



ANSI: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks ^{☆,☆☆}

Sundaram Rajagopalan ^{*}, Chien-Chung Shen

*DEGAS Networking Group, Department of Computer and Information Sciences, University of Delaware,
Newark, DE 19716, USA*

Abstract

We present a hybrid routing protocol for both pure and hybrid ad hoc networks which uses the mechanisms of swarm intelligence to select next hops. Our protocol, Ad hoc Networking with Swarm Intelligence (ANSI), is a congestion-aware routing protocol, which, owing to the self-organizing mechanisms of swarm intelligence, is able to collect more information about the local network and make more effective routing decisions than traditional MANET protocols. Once routes are found, ANSI maintains routes along a path from source to destination effectively by using swarm intelligence techniques, and is able to gauge the slow deterioration of a link and restore a path along newer links as and when necessary. ANSI is thus more responsive to topological fluctuations. ANSI is designed to work over hybrid ad hoc networks: ad hoc networks which consist of both lower-capability, mobile wireless devices and higher-capability, wireless devices which may or may not be mobile. In addition, ANSI works with multiple interfaces and with both wired and wireless interfaces.

Our simulation study compared ANSI with AODV on both hybrid and pure ad hoc network scenarios using both TCP and UDP data flows. The results show that ANSI is able to achieve better results (in terms of packet delivery, number of packets sent, end-to-end delay, and jitter) as compared to AODV in most simulation scenarios. In addition, ANSI achieves this performance with fewer route errors as compared to AODV. Lastly, ANSI is able to perform more consistently, considering the lower variation (measured as the width of the confidence intervals) of the observed values in the results of the experiments. We show that ANSI's performance is aided by both its superior handling of routing information and also its congestion awareness properties, though we see that congestion awareness in ANSI comes at a price.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Swarm intelligence; MANET; Hybrid network; Hybrid routing; Congestion aware routing

1. Introduction

Hybrid ad hoc networks consist of a mixture of mobile, ad hoc network (MANET) nodes and nodes which belong to highly capable infrastructure such as mesh networks or cellular networks. The problem of hybrid ad hoc networks is to make these networks work efficiently without relying on pre-configured

[☆] A section of this work was presented at ICAI 2005, June 2005, Las Vegas, NV, USA.

^{☆☆} This work is supported in part by National Science Foundation under grant ANI-0240398.

^{*} Corresponding author. Tel.: +1 302 831 1131; fax: +1 302 831 8458.

E-mail address: rajagopa@cis.udel.edu (S. Rajagopalan).

network topologies or centralized control. Hybrid ad hoc networks are useful in many situations where impromptu communication facilities are required such as battlefield communications, and disaster relief missions.

Since the problem of hybrid ad hoc networking shares a lot of problems with typical MANET problems, typical routing solutions for hybrid networks start with a MANET routing solution and then apply some optimizations to work for specific scenarios. A number of ad hoc routing protocols have been proposed, for example [1–5], of which some of them, like AODV [1] work on hybrid ad hoc networks. In *proactive* protocols such as [5], nodes in the network maintain routing information to all other nodes in the network by periodically exchanging routing information. Nodes using *reactive* protocols, such as [1,2], delay the route acquisition until a demand for a route is made. *Hybrid* protocols, like [4,6], use a combination of both proactive and reactive activities to gather routes to the destinations in a network—nodes using ZRP, for example, proactively collect routes in their zone, and other routes are collected reactively. In [6], on the other hand, the level of proactive activity and reactive activity are chosen autonomously by the nodes in the network, and proactive activity is only seen around favorite destination nodes. In most traditional reactive protocols, like [1,2], only when a route breaks irreparably does the protocol mechanisms repair the damage. In reality, route deterioration in mobile networks is most often not sudden but gradual,¹ and most often available routes get better/deteriorate gradually and not suddenly. So the routing protocol should continuously maintain information about the nodes in the local area to perform effectively and avoid too many link breakages.

In this paper, we present a hybrid routing suite (with both proactive and reactive components) for hybrid ad hoc networks which uses the mechanisms of swarm intelligence [7] to select good routes to destinations. We use Swarm Intelligence (SI) because SI mechanisms allow for self-organizing systems [8] and maintain state information about the neighboring network better than traditional MANET routing mechanisms. Self-organizing systems are robust environments where erroneous system behavior is corrected autonomously by the coordinated working

of lower-level components. The combination/interaction of lower-level components in SI such as positive/negative feedback and amplification of fluctuations along with multiple interactions are the mechanisms which allow a node to change routing information quickly and efficiently to adjust to an ever-changing local topology and route deterioration, thus initiating fewer link breakages.

Our protocol, ANSI, uses a highly flexible cost function which allows it to use the information collected from the local ant activity, such as the congestion status of the neighboring nodes, in useful ways. In addition, the ant-like working of our protocol allows for the maintenance of multiple routes to a destination. In nodes which use proactive routing in ANSI, this fact is used to perform stochastic routing, and in nodes that use perform reactive routing (pure MANET nodes), when one route fails, others may be used. Our motivation comes from the fact that different networks face different conditions, and thus a protocol suite should allow for various configurations as the network conditions dictate. Furthermore, supporting multiple routes simultaneously is essential to ensure *survivability* of the network [9]. ANSI facilitates ad hoc unicast routing by exploiting route finding behaviors that are *emergent* from ant packets working collectively, rather than explicitly coding them to cope with the problem. We formulate the routing problem at node i as a set of “food foraging” problems from nest i , where each “food source” is a destination d in the network. In this formulation, next hops are evaluated on the basis of the strength of the *pheromone trail*² on the link connecting a node and a next hop.

The remainder of this paper is organized as follows: In the next section, we discuss a number of approaches and protocols which are related to our research. In Section 3, we describe in detail the components of ANSI unicast routing protocol, and follow it with Section 4 where we discuss the results of the comparison of simulated models of ANSI with a popular routing protocol, AODV [1]. We conclude in Section 5 with a brief note on future research effort.

2. Related work

The main ingredients of SI, positive/negative reinforcement, and amplification of fluctuations

¹ Some routes, such as routes to neighbors, break suddenly, when the neighbors go out of range. We are commenting on the general case here.

² The computational equivalent of the chemical deposited on the forest floor by ants.

are achieved in an environment with multiple interactions among nodes [7]. Because the above components of SI are the lower-level components required for self-organizing behavior, the benefits of using SI-based algorithms are not fully accrued if the any of the above lower-level components are not present in a swarm intelligence-based protocol. This is because in any SI-based system, these aspects work together in learning about the network.

In [10] Baras and Mehta describe a swarm intelligence-based reactive ad hoc routing protocol called PERA. PERA uses broadcast *forward ants* as exploratory agents sent out on-demand to find new routes to destinations. Each ant holds a list of nodes that were visited while exploring the network, and since these ants are broadcast at each node, a forward ant can result in several *backward ants*—ants sent by destination nodes in response to forward ants. This uncovers several routes for each forward ant sent, and at each node these multiple routes found to the destinations are maintained as probability values. As with AntNet [11], the routing table R_i at node i is a probability matrix with a probability entry P_{ijd} as the probability that a data packet at i 's FIFO queue will take the next hop j to be routed to d . Positive reinforcement is managed in PERA using forward/backward ants and negative reinforcement is implicit—no explicit aging of the pheromone trails is done. After a route has been established, PERA regularly uses forward ants to find newer routes to destinations. This is wasteful, considering the fact that forward ants cause a lot of network resources to be consumed and should not be sent when not necessary.

In [12], Camara and Loureiro outline a source routing scheme in which the network relies on location information and support from fixed infrastructure. Owing to a source routing approach, the algorithm relies heavily on a source \leftrightarrow destination route which is available at the time of message creation. New nodes in the network start with using their neighbor's routing table. The routing table, generated using shortest path algorithms, on the other hand, may contain information which is outdated. Ants are unicast from a source to specific destinations, for example, the destination node may be the node with the oldest information in the routing table. This mechanism is used to make sure that the routing information in the source is updated and recent. Thereby, ants are used in [12] with the semantics of routing information updates, like classical distance vector protocols such as DSDV or

DBF—ants are not used as feedback agents to reinforce routes positively (in the case when a route is still good), negatively (when a route is no longer good) or explore new routes randomly—ants in this approach are unicast to specified direction, not allowing for amplification of fluctuations, and depending on known metrics such as timestamp of a route in the routing table.

The approach used in [13] by Heissenbttel and Braun also relies on location information, and is a purely proactive routing approach based on dividing the network into logical zones and assigning logical routers to each. Ants—forward ants and backward ants—are used by logical routers in this approach to periodically check if the logical links connecting it to a randomly chosen destination are functional and reflect on the current state of the network surrounding the logical router. Positive and negative reinforcement are achieved by means of multiple interactions and pheromone additions (by forward and backward ants) and pheromone aging, respectively. Random amplification of a new good route in the face of topological fluctuations is possible by random dissemination of ants to destinations.

In [14], Gunes et al. outline ARA, a multipath, purely reactive scheme. ARA uses forward ants and backward ants to create fresh routes from a node to a destination. When routes to a destination D are not known at S , a forward ant is broadcast, taking care to avoid loops and duplicate ants. When a forward ant is received at an intermediate node X via node Y , the ant reinforces the link XY in X to route to all the nodes covered so far by the forward ant. When a forward ant is received at D , a backward ant is created which backtracks the path of the corresponding forward ant. At each node the backward ant is received, the link via which the backward ant is received is reinforced, like the forward ant does, for all nodes which have been visited by the backward ant. In ARA, data packets perform the necessary (positive) reinforcement required to maintain routes. When a path is not taken, it subsequently evaporates (negative reinforcement) and cannot be taken by subsequent data packets. Under the described scheme, amplification of topological and network fluctuations is not possible except under extreme conditions when routes break often.

In [15,16], Di Caro et al. describe AntHocNet, a hybrid, stochastic approach to the routing problem in MANET. AntHocNet is a congestion-aware pro-

protocol which only finds routes on-demand, but once a route is established, the route is proactively maintained. This approach, argued by the authors to be more ant-like [16] than other competing ant-based protocols, will fail to reduce overheads in very high traffic/mobility scenarios, owing to the rate at which proactive ants are potentially unicast when the mobility increases. This is because in high mobility/traffic scenarios, routes get invalidated often and proactive activity has to increase appropriately to keep a valid view of the network for routing, thus increasing the load placed on the network. Indeed, we do agree with the comment that the authors of AntHocNet make regarding the repeated path sampling, and ANSI manages to steer clear from repeated path sampling by carefully choosing when to engage in route discovery activity.

In [17], Wedde et al. present a new routing algorithm for energy efficient routing in mobile ad hoc networks. In their approach, they show that BeeAdHoc, a reactive source-routing protocol inspired by the foraging principles of honey bees, is able to achieve energy consumption characteristics as compared to DSR, AODV and DSDV without compromising on traditional performance metrics such as packet delivery and throughput.

Our protocol, ANSI, is a *hybrid* protocol proposed for *hybrid* ad hoc networks. Some characteristics seen in traditional on-demand routing protocols can be seen in ANSI. For example, an optimization used in AODV, expanding ring search, is also used in ANSI, albeit more efficiently, owing to the use of history information. Unlike traditional MANET protocols which engage in route maintenance/discovery activity only when links break, ANSI continuously updates a node's neighborhood information using data packets and control packets to alleviate the negative effects due to flooding the network with route discovery/maintenance. In addition, unlike traditional MANET protocols, ANSI has a flexible cost function which allows it to perform metric-centered routing. In our implementation, we have performed congestion-aware routing, but it is easy to see how this cost function can be modified to perform, say, energy efficient routing.

When compared to other ant algorithms for MANET routing, we note that to the best of our knowledge, there exists no other ant algorithm for hybrid ad hoc networks, but ANSI is able to perform well in both pure MANET and hybrid ad hoc networks. In addition, the ANSI design understands the advantages of proactive/stochastic routing

in immobile, highly capable infrastructure and applies it only in those nodes, rather than letting pure MANET nodes incur the costs due to the same under high mobility conditions. Lastly, the flexible cost function (specifically, the congestion-awareness property) in ANSI leverages the inherent nature of swarm intelligence by collecting multiple routes and using them to perform load balancing in all sections of the network. This, as we will see, also alleviates the tendency to create hotspots in the network.

3. ANSI unicast routing protocol

3.1. Protocol overview

ANSI is a *hybrid routing protocol* for *hybrid ad hoc networks* comprising of both proactive and reactive routing components. Pure MANET (mobile) nodes in ANSI use *only* reactive routing, and choose routes *deterministically*, while nodes belonging to more capable, infrastructure (immobile) networks use a combination of both proactive and reactive routing and perform *stochastic routing* when multiple paths are available. The outline of the process of ANSI routing is as follows:

1. When a route to a destination D is required, but not known at a node S , S broadcasts a *forward reactive ant* to discover a route to D .
2. When D receives the forward reactive ant from S , it source-routes a *backward reactive ant* to the source S . The backward reactive ant updates the routing table of all the nodes in the path from S to D , allowing for data transfer from S to D .
3. When a route fails at an intermediate node X , X first checks if there are other routes which can be used to route the packet to D . If not, then ANSI buffers the packets which could not be routed and initiates a route discovery to find D by using a *forward reactive ant* to perform local route repair. Additionally, X sends a route error message back to the source node S .
4. Nodes belonging to more capable, infrastructure networks maintain routes to their connected components proactively, by periodic routing updates using *proactive* ants. Nodes belonging to more capable, infrastructure networks also use stochastic routing when multiple paths are available. In addition, each node in the infrastructure collects information about which mobile nodes are connected to which infrastructure node.

5. When a route at D is known at a MANET node S , ANSI *deterministically* chooses the best next hop to reach the destination. If S is part of a highly capable infrastructure, then S may choose to perform stochastic routing to the destination D , depending on the availability of multipath routes.

We claim ANSI will perform better than typical MANET protocols because of the working of the SI mechanisms at each node, which maintain routing information and local information more effectively than traditional MANET routing protocols. In addition, the congestion-awareness of ANSI also helps in controlling the extent of congestion in high traffic scenarios. Lastly, in hybrid networks, ANSI is able to leverage the power of nodes belonging to more capable networks to assist in routing activities of the network. In Sections 3.3.1–3.3.4, we explain the details of the above actions, and show how the SI mechanisms work at each node in maintaining routing information.

3.2. Protocol model

3.2.1. Data structures

Data structure (1) below is the ant structure, carried by all ants, and data structures (2) and (3) below are maintained at each node, and are updated every time an ant arrives at the node.

- (1) *Ant structure*: The following information is carried by an ant π :
 - (a) The ant ID of the ant, which is the (*node ID, sequence number*) pair.
 - (b) The number of nodes, m , which π visits, including the node π originated from.
 - (c) The nodes-visited-stack (adapted from [11]), \mathcal{S}_π , containing information about nodes $V = \{v_1, v_2, \dots, v_m\}$, that can be reached by backtracking the ant π 's movement (using the nodes-visited-stack), and
 - (d) The pheromone amount at $v \in V$, p_v .
- (2) *Ant decision table at node i* , \mathcal{A}_i : (adapted from [18]). An ant decision table is a data structure that stores pheromone trail information for routing from node i to a destination d via k possible next hop nodes $J = \{j_1, j_2, \dots, j_k\}$. The link ij in the ANSI network, between two nodes i and j is assumed to be bidirectional. Routing tables are computed from ant decision tables. Each ant decision table entry \mathcal{A}_{ijd} for node i

maintains a row for the destination-next hop pair (d, j) along with the $\tau_{ijd}(t)$, η_{ijd} , ψ_{ijd} , and a_{ijd} values described below:

- (a) $\tau_{ijd}(t)$ is the pheromone trail concentration left on a trail ij used as a first hop to destination d at current time t due to all the ants that have traversed the trail, taking into consideration the pheromone evaporation (see Eq. (6)). τ is thus a *weighted* measure of how many times the trail ij was traversed by packets intended to d and is thereby a measure of the *goodness* of trail ij .
- (b) η_{ijd} is the heuristic value of going from j to i . In our mapping, η is a measure of the distance to the destination, $dist_{ijd}$, going from i to d , when using next hop j . We set $\eta_{ijd} = 1 + \frac{1}{dist_{ijd}}$.
- (c) $\psi_{ijd} \in [0, 1]$ is the value of the congestion status at node j . If $\psi_{ijd} = 1$, then, node j is considered not congested, and if $\psi_{ijd} = 0$, then the node j is considered congested. The value of ψ at a node j is measured as the ratio of empty space in packets in the IP queue size to the number of packets already in the IP queue at j .
- (d) We see that the goodness of a next hop j is directly proportional to $\tau_{ijd}(t)$, inversely proportional to $dist_{ijd}$ and directly proportional to ψ_{ijd} . Thus, we write:

$$a_{ijd} = (c_\tau \cdot \tau_{ijd}(t)^\alpha) \cdot (c_\eta \cdot \eta_{ijd}^\beta) \cdot (c_\psi \cdot \psi_{ijd}^\gamma) \quad (1)$$

where $c_\tau > 0$, $c_\eta > 0$, and $c_\psi > 0$ are arbitrary constants, and α, β, γ are integers such that $\alpha, \beta, \gamma > 0$.

For our use, we need to normalize the above value of a_{ijd} so that we may gauge the relative effectiveness of each next hop. We normalize it such that $a_{ijd} \in [0, 1]$:

$$a_{ijd} = \frac{(c_\tau \cdot \tau_{ijd}(t)^\alpha) \cdot (c_\eta \cdot \eta_{ijd}^\beta) \cdot (c_\psi \cdot \psi_{ijd}^\gamma)}{\sum_{l \in J} (c_\tau \cdot \tau_{ild}(t)^\alpha) \cdot (c_\eta \cdot \eta_{ild}^\beta) \cdot (c_\psi \cdot \psi_{ild}^\gamma)} \quad (2)$$

where J is the set of next hops at i to destination d . We then set $c_\tau = c_\eta = c_\psi = 1$, and arrive at

$$a_{ijd}(t) = \frac{[\tau_{ijd}(t)]^\alpha [\eta_{ijd}]^\beta [\psi_{ijd}]^\gamma}{\sum_{l \in J} [\tau_{ild}(t)]^\alpha [\eta_{ild}]^\beta [\psi_{ild}]^\gamma} \quad (3)$$

where α, β and γ are chosen appropriately (see Section 4). The above formula was

adapted from Ant Colony Optimization techniques outlined in [18]. The intuition behind this equation is that we want to use the metrics of hop distance and path goodness and allow some flexibility as to how much we rely on either metric by varying α , β and γ values.

As soon as an ant π is received at a node i via neighbor node j , i has the information about j 's congestion status from \mathcal{S}_π . The pheromone τ_{ijv}^π deposited by an ant π and the heuristic η_{ijv} to a destination v in the ant π traversing from node j to node i via nodes $v \in V$ are given by the equations:

$$\tau_{ijv}^\pi = \frac{1}{p_j - p_i} \quad (v, i, j \in V) \quad (4)$$

and

$$\eta_{ijv}^\pi = 1 + \frac{1}{\text{depth}(v)} \quad (v, i, j \in V) \quad (5)$$

where p_i and p_j are the pheromone amounts of ant π at nodes i and j , respectively, and $V = \{v_1, v_2, \dots, v_m\}$ denotes the set of m nodes visited by π . The value $\text{depth}(v)$ is the depth of the node v in π 's nodes visited stack. All τ values in \mathcal{A}_i are evaporated according to Eq. (7) each time another ant, π' , visits node i . Let us say π' traverses the same trail ij at time $(t + \Delta)$ as traversed by π at time t . π' then *positively reinforces* the trail $ij \forall v \in V$ in \mathcal{S}_π . All other trails iJ' , (where J' is the set of all possible next hops from i except j) in the ant decision table \mathcal{A}_i are not positively reinforced, and in the event no ant traverses through any of the other trails iJ' , the trails iJ' eventually become invalidated (*negatively reinforced*) owing to pheromone evaporation. The new τ_{ijv} at time $(t + \Delta)$ is calculated as follows:

$$\tau_{ijv}(t + \Delta) = \text{evaporate}(\tau_{ijv}(t), \Delta) + \tau_{ijv}^{\pi'} \quad (6)$$

where $\tau_{ijv}^{\pi'}$ is the pheromone deposited on the trail by π' over ij (see Eq. (4)). The function $\text{evaporate}(\tau_{ijv}(t), \Delta)$ returns the pheromone amount left on trail ij for destination v (after evaporation) due to the ants which traversed ij before π' . The pheromone evaporation model used to calculate how much of the earlier pheromone trail, $\tau_{ijv}(t)$ is left behind at $(t + \Delta)$ when π' traverses the trail ij is as follows:

$$\text{evaporate}(\tau_{ijv}(t), \Delta) = \frac{\tau_{ijv}(t)}{2^{\Delta/c}} \quad (7)$$

where c is an arbitrary constant. After all the τ values in the ant decision table are evaporated (includ-

ing the $\tau_{iJ'V}$ values on the trails iJ' that were negatively reinforced, i.e., no ants that traveled $V' = \{v'_1, v'_2, \dots, v'_m\}$ were received) and recalculated, the a_{ijv} values for all entries V in \mathcal{S}_π are recomputed and the new best next hops to destinations V are computed again. This is followed by an update of the routing table at node i . Negative reinforcement of routes also happens when a route is explicitly invalidated by a route error message.

(3) *Routing table*: The routing table at node i is a table containing an entry for each destination d reachable from node i along with the *best next hop*, \mathcal{J}_i^d , to d . The best next hop, \mathcal{J}_i^d , to a destination d is the next hop that contains the largest a_{ijd} value in \mathcal{A}_i . The value of \mathcal{J}_i^d is thereby updated every time an ant visits a node i . The routing table also contains the distance of d from i in hops, and this information is used to set the number of hops for route discovery when the routing table entry to d in i becomes defunct. In the case of nodes which are part of highly capable infrastructure, the routing is *stochastic*, and the next hop is chosen directly from the ant decision table *probabilistically*. Specifically, a next hop j at node i for destination d is chosen with a probability of a_{ijd} .

3.2.2. Amplification of fluctuations

The process of broadcasting ants during reactive/proactive route discovery/recovery/maintenance finds new routes to nodes and alters the information in the ant decision table accordingly. Because of the nature of broadcast in the wireless medium, the routes found as a result of forward reactive ant activity reflect the current status of the network and accordingly amplify the current fluctuations in the topology. Another mechanism amplifies the fluctuations in the local area: when a node receives a unicast packet, it notes the neighbor node ID and reinforces the path to the source of the packet via the neighbor. In addition, when a data packet is sent along a next hop, the node reinforces the next hop as a valid next hop to the destination. This mechanism also amplifies local fluctuations of network and topological characteristics and see to it that the nodes in the ANSI network use up-to-date network and topological information.

Some protocols, for e.g., [10], using SI mechanisms for MANET argue for unicasting forward reactive ants along one randomly chosen path to the source and destination to amplify the fluctua-

tions in the network. Yet others, for e.g., [16], argue for sending forward ants at regular intervals from the source while the source is sending packets to the destination. We feel that the above methods will work in low traffic scenarios and in wired networks, where there is little or no mobility, but not in highly mobile MANET with high loads. Besides, we feel that the right model for amplification of fluctuations in a MANET using SI mechanisms is the model we use: that of broadcasting forward ants *only* when absolutely needed both at the source and intermediate nodes (to perform local repair), and using these mechanisms with a neighbor discovery mechanism, and applying the rules of SI on the data collected (viz., Eqs. (3)–(7)). By using the SI mechanisms appropriately in ANSI, we are able to reduce the number of MAC layer resources used wastefully, as well as be responsive to a network with high traffic rates, and provide better packet delivery rates and lower delay jitter characteristics.

3.3. Protocol description

A trail ij to destination d , τ_{ijd} , is positively reinforced in ANSI when (a) a new route to a destination d is found (via ant activity) at i via next hop (neighbor node) j , and (b) when i uses an already known nexthop node j again to route a packet to d . A trail ij is negatively reinforced when (a) the trail ij to destination d is subjected to evaporation (as per Eq. (7)), and (b) when next hop node j to d is no longer available (owing to MAC layer errors, route errors, or congestion at j). In the following sections, we describe the various reinforcement mechanisms at work in ANSI.

3.3.1. Local route management—reinforcement by data packets and the use of neighbor discovery HELLO messages

Local route management is made possible by reinforcement due to both movement of data packets and an explicit neighbor discovery mechanism. These two concepts are illustrated in Fig. 1.

When a data packet arrives at a node i via a neighbor node p and is sent to the destination along next hop j , both the trail to the previous hop, ip , and the trail to the next hop, ij , are reinforced by the SI mechanisms at i .

In addition, nodes in ANSI periodically broadcast a HELLO message. This message can contain a variety of information about the node sending the message, such as congestion status. In ANSI,

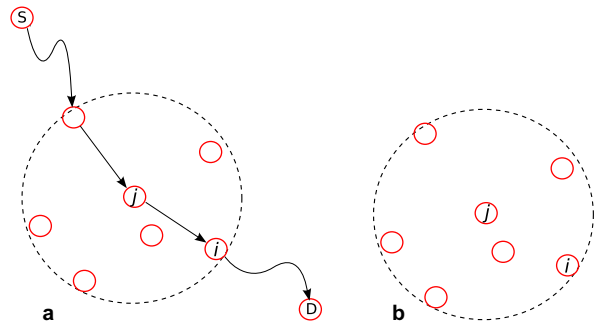


Fig. 1. Local reinforcement in ANSI. (a) Reinforcement by data packets. Node i , upon receiving a data packet from S via node j , reinforces the path to node j via j and the source S via j . (b) Reinforcement in neighbor discovery mechanisms. Upon receiving a HELLO beacon from j all nodes i reinforce trails via j .

hello packets are used to perform local route management by positively reinforcing previously known neighbors and new neighbors. The advantage of using this mechanism can be explained as follows: If a direct route to a destination d is known at i via this process, then a previously known indirect route to d is less favored than the direct route by the reinforcement mechanisms in ANSI. Note that HELLO messages are sent via all available interfaces to facilitate neighbor discovery over all possible paths.

3.3.2. Non-local route management and explicit positive reinforcement

Reactive route discovery is performed by *forward reactive* ants, π_f , and *backward reactive* ants, π_b . Reactive route discovery can be used both at the source of a data packet and at an intermediate node looking for an alternate route to the destination in the event that previously known routes to the destination have proved ineffective. A route request is sent by deploying a forward reactive ant π_f and the route reply is sent using a backward reactive ant, π_b . Even though multiple routes can be gathered by a source sending forward reactive ants (by allowing the destination to send backward reactive ants in response to all copies of the forward reactive ants received), we allow the destination to send a backward reactive ant only for the first forward reactive ant received. This is because we found that in a high traffic/mobility scenario in which a MANET node has many routes to the destination, packet delivery from source to destination can suffer invariably because using several routes will spread the traffic over more nodes, and increase

the contention in the network. In this case, it seems like using one route deterministically, while keeping tabs on the congestion status of neighboring nodes (which is what ANSI does) is a better approach.³

Regardless, multiple routes are collected owing to the interaction of the ant information from the nodes in the network and HELLO beacons, and are used as and when older routes become defunct. Also, note that regardless of collecting information about multiple routes via other mechanisms, ANSI uses a *deterministic* choice of next hops when using pure MANET nodes (highly capable nodes collect multiple routes and use stochastic routing, as we will see later). This is because we found that stochastic approaches in MANET nodes using ANSI are not suited to high data delivery in high traffic scenarios.

In Fig. 2, consider a node S which needs to route data packets to D , but does not have a route to D . Node S buffers the data destined for D and broadcasts (over all interfaces) a forward reactive ant, π_f^{SD} (with a nodes visited stack \mathcal{S}_f), intended to discover the route to D . Because there is a good chance that D has moved, the current implementation of ANSI sets the number of hops, ϕ_f , for the forward reactive ant (sent from S) to be a few hops larger than the last known distance of S from D , which can be obtained from the routing table at S . If S receives data intended to D after π_f^{SD} has been broadcast, S buffers the data. When D receives π_f^{SD} , D copies the nodes visited stack, \mathcal{S}_f , into a new backward reactive ant, π_b^{DS} , and kills π_f^{SD} . D then sends π_b^{DS} to S . π_b^{DS} is not broadcast, it just backtracks to the source S by using the nodes visited stack \mathcal{S}_b in π_b^{DS} . The ant, π_b^{DS} , when visiting a node X along the path to S *positively reinforces* the route to all nodes $v \in \mathcal{S}_b$ upstream from X to D , and adds an entry in \mathcal{A}_X to D via the next hop immediately upstream (in the path from S to D). An intermediate node thereby knows what next hop to use to route to D . In this way, backward reactive ants perform explicit positive reinforcement of routes to destination D . When S receives π_b^{DS} from D , S sends the buffered packets intended for D over the newly discovered route and flushes S 's buffer. Note that multiple paths may be readily collected (for example, by sending another backward reactive ant for the ant proceeding to D via nodes

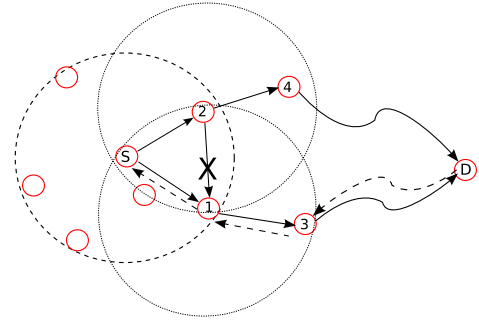


Fig. 2. The propagation of the forward reactive ant (shown in solid arrows) and the return of the backward reactive ant (dashed arrows). The rebroadcast from node 2, when received at node 1 is killed immediately to prevent route loops. At each node X the forward reactive ant enters, it reinforces the path from X to all the other nodes in the nodes-visited-stack. Thus, the forward reactive ant from S when received at node 4 reinforces the trail 4–2 to both node 2 and node S . On the return path, all nodes in path of the backward reactive ant reinforce the trails to the paths to all the nodes in the path leading from the node upstream all the way to the destination. Thus, when the backward reactive ant is received at S via path 1–3, ..., D , S will reinforce trail S – 1 for destinations 1, 3, ..., D .

2 and 4), but this is not done for pure MANET networks.

In the event that π_b^{DS} is not received at S within a timeout period, then the value of ϕ_f is increased by 2 more hops and the search for the route resumes again. The process of route discovery is continued again if a route is not found after the second try. ANSI retries twice for a route to destination.

To control the amount of MAC layer usage at a node X , a scheduled HELLO packet is broadcast at X only if the last broadcast forward reactive ant was sent before the last HELLO message.

3.3.3. Route errors, and negative reinforcement

Route errors occur at a node X when X is unable to provide a route for the destination D owing to non-availability of a routing table entry at X or due to the non-availability of the next hop suggested by the routing table entry at X . When a route error occurs at a node X in a network running ANSI, X first buffers the packet which X needs to forward and then sends a forward reactive ant to find the destination D . If X happens to be an intermediate node, in addition to sending a forward reactive ant, X also sends a route error back to the source S of the packet. The packets buffered at X are relayed across the network after a backward reactive ant from D reaches X .

³ Stochastic approaches to routing in pure MANET networks is an effective approach when the mobility and traffic in the network are low.

In addition, when a route error is received at an intermediate node between X and S , the node explicitly invalidates the routing table entries to D . The packets received at X before the route error is received at S are X 's responsibility (to forward), but the packets generated after the time when the route error is received at S from X are S 's responsibility— S generates a forward reactive ant to find the route to D .

3.3.4. Proactive routing within highly capable sections of the network

As mentioned in Section 3.1, nodes belonging to non-mobile, highly capable infrastructure, such as cellular networks engage in proactive routing as well as reactive routing because these nodes are not concerned about topological fluctuations. These nodes also maintain a list of mobile nodes which are accessible from each other, thus assisting the reactive routing process within the mobile nodes as and when possible. Nodes in non-mobile, highly capable infrastructure send proactive ants periodically to all the other highly capable nodes they are connected to. Proactive ants are not returned like forward reactive ants, and they reinforce the route to the proactive ant sender along the path the proactive ant takes. Proactive ants, apart from carrying a nodes-visited-stack for gathering information about the nodes that were visited, are fixed in hop length and also carry a data structure for indicating the mobile nodes which are accessible from the proactive ant sender. These nodes engage in proactive route collecting activity using all their interfaces, and so are able to combine routes found via different interfaces effectively during the routing process.

Because nodes belonging to a highly capable network need not be concerned about the issues due to mobility, these nodes are able to effectively utilize the benefits due to stochastic routing (see Fig. 3). As mentioned before, ant-based routing naturally lends itself to stochastic routing because multiple routes are found and maintained.

3.3.5. Driving the routing process via more desirable nodes

By choosing higher values for α , β , and γ , the process of next hop selection in ANSI favors the next hops with higher values for τ , η , and ψ , respectively. However, by choosing values which are too high, the route selection is too skewed towards the best next hops and it becomes very difficult for the SI mechanisms at the nodes to respond quickly to

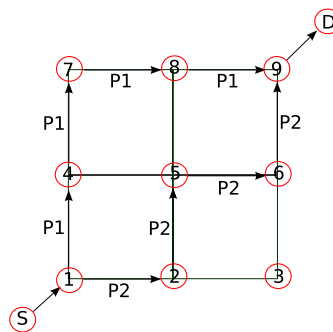


Fig. 3. In a hybrid network (nodes 1–9 are part of a highly capable network in this case, and are connected by the grid shown), nodes belonging to more capable infrastructure are able to perform stochastic routing. In this figure, two possible paths that 1 can take to route to D are one via nodes $1 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9$ ($P1$) and another via nodes $1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 9$ ($P2$).

the changes in the network. Hence, a choice for α , β , and γ should be made carefully to allow for responsiveness of the system. Using insights from our preliminary results, we arrived at a value of $\alpha = \beta = \gamma = 2$, and these are the values we use in our implementation.

4. Simulation results

ANSI was simulated in QualNet (Version 3.7), and the performance of ANSI was compared with a popular routing protocol, AODV [1], for the same network and load characteristics. We chose to compare ANSI with AODV because AODV has been shown to perform well in a vast majority of ad hoc network scenarios. In addition, AODV also works on hybrid ad hoc networks. Our work here is an extension of our earlier work [19] which only tested ANSI under UDP loads over a pure MANET. As we mentioned earlier, ANSI functions as a purely reactive protocol in a pure MANET environment.

4.1. Simulation and network model

4.1.1. ANSI parameters

The current implementation of ANSI used $\alpha = \beta = \gamma = 2$. In both AODV and ANSI, the reactive route recovery is retried twice, and for ANSI, the last try uses $\phi_f = 15$. For the first two trials in ANSI, ϕ_f is determined according to the information available about the unknown destination: if the destination had a valid entry in the routing table earlier, then ϕ_f is set to one more than the earlier

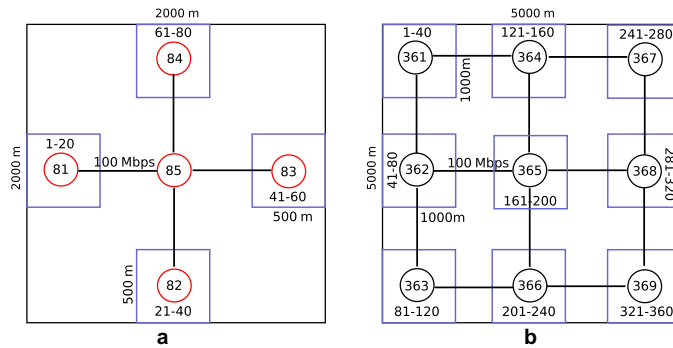


Fig. 4. Hybrid network topologies used for Experiments 1–3: (a) The hybrid network topology for Experiments 1 and 2. (b) The hybrid network topology for Experiment 3.

number of hops to the destination. Otherwise, $\phi_f = 5$. The evaporation constant, c , used in Eq. (7) is 15 s. In hybrid networks, the nodes which are part of the high-speed Ethernet (see Section 4.1.2) used a proactive route update interval of 10 s.

4.1.2. Network and application parameters

We performed five experiments, in which we studied the performance of ANSI and AODV with increasing traffic and increasing number of nodes of both UDP and TCP flows in both hybrid and pure MANETs. In all these experiments, the source and destination are chosen randomly and are pairwise-distinct for each trial.

In *Experiments 1 and 2*, we studied the performance of ANSI vs. AODV in a hybrid network, for both UDP (Experiment 1) and TCP (Experiment 2) flows. In these experiments, the non-mobile nodes are connected to each other over a 100 Mbps Ethernet link. Fig. 4(a) shows the simulation topology. The size of the entire terrain is 2000 m \times 2000 m. Inside this terrain, there are four MANET “regions”, each of which contain 20 MANET nodes inside a terrain of size 500 m \times 500 m, and “served” by one highly capable, immobile node (nodes 81–84) located in the center of the mobile region. This highly capable node, located in the center of each of the regions, manages both an Ethernet interface and an 802.11 interface, and is connected to the others by another highly capable node, node 85, which has 4 Ethernet interfaces. Note that MANET nodes within a region are not able to communicate with MANET nodes of other regions directly (the closest they can get is around 353 m, which is beyond the transmission range of the MANET nodes).

Four streams are chosen for each region, with one stream headed towards each of the regions

(thus, one stream will be an “internal” stream). There are thus, altogether, 16 traffic streams in this experiment. Each of these streams send 512-byte packets at a uniform rate of 1–20 packets/s.

In *Experiment 3*, we studied the performance of ANSI and AODV in a larger hybrid network consisting of 360 pure MANET nodes spread over 9 mobile regions uniformly located in a 5000 m \times 5000 m terrain, each of size 1000 m \times 1000 m and serviced by one highly capable, immobile node (nodes 361–369) located at the center of each mobile region. The highly capable nodes are all connected via a 100 Mbps Ethernet link. The topology of our experiment is shown in Fig. 4(b). Each highly capable node has both Ethernet interfaces and an 802.11 interface. The size of the data packets sent was 512 bytes. Six UDP streams are randomly generated, with the following profile of the source–destination pairs: (a) regions 1–4, (b) regions 1–7, (c) regions 8–5, (d) regions 8–2, (e) regions 3–6, and (f) regions 3–9. The data sources generated packets at the uniform rate of 2 to 20 packets/s in steps of 2 packets/s.

In *Experiment 4*, we studied the effect of increasing TCP traffic in a pure MANET network. In this experiment, 50 nodes were placed uniformly in a network of size 1100 m \times 1100 m. This maintains a node density⁴ of 8.15 m⁻², which, according to [20], is sparse for a network with mobile nodes. The experiment simulates 25 streams of TCP traffic sending 64-byte packets at a uniform packet rate varying from 1 to 20 packets/s.

⁴ Node density is defined as the number of nodes in an area covered by the transmission range of a node.

In *Experiment 5*, we studied the performance of ANSI and AODV under UDP loads in a pure MANET environment with an increasing number of nodes. The number of nodes was varied from 50 to 250 and exactly half the number of nodes were data sources. The terrain size was such that the node density was constant at 8.15 m^{-2} (for example, for 50 nodes, the terrain size was $1100 \text{ m} \times 1100 \text{ m}$). The data sources generated one 64-byte packet a second to be sent to the data sink.

In all the experiments, the MANET nodes were uniformly distributed initially in the terrain and the mobile nodes moved as per the Random Waypoint Model with a minimum speed of 0.001 m/s, maximum speed of 20 m/s, with a pause time of 10 s. In hybrid networks (Experiments 1–3), the mobile nodes were restricted to move only within their region (bounded by a $500 \text{ m} \times 500 \text{ m}$ terrain for Experiments 1 and 2 and a $1000 \text{ m} \times 1000 \text{ m}$ terrain for Experiment 3). The MANET nodes in the experiments used one 802.11 interface with omnidirectional antennas and a transmission range of 250 m at the physical layer and 802.11DCF at the MAC layer. The link bandwidth for the mobile nodes using 802.11 was 2 Mbps. In addition to using 802.11, the non-mobile nodes also used Ethernet with a capacity of 100 Mbps. The simulations used a two-ray pathloss model and no propagation fading model was assumed. The application used was CBR, and sources and destinations were pairwise distinct and chosen randomly. Both TCP and UDP-based CBR flows were studied. Super application was used for generating a reliable CBR traffic stream using TCP (regular CBR application uses UDP).

All experiments were run for a simulated time of 5 min and all sources started sending packets at exactly 40 s into the simulation and ended data generation at exactly 260 s. TCP-LITE, used for the TCP flows in our experiments, is a variant of TCP-RENO, and used an MSS of 512 bytes, maximum send/receive buffer of 16384 bytes each, and delayed ACKs.

We studied the following end-to-end and network-wide characteristics:

1. (*End-to-end metric 1*) *Packet delivery fraction*: measured at the application layer as the ratio of the total number of packets which were received (at the application layer) at the data sinks to the total number of packets that were sent from the data sources (at the application layer), and aver-

aged over the number of source–destination pairs. For TCP flows, the above described quantity is the *measured* packet delivery ratio. The *actual* packet delivery for TCP flows is calculated using the *expected* number of packets that should be sent at the application layer at the data sources. UDP does not perform congestion-control so the expected and measured number of packets sent at the application layer of the data source are the same.

2. (*End-to-end metric 2*) *End-to-end delay*: measured as the average delay in sending packets from source to destination and averaged over the number of source–destination pairs.
3. (*End-to-end metric 3*) *Delay jitter*: measured as the average variance of the interarrival times at the destinations and averaged over the number of source–destination pairs.
4. (*End-to-end metric 4*) *Number of packets sent by Super application sender*: measured as the total number of packets which are actually sent by Super Application senders. For Super Application using TCP, this number depends on how long the TCP connection lasts.
5. (*End-to-end metric 5*) *Variation of the congestion window of a sender*: measured as the TCP congestion window (`snd_cwnd`) at *one* sender for one flow for *one* trial as it varies with simulation time.
6. (*Network-wide metric 1*) *Total number of route errors initiated*: is the total number of route errors generated in the network.
7. (*Network-wide metric 2*) *Total number of 802.11 DCF MAC layer unicasts sent*: is the total number of all (successful) 802.11DCF unicast transmissions sent in the network. For AODV, this measures the total number of data packets, RREP and RERR sent out at the 802.11DCF interface. For ANSI, this measures the total number of data packets, backward reactive ants, and RERRs sent at the 802.11 interface.
8. (*Network-wide metric 3*) *Total number of 802.11 DCF MAC layer broadcasts sent*: is the total number of all 802.11DCF broadcasts sent by all nodes in the network. For AODV, this measures the total number of RREQ and Hello packets sent at the 802.11DCF interfaces, and for ANSI, this measures the total number of forward reactive ants, proactive ants and the Hello packets sent at the 802.11 interfaces.

We do not report end-to-end delay and delay jitter for TCP flows as these metrics are typically not

reported for TCP flows, because of the fact that TCP has to deal with out-of-order deliveries and the large delays (as compared to UDP flows) owing to reliability and congestion-control mechanisms.

We analyzed the results from the above experiments and show them using graphs with 95% confidence intervals of the measured values.

4.2. Simulation results

4.2.1. Experiment 1: Hybrid network—effect of increasing the UDP packet rate

Fig. 5 shows the results for the performance of ANSI vs. AODV over a hybrid network using UDP flows. We see that ANSI consistently outperforms AODV in terms of packet delivery, delay, jitter and number of RERR initiated. ANSI and

AODV send a comparable number of MAC unicasts. ANSI sends fewer MAC broadcasts when the packet rate is low to moderate, but as the packet rate increases, ANSI sends more MAC broadcasts.

The reason why ANSI performs better than AODV—delivering more packets with better metrics such as delay, jitter and number of route errors—is because ANSI manages the local network information better than AODV does, and performs congestion-aware routing. This is why ANSI shows lower route errors as compared to AODV (see Fig. 5(d)). Owing to the above reasons, routes break less often and result in fewer route request operations in ANSI as compared to AODV. When routes do break in ANSI, they are managed by the protocol mechanisms locally rather than a network-wide flooding. This in turn results in lower congestion at the nodes. This is why, even though ANSI shows

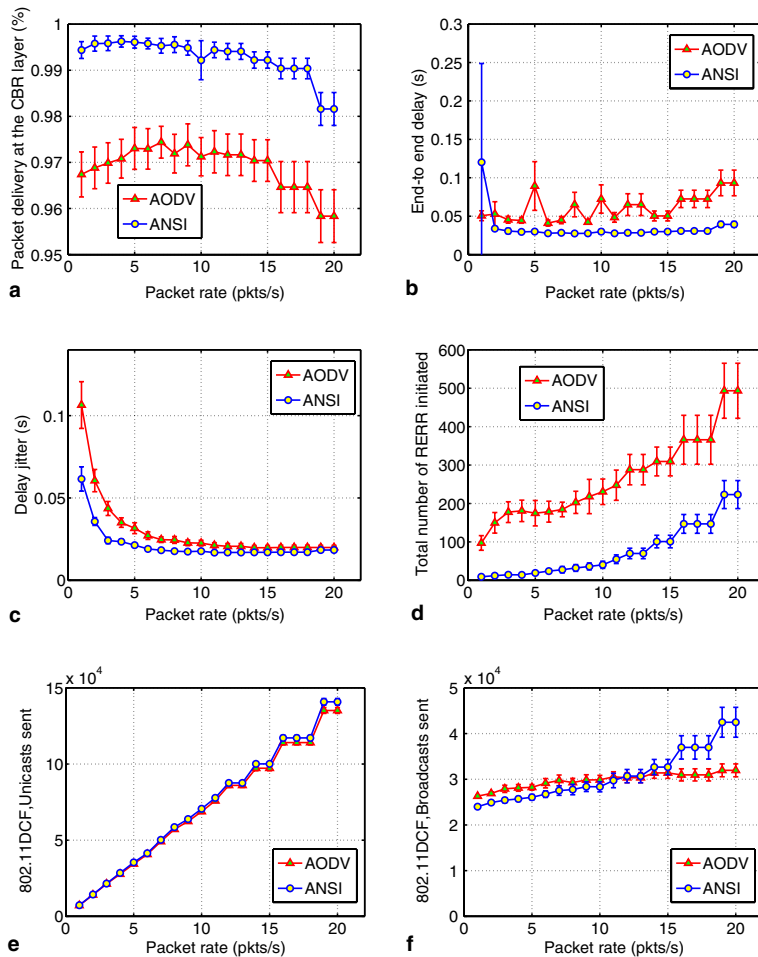


Fig. 5. Experiment 1: Performance studies of ANSI vs. AODV in a hybrid network with UDP flows: (a) Packet delivery ratio, (b) end-to-end delay, (c) delay jitter, (d) number of RERR initiated, (e) 802.11DCF, Unicasts sent and (f) 802.11DCF, Broadcasts sent.

larger MAC broadcasts in higher packet rates, it still shows delays and jitter lower than that for AODV. The higher number of broadcasts when the packet rate increases is because of the congestion-aware properties of ANSI, which allow it to drop badly congested routes and look for new ones. This, while delivering packets more quickly and smoothly, obviously makes ANSI incur more route discovery overheads, which is what we see in terms of larger MAC broadcast overheads. The new, congestion-free (or low congestion) routes are then used to deliver more packets in ANSI. Note that AODV, does not show an appreciable increase in the number of MAC broadcasts as the packet rate increases because it does not perform congestion-aware routing, but owing to this, the performance of AODV degrades. The fact that the number of ANSI's MAC unicasts are comparable to that of AODV (in the context of better performance metrics), along with its fewer route errors is a clear indication of the fact that ANSI is engaged in providing/finding better routes as compared to AODV.

The reason why delay jitter decreases with increasing packet rate in both ANSI and AODV (see Fig. 5(b) and (c)) is as follows: Delay jitter is a measure of the variation of interarrival times at the destination. Thus, if end-to-end delay measured at the destination varies very little, then delay jitter is bound to be low. ANSI, being congestion-aware, chooses congestion-free routes and delivers packets at the destination with little variation in end-to-end delay. AODV, because it is not congestion-aware, delivers packets along congested routes, which results in higher end-to-end delays because a node running AODV does not react to congestion until a congested node along the path is no longer able to receive or transmit packets. Thus, for a single stream of UDP traffic from one source to destination in AODV, the destination first experiences low variation in end-to-end delay, but thereafter, the path becomes more congested and the variation in end-to-end delay progressively increases until the path breaks. AODV then engages in route discovery and finds a congestion-free path, and once again the measurement of end-to-end delay at the destination shows low variation until the new path becomes congested again. Statistically, the value of delay jitter depends on the percentage of the packets that are delivered at the destination with low variation in end-to-end delay, and so if a higher percentage of packets are delivered with a higher variation, the jitter is bound to be larger.

As packet rate increases, for both AODV and ANSI, the mean time before links break owing to node mobility is still the same, but because of the use of highly capable nodes (which are within 2 hops away for any MANET node), the percentage of packets delivered with lower variation in end-to-end delay increases (in comparison to the number of packets delivered at higher variations in end-to-end delay) at both AODV and ANSI, thus bringing the overall variation down. This is why we see a decrease in delay jitter as packet rate increases for both ANSI and AODV.

4.2.2. Experiment 2: Hybrid network—effect of increasing the TCP packet rate

Fig. 6 shows the results for Experiment 2. For TCP flows, we see that ANSI's measured and actual packet delivery ratio is higher than the same metrics for AODV. We also see that for AODV, the measured packet delivery ratio improves as the packet rate increases, but the actual packet delivery ratio decreases. ANSI's actual packet delivery is nearly 5–10% more than AODV's actual packet delivery ratio. ANSI also sends more packets during the simulation as compared to AODV—we see that the number of packets which ANSI sends is very close to the number expected to be sent.⁵ In terms of the effect of the routing protocol on TCP, the congestion window for the output queue at node 53 (for packet rate 1 packets/s, sent from node 53 to node 48) shows steady growth, while AODV's congestion window (for the same stream, output queue at node 53) shows substantially slower growth. ANSI, as before, shows a lot fewer route errors (see Fig. 6(d)). ANSI shows more MAC unicast traffic as compared to AODV. Though ANSI shows lower MAC broadcast traffic when the packet rate is low, it shows more MAC broadcast overheads when the packet rate increases.

The reason why ANSI performs better (with 5–10% higher actual packet delivery ratio) than AODV under TCP loads is because of the congestion-aware routing in ANSI. Owing to this property, ANSI is able to supply congestion-free routes which allow for the smooth passage of ACKs back to the data source, allowing TCP operations to perform smoothly.

We would like to draw attention to the graphs showing the measured packet delivery ratio in Fig. 6(a). These results for measured packet delivery

⁵ This amount is $16 \times 220 \times x = 3520x$ packets total (x is the packet rate), and indicated by the straight line graph in Fig. 6(b).

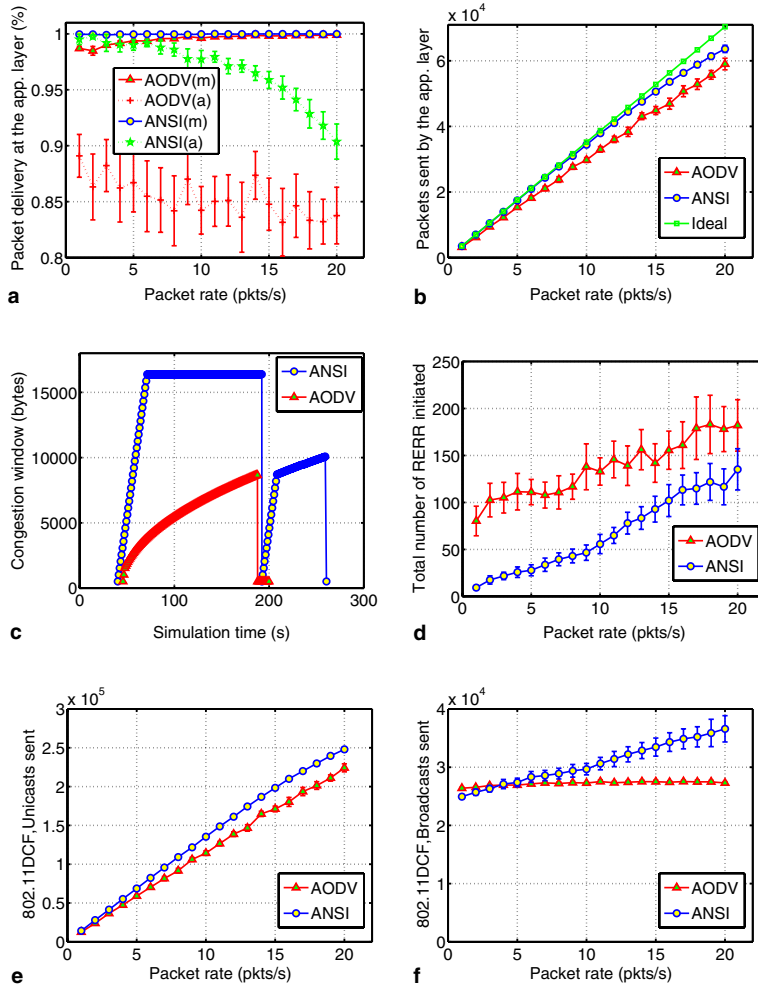


Fig. 6. Experiment 2: Performance studies of ANSI vs. AODV in a hybrid network with TCP flows: (a) measured (m) and actual (a) packet delivery ratio, (b) packets sent by the Super Application Layer, (c) congestion window for TCP output at node 53 for 1 pkt/s, (d) number of RERR initiated, (e) 802.11DCF, Unicasts sent and (f) 802.11DCF, Broadcasts sent.

ratio are counter-intuitive. While the measured packet delivery ratio of AODV increases with packet rate, we note that the percentage of packets sent to the data sink increasingly decreases. Thus, the *actual* packet delivery ratio, measured as the percentage of packets that are received to the percentage of packets that are *expected to be sent* (in this case $x \times 16 \times 220 = 3520x$, where x is the packet rate), actually decreases. So, the traditional packet delivery ratio metrics, defined as the ratio of the number of application layer packets delivered to the number of application layer packets sent, is actually a misleading metric to measure when studying MANET performance under TCP loads.

The behavior of ANSI and AODV under TCP loads can be summarized clearly by Fig. 6(c). Note

that we had fixed the TCP send buffer to be 16,384 bytes, and the congestion window cannot grow beyond this size. In this figure, we see how Super application works TCP when sending CBR traffic. Note that this is traffic inside a mobile region (both node 53 and node 48 are inside the same mobile region as per Fig. 4(a)). TCP, when working on top of ANSI, is able to increase the congestion window as per congestion avoidance algorithms, but in AODV, congestion avoidance is quickly thwarted by congestion occurring along the path from node 53 to node 48, which is why the TCP stack at node 53 shows fast recovery behavior [21] for the TCP output queue. This is the case owing to losing a lot of ACKs in AODV. Indeed, we see that the congestion window in AODV does not grow/change

after a certain point into the simulation (around 200 s) for AODV. Whereas, for ANSI, we see a “healthy” growth of the congestion window, controlled by the congestion-avoiding sender (linear growth of congestion window) rather than being controlled by congestion elsewhere in the network.

This behavior for TCP running over ANSI results from ANSI’s congestion awareness, which constantly maintains routes with low congestion and chooses them in favor of the ones with higher congestion. This permits TCP running over ANSI to receive ACKs more frequently and regularly than in the AODV case, where losing ACKs causes fast recovery behavior. AODV, not being congestion aware, chooses congested routes frequently because it has no way of knowing which routes are congested and which ones are not, making the passage of ACKs more difficult.

More packets are sent by ANSI at the Super application layer as a result of larger congestion windows, and subsequently, more MAC unicasts are sent for ANSI, which is why we see the number of MAC unicasts for ANSI is more. More MAC broadcasts are sent in ANSI as a response to finding newer routes which are less congested. As before, AODV does not respond to congestion, and so it shows only a small increase in the number of MAC broadcasts as the packet rate increases.

4.3. Experiment 3: Large hybrid network—effect of increasing UDP packet rate

Fig. 7 shows the results for the performance of ANSI and AODV in a larger hybrid network. As we can see, the results are similar to the results of Experiment 1, shown in Fig. 5. We also see ANSI’s

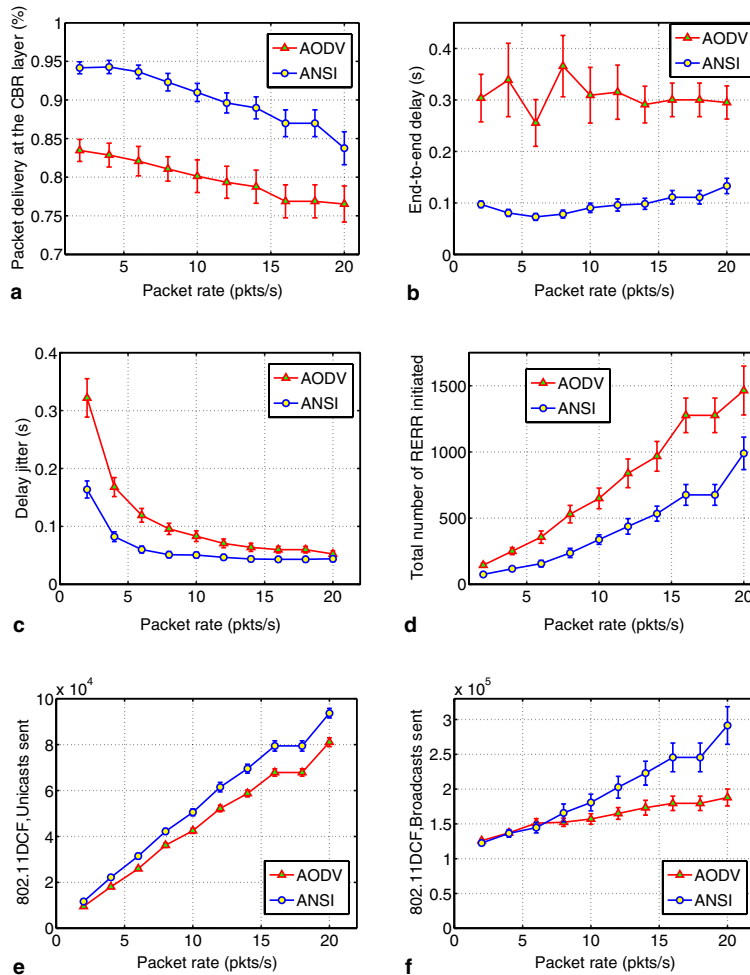


Fig. 7. Experiment 3: Performance studies of ANSI vs. AODV in a (larger) hybrid network with UDP flows: (a) packet delivery ratio, (b) end-to-end delay, (c) delay jitter, (d) number of RERR initiated, (e) 802.11DCF, Unicasts sent and (f) 802.11DCF, Broadcasts sent.

packet delivery metrics are on average around 10% more; ANSI also delivers these packets in roughly 1/3 as much time as AODV, with lower delay jitter, and fewer route errors. As with Experiment 1, we also see that these performance improvements are made in ANSI at the cost of higher MAC layer resource consumption owing to the congestion-aware routing in ANSI.

ANSI's performance in comparison to AODV is better in Experiment 3 than in Experiment 1 is because of the fact that in Experiment 3, ANSI is able to take advantage of proactive routing/stochastic routing in the highly capable nodes.

The reason why packet delivery decreases for both ANSI and AODV more drastically (as packet rate increases) for this experiment as compared to Experiment 1 is because of the effect of a larger

mobile area. In Experiment 1, the highly capable node in each mobile region is in the worst 353 m away ($250\text{ m} \times \sqrt{2}$) from a mobile node, which is 2 hops, but in Experiment 3, the highly capable node is in the worst case 707 m away, which is 3 hops away. Thus, the effects of traffic under high mobility scenarios weigh in more in Experiment 3 than in Experiment 1.

Finally, we note that the graphs for end-to-end delay, Fig. 7(b), and delay jitter, Fig. 7(c), are similar to the corresponding graphs for Experiment 1 for the same reasons.

4.3.1. Experiment 4: Pure MANET—effect of increasing the TCP packet rate

In [19], we showed that the ANSI network is able to do better than the AODV network in a pure

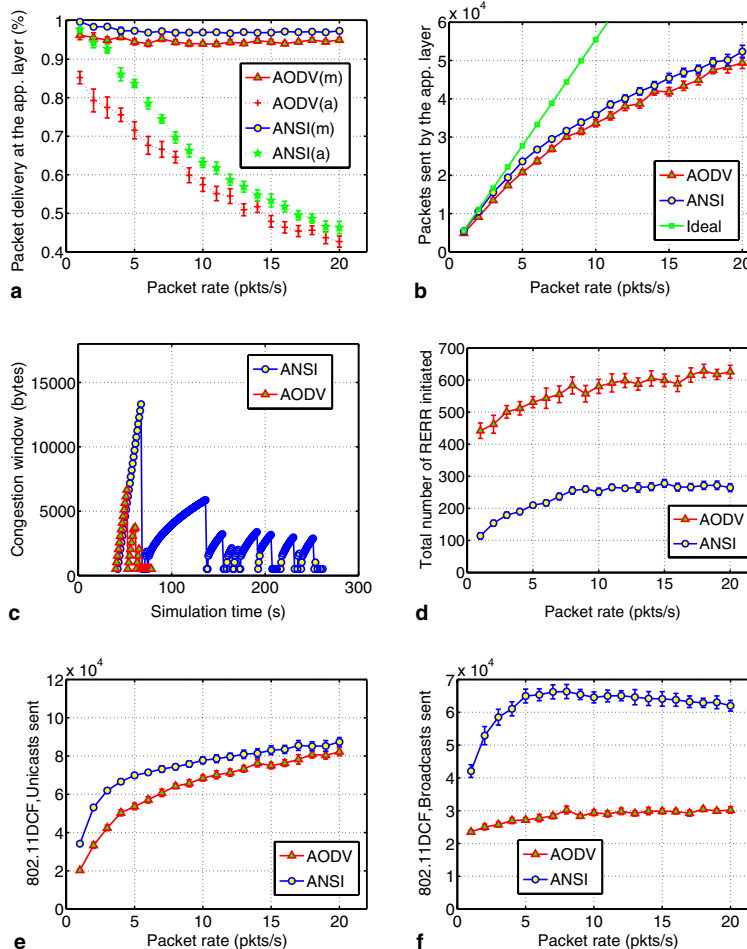


Fig. 8. Experiment 4: Performance studies of ANSI vs. AODV in a pure MANET with TCP flows: (a) measured (m) and actual (a) packet delivery ratio, (b) packets sent by the Super Application Layer, (c) congestion window for TCP output at node 5 for 1 packets/s, (d) number of RERR initiated, (e) 802.11DCF, Unicasts sent and (f) 802.11DCF, Broadcasts sent.

MANET scenario for UDP flows. Here, we show that the ANSI network is able to do better than the AODV network for TCP flows as well. Fig. 8 shows the performance of ANSI and AODV over a pure MANET for TCP flows. We see that ANSI is able to deliver more packets at a higher packet delivery ratio (both measured as well as actual) as compared to AODV, and in Fig. 8(c), we see that TCP over ANSI is able to increase its congestion window, whereas TCP over AODV is not able to do the same. We also see that ANSI shows fewer route errors as compared to AODV. In terms of MAC layer overheads, ANSI shows higher overheads, both with unicast and broadcast MAC traffic.

The packet delivery ratio (measured and actual) and number of packets sent metrics are better for ANSI because it manages its routes better and also performs congestion-aware routing. As far as packet delivery ratio and number of packets sent (see Fig. 8(a) and (b)) is concerned, in Fig. 8(b), we see that the total number of packets sent by the application for either protocol is actually well below the total number which is *expected* to be sent, under ideal cases.⁶ So, we see that the actual packet delivery ratio for either protocol decreases, even though the measured packet delivery ratio is very high. We note however that ANSI is closer to the expected number of packets sent as compared to AODV. As with Experiment 2, we note that studying measured packet delivery ratios for MANETs under TCP loads is misleading without Fig. 8(b).

In Fig. 8(c), we see that ANSI's effect on TCP is better than AODV's effect on TCP—indeed, we see that the AODV congestion window drops to 512 bytes (MTU of the 802.11 interface) and stays there after (about) only 1/4 of the simulation time. We also see, as before, that AODV generates a lot more route errors as compared to ANSI because it has no way of telling which routes are heading towards congestion and which ones are not, making routes break more often. The ANSI network sends more unicast packets as compared to AODV because more packets are delivered in the ANSI network as compared to the AODV network. The number of broadcasts are also higher in the ANSI network. As before, this is owing to the congestion-aware routing performed by ANSI.

⁶ This amount is $25 \times 220 \times x = 5500x$ (x is the packet rate), as indicated by the straight line in Fig. 8(b).

4.4. Experiment 5: Pure MANET—effect of increasing the number of nodes under UDP flows

Fig. 9 shows the results for Experiment 5. We see that ANSI has consistently better (or comparable) packet delivery metrics, fewer RERR, and fewer MAC broadcasts as compared to AODV. We also see that the end-to-end delay, delay jitter, and MAC unicasts for ANSI are more than AODV until about 150 nodes, after which the metrics for ANSI are better or comparable to that of AODV.

We explain the above results as follows: when the number of nodes and the terrain size increases (recall that the terrain size was increased in this experiment to keep node density constant), the average length of the route increases in ANSI owing to its congestion-aware routing. In addition, half the number of nodes in the network are data sources, making the effect of congestion in the network an important factor in deciding routes. This is why we see that ANSI has larger delays (owing to longer paths) with higher delay jitter and more MAC unicasts when the number of nodes increases, until 150 nodes. Under these conditions, the expenses of congestion-aware routing are more than the expenses of not performing the same. Note that owing to the absence of highly capable nodes, the effects due to link breakages in the network only increase as the network increases in size. This is why end-to-end delays and delay jitter increase as the number of nodes increase.

As the number of nodes in the network increase beyond 150 nodes and the traffic increases, the effects due to bad management of routes in AODV far outweigh the advantages due to sending packets via the shortest paths, which creates hotspots. This is why we see ANSI's performance degradation is more gradual and steady, whereas AODV's deterioration in performance metrics is unstable and steep. This insight is corroborated in Fig. 9(d) and (f), where we see that the AODV loses more routes than ANSI does, and thereby engages in route discovery activity more often (as seen by the amount of MAC layer broadcasts sent).

4.5. Discussion

ANSI is able to perform better as compared to AODV owing to a combination of both better route management and congestion-aware characteristics. In addition, in hybrid ad hoc networks, ANSI is

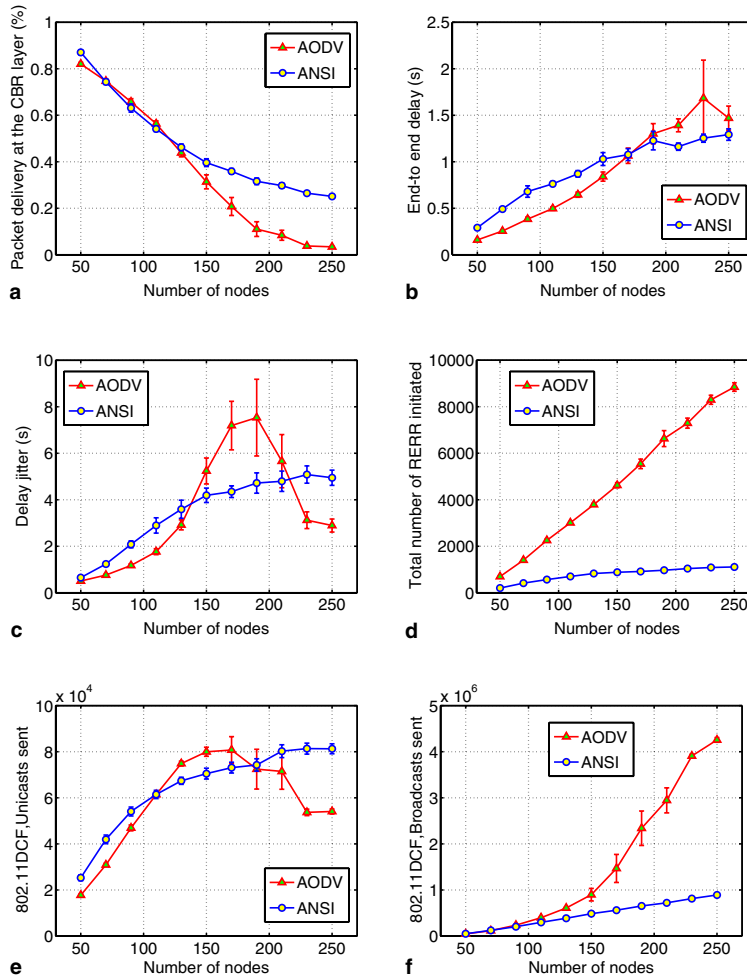


Fig. 9. Experiment 5: Performance studies of ANSI vs. AODV in a pure MANET network with UDP flows and a varying number of nodes: (a) packet delivery ratio, (b) end-to-end delay, (c) delay jitter, (d) number of RERR initiated, (e) 802.11DCF, Unicasts sent and (f) 802.11DCF, Broadcasts sent.

able to harness the power of proactive/stochastic routing in immobile, highly capable nodes connected to each other over Ethernet links. An insight we gain in the experiments above is with regard to congestion-aware routing. We see that there are benefits to doing congestion-aware routing, but it comes at a cost—MAC layer resources. ANSI, being congestion-aware, results in fewer route errors, but incurs a larger number of MAC resources in general. Congestion-aware routing protocols will have to invalidate routes more often than their counterparts which are not congestion-aware. This presents a trade-off between performance and finding congestion-free paths. When the traffic load increases, which is when performing congestion-aware routing is most useful, this is a very delicate

balance. On the one hand, performing congestion-aware routing adapts the network to congestion, but it also decreases the resources available to send data. ANSI is able to reach this balance and thus has significant advantages in low traffic networks, as also in high traffic scenarios.

Another insight is regarding the use of TCP to study MANET routing protocols. Typically, use of TCP over MANETs is a very divisive idea, with a lot of research leaning towards the fact that it is a bad protocol to use over MANET [22]. Regardless of the stand taken by MANET transport layer researchers, we see that studying TCP loads over routing protocols can lead to better understanding of routing protocols. Here, we see that when reliability mechanisms of the routing protocol are used

along with a transport layer mechanism, the results can be very compelling.

5. Conclusions and future work

This paper describes the design, implementation, and performance of a swarm intelligence-based hybrid routing protocol, ANSI, for hybrid ad hoc networks. We simulated ANSI and carried out a performance comparison with AODV, and the results show that ANSI performs better than AODV for both UDP and TCP flows in both pure MANET and hybrid ad hoc networks with respect to packet delivery, number of packets delivered, end-to-end delay, and delay jitter. We also see ANSI affects the upper layer protocols such as TCP and Super Application favorably. In addition, we see that the variance of the observed values (as measured by the width of the confidence intervals) is most often lower in ANSI, indicating a more stable performance.

We also see that implementing congestion-awareness at the routing layer comes at a cost. In the case of the ANSI network, congestion awareness translates to higher route discovery activity because the routing layer invalidates congested routes, allowing TCP to perform more smoothly. Even though congestion-awareness improves the performance of the network, it does raise scalability issues when the traffic increases. In general, we note that it is very difficult to design routing protocols which are scalable under extreme traffic conditions, but incorporating congestion awareness complicates the problem by incurring overheads in an already bogged network. The trade-off between the amount of overhead expended in finding congestion-free paths and the amount of resources remaining for actual data delivery is very delicate at high traffic loads and is worth studying. However, we see that ANSI is able to achieve this balance and perform better than AODV in higher traffic load conditions, even though these come at the cost of network resources.

We believe that the basic ANSI structure provides for implementing a more general purpose, self-organizing routing protocol incorporating autonomously adaptive characteristics that enable it to behave well under all network and traffic conditions. For example, the nodes-visited-stack used in the ants in ANSI can be used to collect a wide variety of information at the nodes the ant visits, such as energy reserves at a node, which in turn can be used to make better next hop selections.

References

- [1] C.E. Perkins, E.M. Royer, S.R. Das, Ad hoc on-demand distance vector (AODV) routing, Internet draft (draft-ietf-manet-aodv-09.txt), November 2001, in preparation.
- [2] D.B. Johnson, D.A. Maltz, Y.-C. Hu, J.G. Jetcheva, The dynamic source routing protocol for mobile ad hoc networks (DSR), Internet draft (draft-ietf-manet-dsr-07.txt), February 21, 2002.
- [3] C. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications (1994) 234–244. Available from: citeseer.nj.nec.com/perkins94highly.html [Online].
- [4] Z.J. Haas, M.R. Pearlman, P. Samar, The zone routing protocol (ZRP) for ad hoc networks, July 2002, IETF Internet draft, draft-ietf-manet-zone-zrp-04.txt.
- [5] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized link state routing protocol, INRIA Rocquencourt, Internet draft (draft-ietf-manet-olsr-09.txt), April 15, 2003, in preparation.
- [6] V. Ramasubramaniam, Z.J. Haas, E.G. Sirer, SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks, in: The ACM Symposium on Mobile Adhoc Networking and Computing (MobiHoc 2003), Annapolis, Maryland, June 1–3, 2003.
- [7] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence—From Natural to Artificial Systems, Oxford University Press, New York, 1999.
- [8] C. Prehofer, C. Bettstetter, Self-organization in communication networks: principles and design paradigms, IEEE Commun. Mag. (2005) 78–85.
- [9] J.P. Sterbenz, R. Krishnan, R.R. Hain, A.W. Jackson, D. Levin, R. Ramanathan, J. Zao, Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions, BBN Technologies, Cambridge, MA, 2002, September.
- [10] J.S. Baras, H. Mehta, A probabilistic emergent routing algorithm for mobile ad hoc networks, in: WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, March 3–5, 2003.
- [11] G.D. Caro, M. Dorigo, AntNet: A Mobile Agents Approach to Adaptive Routing, Universite Libre de Bruxelles, Belgium, Tech. Rep. IRIDIA/97-12, 1997.
- [12] D. Camara, A.A.F. Loureiro, A GPS/ant-like routing algorithm for ad hoc networks, IEEE Wireless Communications and Networking Conference (WCNC'00), Chicago, IL (September) (2000).
- [13] M. Heissenbttel, T. Braun, Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks, University of Bern, 2003, Tech. Rep.
- [14] M. Gunes, U. Sorges, I. Bouazizi, ARA—The ant-colony based routing algorithm for MANETs, in: International Conference on Parallel Processing Workshops (ICPPW'02), Vancouver, BC, Canada, August 18–21, 2001.
- [15] F. Ducatelle, G. Di Caro, L.M. Gambardella, Using ant agents to combine reactive and proactive strategies for routing in mobile ad-hoc networks, IDSIA/USI-SUPSI, Dalle Molle Institute for Artificial Intelligence Galleria 2, 6928 Manno, Switzerland, Technical Report IDSIA 28-04, 2005.

- [16] G. Di Caro, F. Ducatelle, L.M. Gambardella, AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks, in: In Proceedings of the PPSN VIII—Eighth International Conference on Parallel Problem Solving from Nature Lecture Notes in Computer Science, 3242, Springer-Verlag, Birmingham, UK, 2004, best paper award.
- [17] H.F. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior, in: GECCO, June 2005.
- [18] M. Dorigo, G.D. Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Universite Libre de Bruxelles, Tech. Rep. IRIDIA/98-10, 1999.
- [19] S. Rajagopalan, C.-C. Shen, ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence, in: International Conference on Artificial Intelligence (ICAI), Las Vegas, NV, USA, June 27–30, 2005.
- [20] E.M. Royer, P.M. Melliar-Smith, L.E. Moser, An analysis of optimum node density for ad hoc mobile networks, in: IEEE International Conference on Communications, June 2001.
- [21] W.R. Stevens, TCP/IP Illustrated The Protocols, vol. 1, Addison Wesley Professional Computing Series, Reading, Massachusetts, 1994.
- [22] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, in: INFOCOM, April 2003.