape" scenario shown in Fig. 1 where an object (running man) has to evacuate itself through one of the emergency exits (*RE* or *LE*). Instead of heading to the exit following the *shortest* path, the object should choose the *safest* path (i.e., it should prefer exit *RE* to *LE*) staying away from fire hotbeds. In case of multiple equally safe exists the closest one should be selected.

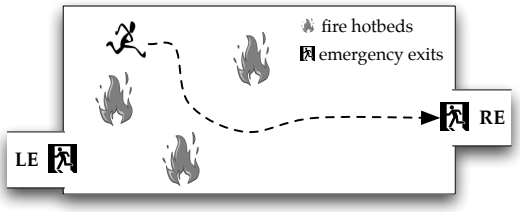


Fig. 1: We choose the safest way to the exit by staying away from the fire.

This type of network task is not easily programmable using any existing solutions. In this paper we present a framework that adheres to a *physical metaphor*: Scenario-specific sources (fire hotbeds, exits) announce their position to the surrounding area by propagating *fields*. The fields of multiple sources overlap and can interact. In turn, in-network *mobile elements* interact with different fields through *forces* in analogy to physical particles that are attracted to or repulsed by a field source. To this end, mobile elements include logic to describe the interaction.

The novelty of our approach is fourfold: 1) We support multiple interacting fields, which represent different roles in

the network. 2) Other than in landmark or gradient routing, mobile elements decide themselves how to interact with each individual field. This allows different types of mobile elements to move to different locations, or even to change their behavior *en route*. 3) By introducing repulsive forces we give mobile elements the possibility to move away from a certain field source, for example, to avoid a dangerous area of the network (e.g., fire, high network traffic). Such a behavior is not possible with traditional routing. 4) In analogy to physical forces (e.g., Coloumb or gravitational force), the influence of a field source is stronger the closer a mobile element is. This often allows resolving conflicts when similar forces compete.

This paper is structured as follows: Section II explains the system architecture and introduces basic concepts such as mobile elements, fields, and forces. In Section III, we illustrate the applicability of our approach with the fire escape scenario. The paper rounds up with a discussion in Section IV where we assess our system and point out future directions. We conclude the work in Section V.

II. System Architecture

In this section we describe the conceptual building blocks of our framework and how they interact. Our system incorporates three basic primitives: mobile elements, fields and forces. The relation between these three is shown in Fig. 2.

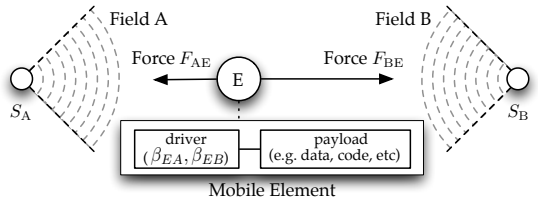


Fig. 2: Fields (A, B) generated by sources (S_A, S_B) interact with a mobile element (E); the induced forces (F_{AE}, F_{BE}) push the mobile element.

a) Mobile Elements: These represent some mobile tasks (mobile agent, data delivery, etc.; E). A mobile element has two main structural blocks: driver and payload. The driver encapsulates the logic needed to make decisions on how the payload should be navigated through the network. The payload is detachable from the driver, meaning that drivers can be dynamically exchanged to provide different navigation rules. The main task of the driver is to demonstrate the level of attraction with respect to all (or some) of the existing field sources. To this end, the driver contains a coupling coefficient for each field (β_{EA}, β_{EB}) it wants to relate to.[1] The execution environment makes the decision on the next hop by considering this information. Those fields are ignored to which the driver does not show any attractiveness (i.e., $\beta = 0$).

[1]In analogy to physics the coupling coefficient is comparable to the charge of a particle or its mass when interacting with an electrostatic or a gravitational field, respectively.

b) Fields: A field allows a node to announce its role or state globally or in an extended neighborhood. Every node can be a source (S_A, S_B) of a scalar field (A, B). Sources initiate field propagation, and each node can host a source of multiple fields. In each point (node) of the network a scalar field is characterized by the magnitude, which has its highest value at the source node and decays with each hop until it reaches zero.

c) Forces: In analogy to physical forces we define a force in a computer network as a preference of a mobile element (particle) to move towards or away from a field source. Like in physics, the strength of a force – induced by a field – is proportional to the magnitude of the field at that point and to the coupling coefficient (particle's property) expressed by the corresponding mobile element. This means that unlike in gradient routing, a mobile element should prefer the closer and the stronger field (as discussed in [11]) rather than performing a gradient ascent algorithm [9]. This behavior helps to resolve the situation where two competing forces are dragging a mobile element in the opposite directions, e.g., towards the opposite corners of the network. In this case a mobile element should move to the closer one. In our system, on each node covered by the field, a force (F_{AE}, F_{BE}) is induced.[2] The force vectors induced by different fields are independent. However, as we show in Section II-B, fields can interact. This interaction has an indirect impact on the induced forces.

A. Field Propagation

Every node in our system can be assigned one or more field sources; and multiple sources can generate the same type of field. Field propagation starts at the source with a broadcast transmission of a *field-announce*(k, m_k, f, α) message, where the field identifier k is a label (name) that uniquely identifies the field in the network, m_k is the field magnitude, $f(m_k, \alpha)$ is a fading function that decreases the magnitude at each hop. Fig. 3 shows commonly used linear and exponential fading functions; the constant parameter α defines how fast the magnitude fades from hop to hop.

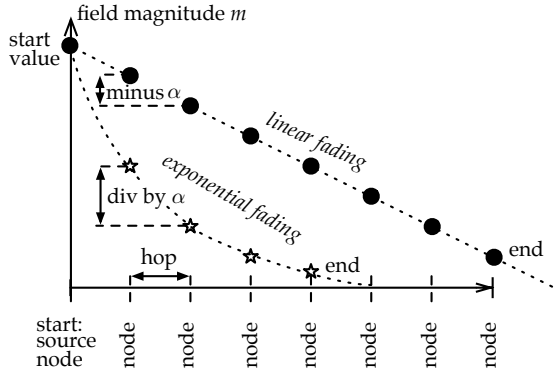


Fig. 3: Hop-to-hop (linear or exponential) fading of a field

[2]In Fig. 2, forces F_{AE} and F_{BE} may have the opposite direction depending on the coupling coefficients.

Each node i maintains a local state which includes a magnitude vector that stores the magnitudes of all K fields intersecting on this node, $\vec{m}^{(i)} = (m_1^{(i)}\ m_2^{(i)}\ \dots\ m_K^{(i)})$, and a neighbor magnitude vector that remembers the magnitudes $\vec{m}^{(ij)} = (m_1^{(ij)}\ m_2^{(ij)}\ \dots\ m_K^{(ij)})$, announced by all neighbors $j \in \mathcal{N}_i$.

When node i receives a *field-announce* message it updates its local state. Let us assume this message has arrived from neighbor j, telling that the magnitude of field k has been set to $m_k^{(j)}$. Node i first saves this value in the corresponding element of the local neighbor magnitude vector $m_k^{(ij)} := m_k^{(j)}$. As a second step the fading function is applied resulting in a proposed local magnitude $m_{k,\text{prop}} := f(m_k^{(j)}, \alpha)$. Node i then updates its local magnitude vector ($m_k^{(i)} := m_{k,\text{prop}}$), but only if the proposed magnitude is larger than the current value. In this case the node also composes another *field-announce* message, *field-announce*$(k, m_k^{(i)}, f, \alpha)$, which is broadcast to all neighbors.

Note that neighbor j, which originally sent the announcement to i, will also receive this new announcement from i and store the new magnitude in its neighbor magnitude vector. Because the value has been reduced in i by applying the fading function, the originator j does not overwrite its own magnitude and does not broadcast the announcement message further. As a result, the field fades away from the source and no loops are created. In fact, the presented algorithm is similar to the update mechanism used in Distance Vector (DV) routing protocols [12] and the viral propagation, a part of many network dissemination protocols, too [13].

Currently, our design does not support a periodic field update mechanism. For the case where we want to change some field parameters we may propagate a new field with a different name and an updated parameter set.

B. Field Interaction

We also allow fields with different names to interfere. In Section III, we will demonstrate that in some scenarios multiple competitive forces can cause a trap which is similar to a routing loop. To resolve such situations we use a trick, which we call *field puncturing*.

As shown in Fig. 4, the main idea of this approach is that during propagation a newly covering field (field 1; generated by source 1) is weakened in each node by the magnitudes of the fields that already reside in that node (field 2; generated by source 2). When parameterized correctly, the covering field drops to zero when overlapping with those fields. By doing so we create a puncture in the covering field. Consequently, mobile elements attracted by the covering field will avoid going through the punctured area while navigating towards the source (shown as vector field in Fig. 4 and by a sample trajectory).

The parameterization may be a non-trivial task in case of multiple overlapping fields and the need to create mutually exclusive punctures. In this work, we further assume that fields are propagated sequentially in a well-defined order to avoid miscalculation of the induced forces.

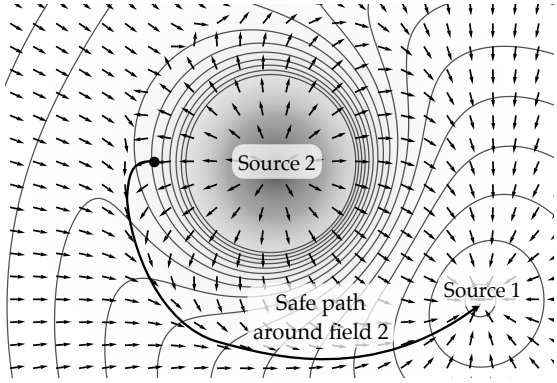


Fig. 4: Interaction of two fields. Equi-magnitude lines: magnitude of the scalar field 1, influenced by field 2; gray scale gradient: magnitude of the scalar field 2; vectors: gradient of field 1, i.e., vector field pointing towards higher magnitudes; arrow: sample trajectory of a mobile element attracted by source 1.

C. Navigation using Forces Based on Fields

Every node in our network can be addressed by its magnitude vector, i.e., by the "distance" to the different field sources. The problem is that, unlike in many position-based routing schemes, this "address" is not guaranteed to be unique.[3] As a result, our navigation system cannot deliver to a specific node that is not a field source. Conversely, mobile elements in our system can be routed as follows: 1) Move towards a source; 2) move away from a source; and 3) combinations thereof.

In our design each mobile element defines itself how it wants to interact with various fields. By coupling to a certain field the driver of a mobile element defines whether it is attracted or repelled by the field source and to what extent. The driver can express itself with respect to multiple fields at a time. The resulting force that pushes the mobile element towards the next hop is determined by a simple linear superposition of the individual forces in magnitude space. We have developed an algorithm that allows to find a next hop in such networks based on the Compass position-based routing method [7].

Our Next Hop Discovery (NHD) algorithm is triggered when receiving a *mobile-element*$(\vec{\beta}, payload)$, where $\vec{\beta} = (\beta_1\ \cdots\ \beta_K)$ is the coupling coefficient vector, which determines how the mobile element interacts with all the fields. The NHD algorithm simulates an artificial physics in which the mobile element is affected by forces to find the next hop. Two steps of the algorithm can be distinguished: 1) Compute the total force vector in magnitude space; and 2) find the neighbor that is closest to the direction of the force vector. The algorithm makes use of local information only. At node

[3]For example, if there is only one field, all nodes with the same hop-distance to its source have the same "address".

i this includes a) the coefficient vector of the mobile element $\vec{\beta}$; b) the local magnitude vector $\vec{m}^{(i)}$ received during the field propagation phase; and c) the neighbor magnitude vectors $\vec{m}^{(ij)}$ storing the field magnitudes of all neighbors $j \in \mathcal{N}_i$.

Below we present the NHD algorithm in a pseudo-code form:

$\vec{F}^{(i)} := \mathrm{diag}(\vec{\beta})\vec{m}^{(i)}$ {Compute the force vector by multiplying the coupling coefficient vector with the local field magnitude vector component-wise.}

for all neighbors $j \in \mathcal{N}_i$ **do**

$\vec{d}^{(ij)} := \vec{m}^{(ij)} - \vec{m}^{(i)}$ {Compute the displacement vector, i.e., the magnitude differences towards neighbor j.}

$P^{(ij)} := ||\vec{P}^{(ij)}|| = \vec{F}^{(i)} \frac{\vec{d}^{(ij)}}{||\vec{d}^{(ij)}||}$ {Project the force vector to the displacement vector.}

end for

return $\arg\min_j P^{(ij)}$ {Return the neighbor that is closest to the direction of the force vector.}

As can be seen our algorithm computes the force vector that drags a mobile element to a certain direction and then tries to minimize the angle to the neighbor's position. Unlike in compass-based routing [5], [7] we do not use the distance to landmarks directly, but let the mobile element interpret the fields' meaning, resulting in a force that gives its direction. The problem with this method is that unlike in real physics or position-based routing, nodes only have limited information about their position in the two- or three-dimensional continuous physical space. Nodes have to derive their location directly from the field magnitudes. As shown in Fig. 5 below, we therefore use the magnitude vector as a position, placing ourselves into a virtual magnitude space.

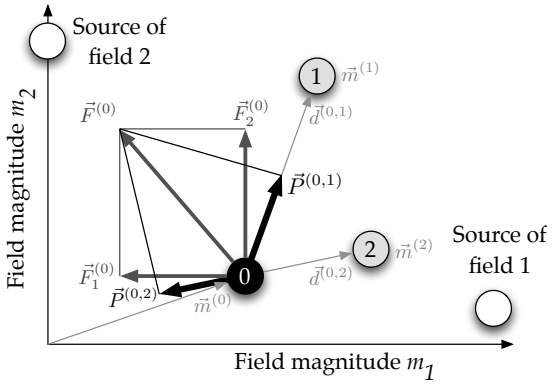


Fig. 5: Forces and projection to neighbor displacement vectors in the virtual magnitude space.

This figure shows a hypothetical node 0 in magnitude space surrounded by two neighbors, 1 and 2. There are two field sources, thus the magnitude space is two-dimensional. Computation of the force vector $\vec{F}^{(0)}$ as well as the projections to the neighbor displacement vectors $\vec{d}^{(0j)}$ are performed using standard vector calculus in the magnitude space. This trick allows mobile elements to have a global orientation with respect to field sources (landmarks) without the need for global addresses or explicit knowledge about the distance to the landmarks.

III. Fire Escape Scenario

The example application we present in this section is a fire escape scenario where a mobile element tries to evacuate itself through a safe exit. Instead of approaching an exit on the *shortest* path, the mobile element chooses the *safest* path by staying away from fire hotbeds. We show the experimental results obtained from OMNeT++ simulations.

A. Experimental Setting

Both fire hotbeds and exits are announced by the field sources *FIRE* and *EXIT*, respectively. Mobile elements populating the network are equipped with a driver that imprints the "instinct" to move away from fire hotbeds and towards a safe exit. This is achieved with a coupling coefficient vector of $\vec{\beta} = (\beta_{\text{FIRE}}, \beta_{\text{EXIT}}) = (-1, +1)$. We assume that field sources and mobile elements are pre-deployed in the network. Mobile elements can migrate while physical network nodes and field sources remain static. The testbed area covers an area of $260\,\mathrm{m} \times 260\,\mathrm{m}$ where 200 nodes are randomly placed mimicking a WSN deployment site for fire monitoring as depicted in Fig. 6. Two nodes closer than $25\,\mathrm{m}$ from each other are within each other's transmission range[4] indicated by interconnecting lines. Those lines represent bidirectional links, which are assumed to be close to ideal: Message exchange does not suffer from packet loss, nor from bit errors. A cumulative propagation and transmission delay is set to $20\,\mathrm{ms}$.

B. Escape from Fire...

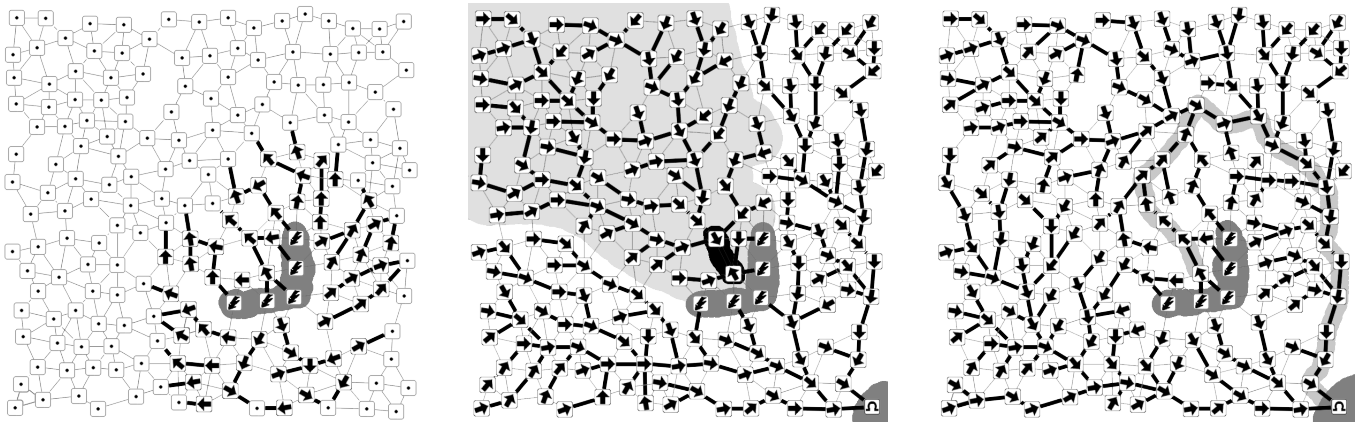
Fig. 6a shows the simulation scenario after the fire breaks out in the center of the network and is detected by the nodes marked with a flash symbol. These nodes generate a *FIRE*-field which decays exponentially (magnitude is divided by $\alpha = 4$ at each hop) with an initial value of 200.

Since mobile elements are configured to be repulsed by a *FIRE* source, they escape the fire, moving toward the neighbor with the lowest fire field magnitude until they reach a node where the magnitude is zero (nodes with a dot symbol). Fig. 6a shows the paths of mobile elements in the network. In particular, the arrow of each intermediate node points towards the selected neighbor, and the link to this preferred neighbor is represented by a bold line.

C. ...and Approach an Exit

As soon as the node at the bottom-right corner announces an exit, the simulation scenario dynamically reconfigures itself.

[4]The maximum communication range of TelosB motes, a classic WSN prototype platform, is around 75-100 meters if there is no obstacle in between. So, 25 meters, the breaking distance between nodes in our experiment, seems reasonable.



(a) Outbreak of fire in the center region detected by the nodes marked with flash symbols. Mobile elements escape from the fire by moving along the bold links. Mobile elements are not subject to forces in nodes with a dot symbol – they stay there.

(b) In addition to the fire an exit is announced at the bottom-right corner ("gate" symbol). The shaded area in the top-left corner marks a basin of attraction in which the mobile elements are trapped to the black fixed point.

(c) The trap can be avoided by letting the exit field interact (weaken) with the fire field. A sample escape path from behind the fire is highlighted.

Fig. 6: OMNeT++ simulation of self-evacuating mobile elements. Thick links mark their escape path, which is determined by repulsion from fire and attraction to an exit.

The evacuation point (exit) is represented by a "gate" symbol in Fig. 6b. This node generates an *EXIT*-field that linearly decays (decremented by $\alpha = 1$ one in each hop) with an initial value of 100. Paths (arrows and bold lines) in Fig. 6b indicate that the elements strive towards the exit, still staying away from the fire as expected.

Note that the navigation of mobile elements is defined by a purely local decision making process: each node has access to information about the field magnitudes on all neighbor nodes and its own only. This locality in the decision-making mechanism may lead to *traps*: Mobile elements are not able to reach the exit from the network region in the top-left corner of Fig. 6b. This shaded region represents a basin of attraction. A mobile element within this basin strives towards the local attractor (black background), which is a set of two neighbor nodes between which the element is sent along the back and forth link in a routing loop. This phenomenon is due to the presence of multiple competitive forces, resulting in locations where the decision making process becomes ambiguous.

D. Avoid Traps by Allowing Fields to Interact

For some topologies, traps do not occur at all or can be avoided by changing field parameters (initial field magnitude and field fading function). What can also help is to provide additional exits at different locations in the network. However, to generally guarantee the prevention of local traps, interactions between fields are required. The conflicting situation depicted in Fig. 6b is resolved with this advanced approach. The result is shown in Fig. 6c.

In fact, what happens is that we apply an advanced fading function for the exit field: along with the linear fading function the local *FIRE* magnitude is also subtracted from the *EXIT* magnitude. This leads to the new scenario where the *EXIT* field is punctured in magnitude space by the *FIRE* field as discussed in Section II-B. The *EXIT* field fades to zero when directly crossing a fire wall but is still able to propagate around the fire without hindrance. Therefore, any mobile element behind the fire is not guided through the fire, but rather safely guided around the hot beds. The lightly highlighted path in Fig. 6c represents the trajectory taken by one of the mobile elements (c.f. Fig. 4). The starting position of the mobile element was arbitrarily chosen closely behind the *FIRE* source. Despite its desperate location, the mobile element is able to safely reach the exit circumnavigating the unsafe area. Moreover, the local attractor in Fig. 6b dissolves in Fig. 6c. There is only one basin of attraction left, the one with the evacuation point (exit) as attractor.

E. Characteristics

While operating with gradients our method does not require globally unique addresses. The amount of memory needed to store local positioning information is drastically reduced compared to traditional end-to-end routing where we would have to store routing tables. This is an important feature for resource-constrained devices like WSN. Apart from the fields there is no need for other information or name resolution.

The number of messages sent to deploy a field is in the order of number of links in the network, since each node has to broadcast a *field-announce* message. If we allow fields to interact, the message complexity rises because nodes on the most direct propagation path will probably later be updated with higher magnitudes from detours. Currently, our fields are static, meaning that once deployed they do not adapt to changes in the network topology. We also require that fields are propagated sequentially – one after another – to avoid miscalculation of induced forces.

IV. Discussion

The presented approach shows some similarities with the existing work on emergency navigation. [9], [14] use gradient-based routing. TORA [9], for example, builds a gradient field towards the exit. Nodes detect emergency incidents and modify this field similar to our field puncturing method. Packets then follow the steepest ascent path through the field valleys towards the exit. Our system operates similarly when there is only one field type (e.g., the *EXIT*-field). Our main contribution is to extend and generalize gradient routing in two aspects. First, we generalize the scheme by allowing multiple overlapping, (non-)interacting fields. Second, by making routing decisions with respect to the induced forces instead of the fields directly, we treat the fields as a landmark information offered to mobile elements but let the mobile element decide how to react. While one type of mobile element may try to avoid dangerous zones and approach an exit, other mobile elements may want to move towards the danger in order to chart or defeat them. Another difference from the existing solutions is that we can regulate the coverage of the landmark information, thus providing local or global navigation metrics.

With our system, although not explicitly demonstrated, we also made a leap towards a simple network re-tasking. All parts of our system (field propagation, field interaction, mobile elements, etc.) were implemented in Fraglets [15], a prefix language developed for active networking. The advantage of using Fraglets or another interpreted language is that apart from the compact interpreter, no code has to be deployed beforehand. In such a setting, we can combine role-assignment and network functions in a flexible way. For example, a mobile element (active code) may navigate through the network and interact with the environment: If a mobile element reaches a node with a certain sensor value range (e.g., temperature too high: fire detected), it may activate another part of its program that assigns the node the role of a field source, which influences other mobile elements in turn. There is no known framework for WSN, which would allow that.

In the future, in order to be able to model more dynamic and adaptive scenarios (in our example this might mean new fire hotbeds to occur or exits to get blocked while the elements are moving), we are going to extend our framework with continuous field updates. This requires to introduce version numbers to the *field-announce* messages and to decay a field after a certain time. This also would provide support for mobile sources. But before that we have to estimate the rise in the power consumption due to the increased message complexity. Other navigation strategies such as move perpendicular to a field source are also pending. The last would be useful to border an incident area.

In addition to the artificial source-induced fields that we are using in this paper, other types of scalar fields including natural ones (e.g., temperature, humidity, brightness, etc.) can be used. The navigation principle remains the same while the field propagation must change. In this case the field magnitude becomes a function of the sensor reading that is exchanged within the local neighborhood only.

V. Conclusions

In this paper we presented an approach and a framework that together implement a network navigation system based on multiple interacting gradient fields. Mobile software elements communicate with this new media to make a decision on where to move next. Our method uses the basic element of classic physics, forces, to represent this communication. Conceptually, our design extends the existing works on gradient routing and emergency navigation but allows for more flexible and powerful routing mechanism. This makes implementation of certain network tasks, which include high level of mobility an easy job. The modeled fire escape example for WSN has shown that complex navigation tasks can be easily programmed using the proposed tools.

References

[1] J. Moy, "RFC 2328 "OSPF Version 2"," The Internet Society, April 1998.

[2] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," in *ACM Transactions on Database Systems 30, 1*, 2005.

[3] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," in *In Proc. of 1st Symp. on Networked Systems Design and Implementation (NSDI)*, 2004.

[4] C.-L. Fok, G.-C. Roman, and C. Lu, "Rapid development and flexible deployment of adaptive wireless sensor network applications," in *In Proc. of the 25th Int. Conf. on Distributed Computing Systems (ICDCS)*, 2005.

[5] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, 2001.

[6] I. Stojmenovic, "Position-based routing in ad hoc networks," *IEEE Communications Magazine*, vol. 40, no. 7, pp. 128–134, 2002.

[7] I. Stojmenovic and X. Lin, "Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, pp. 1023–1032, 2001. [Online]. Available: http://dx.doi.org/10.1109/71.963415

[8] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *SIGCOMM Computer Communication Review*, vol. 18, pp. 35–42, August 1988.

[9] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai, "Wireless sensor networks for emergency navigation," *Computer*, vol. 39, pp. 55–62, 2006.

[10] J. Li, S. Ji, H. Jin, and Q. Ren, "Routing in multi-sink sensor networks based on gravitational field," in *Proc. 2008 International Conference on Embedded Software and Systems*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 368–375.

[11] V. Lenders, M. May, and B. Plattner, "Service discovery in mobile ad hoc networks: A field theoretic approach," *Pervasive and Mobile Computing*, vol. 1, no. 3, pp. 343–370, 2005.

[12] A. Tanenbaum, *Computer Networks*. Prentice Hall, 2002.

[13] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[14] Q. Li, M. De Rosa, and D. Rus, "Distributed algorithm for guiding navigation across a sensor network." in *Proc. 9th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, 2003.

[15] C. Tschudin, "Fraglets - a metabolistic execution model for communication protocols," 2003.