Available online at www.sciencedirect.com
ScienceDirect
Ad Hoc Networks 6 (2008) 485–507
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Safari: A self-organizing, hierarchical architecture for scalable ad hoc networking

Shu Du, Ahamed Khan, Santashil PalChaudhuri, Ansley Post, Amit Kumar Saha, Peter Druschel, David B. Johnson *, Rudolf Riedi

Rice University, Department of Computer Science, 6100 Main Street, MS 132, Houston, TX 77005-1892, USA

Received 14 October 2006; received in revised form 8 April 2007; accepted 9 April 2007
Available online 24 May 2007

Abstract

As wireless devices become more pervasive, mobile ad hoc networks are gaining importance, motivating the development of highly scalable ad hoc networking techniques. In this paper, we give an overview of the *Safari* architecture for highly scalable ad hoc network routing, and we present the design and evaluation of a specific realization of the Safari architecture, which we call *Masai*. We focus in this work on the scalability of learning and maintaining the routing state necessary for a large ad hoc network. The Safari architecture provides scalable ad hoc network routing, the seamless integration of infrastructure networks when and where they are available, and the support of self-organizing, decentralized network applications. Safari's architecture is based on (1) a self-organizing network hierarchy that recursively groups participating nodes into an adaptive, locality-based hierarchy of cells; (2) a routing protocol that uses a hybrid of proactive and reactive routing information in the cells and scales to much larger numbers of nodes than previous ad hoc network routing protocols; and (3) a distributed hash table grounded in the network hierarchy, which supports decentralized network services on top of Safari. We evaluate the Masai realization of the Safari architecture through analysis and simulations, under varying network sizes, fraction of mobile nodes, and offered traffic loads. Compared to both the DSR and the L+ routing protocols, our results show that the Masai realization of the Safari architecture is significantly more scalable, with much higher packet delivery ratio and lower overhead.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Ad hoc network routing; Peer-to-peer networking; Landmark routing; Hierarchical routing; Reactive routing; Proactive routing; Hybrid routing; Safari; Masai

1. Introduction

In an ad hoc network, individual, potentially mobile nodes cooperate to form a network without the aid of existing infrastructure such as wireless base stations or access points. Instead, each mobile node acts not only as a host but also as a router, forwarding packets for other mobile nodes, to allow

* Corresponding author. Tel.: +1 713 348 3063; fax: +1 713 348 5930.
E-mail addresses: dushu@cs.rice.edu (S. Du), ahamed_khan@yahoo.com (A. Khan), santapc@gmail.com (S. PalChaudhuri), abpost@mpi-sws.mpg.de (A. Post), amsaha@gmail.com (A.K. Saha), druschel@mpi-sws.mpg.de (P. Druschel), dbj@cs.rice.edu (D.B. Johnson), riedi@rice.edu (R. Riedi).

1570-8705/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2007.04.007

nodes to communicate even if they are not directly within radio transmission range of each other. This infrastructure independence makes ad hoc networks very useful in many scenarios such as disaster relief efforts, battlefields communications, and network connectivity in economically disadvantaged areas of the world.

With the rapid proliferation of wireless devices, the use of ad hoc networking is expected to grow, and with it, the size of ad hoc networks that may be created. At the same time, the field of decentralized, self-organizing distributed systems has seen significant advances in recent years and has opened new alternatives in providing ad hoc network services. Work on these two areas has in the past proceeded largely independently. Our *Safari* architecture brings together these two areas, aiming to create a framework for protocols and algorithms that provides large-scale ad hoc network connectivity, seamlessly integrated with infrastructure networks when and where they are available, supporting mobile and stationary nodes, together with decentralized network services. Safari exploits synergies among ad hoc networking and decentralized distributed systems research.

In this paper, we give an overview of our *Safari* architecture, and we present the design and evaluation of a specific realization of the Safari architecture, which we call *Masai*. We focus in this work on the scalability of learning and maintaining the routing state necessary for a large ad hoc network. Masai consists of a scalable routing protocol and an automatic self-organizing hierarchy formation protocol on which the routing is built. Routing of packets in the network is guided by this hierarchy, and is capable of scaling to large numbers of mobile nodes. We assume that nodes in the ad hoc network are willing to cooperate with each other. Many nodes may be power constrained, but for example, stationary nodes may not be; we strive, however, to make the protocol efficient in its network usage, as doing so conserves network bandwidth as well as power.

The Masai realization of our Safari architecture is based in general on the concept of *landmark routing* [36,35,37] and has similarities to existing protocols that apply landmark routing to ad hoc networks, such as LANMAR [27] and L+ [9]. However, unlike these previous systems, Masai is a *hybrid* protocol, carefully combining proactive and reactive routing mechanisms to substantially increase the network's scalability and the protocol's ability to successfully deliver data packets with very low overhead in spite of high node mobility. We provide a detailed discussion of this and other differences between the Masai realization of Safari and previous systems in Section 6.

Our evaluation in this paper is based on both analysis and simulation, under different network sizes, percentage of mobile nodes, and workloads. Our simulation results demonstrate that the protocol is significantly more scalable than existing protocols.

In Section 2, of this paper, we describe our Safari architecture. The design of the Masai realization of the Safari architecture is presented in Section 3, including the protocols for hierarchy self-organization and routing. In Section 4, we present modeling and analysis results of the Masai realization of Safari, and in Section 5, we give detailed simulation-based performance results for Masai. In Section 6, we discuss related work in the area of scalable ad hoc networking, and we conclude in Section 7.

2. Overview of the safari architecture

In this section, we provide an overview of the Safari architecture. Safari provides a self-organizing network hierarchy, a scalable routing protocol, and seamless integration of infrastructure network components where and when available. Safari also includes an integrated distributed hash table (DHT) service, which supports decentralized network services and applications. This hierarchical structure allows routing in the Safari architecture to be based on a hybrid of proactive and reactive routing information, greatly increasing the network's scalability. Any node in the Safari architecture may be mobile or stationary.

The Safari hierarchy recursively groups nodes into cells, cells into supercells, and so on, based on an automatic self-selection of a subset of the nodes to operate as "landmarks" [36], called *drums* in the Safari architecture. Each drum node transmits periodic *beacon* packets, which are forwarded by all nodes within a well-defined, limited scope in the network. The drums are not otherwise actively involved in routing data packets from any source node to its destination; instead, hearing the beacon packets from drums gives nodes forwarding data packets "a sense of direction" within the network topology of the hierarchical Safari architecture.

In general, for $k \geqslant 0$, level-k cells in Safari are grouped into level-$(k+1)$ cells, and so on, within the recursively defined hierarchy; for simplicity of terminology, we refer to individual nodes as level-0 cells. The lowest level at which drums exist is at level-1; individual nodes at level-0 do *not* operate as drums. We refer to level-1 cells also as *fundamental cells*, as at this level, the cell is composed only of individual nodes.

The drums are likewise organized hierarchically, with a subset of the individual nodes self-selecting to become level-1 drums, and recursively, a subset of level-k drums self-selecting to become level-$(k+1)$ drums. Each level-k drum is at the same time also a level-i drum for all levels $i < k$. Each drum has a unique identifier, and each drum at level-i identifies a cell at level-i. The drum selection is based on a distributed algorithm with no centralized coordination. Nodes of the same level are roughly equally spaced (in terms of hop counts) throughout the entire ad hoc network. As nodes, including possibly drum nodes, move within the network, the hierarchy is maintained through the beacon packets and the automatic self-selection of nodes to operate as drums; over time, the set of nodes that are currently drums is not fixed.

Overall, the periodic beacon packets from each drum node aid the hierarchy formation, give nodes an indication of their topological location within the hierarchy, and provide routing information *toward* the drum's cell. As noted above, the drums do *not* have any special role in data packet forwarding, and they are thus no more loaded with handling data packets than normal nodes. The drum identifiers form a topological location-dependent hierarchical address for each node, which the node stores under its own unique identifier in the network using a distributed hash table (DHT); carefully choosing multiple storage nodes improves robustness and efficiency of lookup.

To route a data packet, the packet is forwarded according to the hierarchical address of the packet's destination, routing recursively at each level *towards* the drum for the destination node's cell. We assume the existence of bidirectional wireless links. To route towards a drum at a given level, packets in Safari are routed following the reverse path of the most recent beacon received from that drum. Once the packet reaches the fundamental cell of the destination, any effective traditional ad hoc network routing protocol can be used, since the size of a fundamental cell is limited.

Table 1
Summary of architectural components described in this paper

Component	Description	Section
Beaconing	Provides proximity and routes toward existing drums	Section III-A
Drum selection	Provides automatic selection of drum nodes	Section III-B
Cell membership	Provides association between nodes and drums to form cells	Section III-C
Routing	Provides routing between nodes in the network	Section III-D

The Safari architecture owes its scalability to the following design features:

- *Self-organization*: Beacons maintain the hierarchy, allow each node to determine its hierarchical address, and provide next-hop routing information. The overhead for disseminating beacons is logarithmic in the size of the network and independent of the traffic load or the level of mobility.
- *Scalable routing*: Each node maintains information about only the beacons it overhears, providing it with next-hop routing information. The amount of state a node maintains is logarithmic in the size of the network.
- *Decentralized operation*: The Safari architecture is fully self-organizing. Participating nodes play symmetric roles. All nodes equally share the load of disseminating beacons.
- *Local view*: Each participating node maintains a local view of its surrounding network, with detailed information about its immediate neighborhood and progressively more coarse-grained information about distant parts of the network.

The following section details a particular realization of the Safari architecture, which we call Masai. The design of Masai consists of specific protocols that provide the *self organization* and *scalable routing* aspects of the Safari architecture. Specific protocols for providing distributed address resolution [10,22], a distributed hash table, and seamless integration of any available infrastructure-based network components are beyond the scope of this paper. The components described in this paper are summarized in Table 1.

3. The masai realization of the safari architecture

Masai is a specific realization of the Safari architecture; this section describes the four basic mechanisms that make up this realization: the *beaconing*

protocol, the *drum level selection algorithm*, the *membership algorithm*, and the *scalable routing protocol*. The first three mechanisms allow the network to self-organize, achieving and maintaining the desired Safari hierarchical structure of the network, even under node mobility, node failures, and partition and merging of networks. The fourth mechanism, the Masai routing protocol, is composed of mechanisms for inter-cell routing, route repair in inter-cell routes, and intra-cell routing.

3.1. Beaconing protocol

Each drum periodically locally broadcasts to its neighbors a *beacon* packet advertising the drum's existence and providing location information. A beacon originating from a drum of level-n is called a level-n beacon. Each beacon contains a *beacon sequence number*, a *beacon level*, a *hierarchical address* that equal those of the originating drum; and a *hop count*, that is set to zero at the originating drum and incremented by each forwarding node. As mentioned in Section 2, a level-n drum is also a level-i drum for all levels $i < n$, and a drum thus originates beacons for all levels for which it is a drum. A drum maintains a single beacon sequence number and increments it for each new beacon that it originates at any level.

A drum at some level-n transmits a level-n beacon every T_n seconds, which is forwarded by all nodes within D_n number of hops from that drum. This forwarding rule allows beacons to reach all nodes that could potentially associate with the originating drum according to the membership algorithm described in Section 3.3. Higher level beacons are emitted at a lower frequency than lower level beacons, since mobile nodes cross-over the larger regions covered by higher level cells less frequently than they cross-over the regions covered by lower level cells. For scalability, T_n and D_n are given by the geometric progressions:

$$\begin{aligned} D_n &= \gamma \times D_{n-1} = \gamma^{n-1} \times D_1 \\ T_n &= \beta \times T_{n-1} = \beta^{n-1} \times T_1 \end{aligned} \tag{1}$$

where $\gamma > 1$ and $\beta > 1$ are system parameters. The value of D_1 is based in general on the largest hop count that the on-demand routing protocol used within fundamental cells can generally support efficiently.

Although a drum originates beacons for all levels for which it is a drum, if some level-n drum is scheduled to originate a level-j beacon and a level-k beacon at (approximately) the same time, for $j < k \leqslant n$, the drum omits originating this level-j beacon, since the range over which the level-k beacon will be forwarded covers the range of the level-j beacon. When a node receives a beacon of some level-k, it treats it also as a beacon of the same sequence number for all levels $i < k$.

In addition, to increase routing scalability (Section 3.4), a level-n beacon is also forwarded by all nodes already associated in the level-$(n+1)$ cell of the originating drum. The exact mechanism by which a cell structure is formed is discussed in Section 3.3. Beacons are forwarded according to the union of the two forwarding rules described above.

Each node stores information from the beacons it receives in a local cache of beacons, called the Drum Ad Hoc Routing Table (DART) for that node. This cache is used for self-organization and for routing. In addition to information from the beacon, a node also stores in its DART the *time of reception of the beacon* and the *neighbor node identifier from which it received the beacon*. The latter allows data packet forwarding along the reverse path of the beacon, while the former is used to keep the cache up to date. Upon receipt of a beacon, the node creates a new DART entry and starts a timer for that entry. Whenever a new entry is created in the DART or the timer for a DART entry expires, the drum level selection (Section 3.2) and the membership algorithms (Section 3.3) are invoked. Given that the distance between drums at level-n, D_n, is a geometric progression, there should be only $O(\log x)$, where x is the number of nodes, DART entries. It is not necessary to keep DART entries for nodes that are reachable via intra-cell routing. Due to these two facts, the memory and processing overhead of the DART should be small, even as the size of the network increases.

As mentioned earlier, a drum has no active role in routing or in maintenance of the hierarchy. The only special function of a drum is to originate beacons, while other nodes forward those beacons. In particular, data packets are routed only *towards* but not necessarily *through* drums, as will be described in Section 3.4. Thus, all nodes share the forwarding workload (for data and beacon packets) equally.

3.2. Drum level selection algorithm

In order to be efficient under dynamic changes in the network such as node mobility or failure, new

drums can arise and existing drums can retire. The drum level selection algorithm is run at a node after each change in the DART at that node. The algorithm ensures that eventually, the DART at a node of some level-n satisfies the following conditions:

(i) The DART contains at least one non-expired beacon of a level-$(n+1)$ drum at most D_{n+1} hops away.
(ii) There is no non-expired beacon of a level-n drum less than $h \times D_n$ hops away ($0 < h < 1$ is a hysteresis factor).

The desired state of the DART is achieved by the node changing its level so as to meet the invariants.

If condition (i) above is violated, the level of the node is changed to $n+1$ and the node waits a random back-off time before it announces its new level with its level-$(n+1)$ beacon.

If condition (ii) is violated, two or more drums of the same level are too close to each other. The drum with the highest node identifier remains at the same level, and the rest of these drums reduce their level by 1. The factor h ($0 < h < 1$) creates a "hysteresis" that prevents oscillations in this drum retirement process. A level-n drum retiring could cause condition (i) to be violated for other nearby nodes, each of which will then increase its level. However, these nodes will be at least D_n hops away from any level-n drum. Since a conflict between drums requires this distance to reduce to $h \times D_n < D_n$, there is no oscillation. Values of h close to 1 causes extra drums to retire immediately, possibly resulting in oscillations as new drums arise to replace the retired drum. Values of h close to 0 allows extra drums to survive in closer proximity, at the cost of more beacon overhead. Exploring appropriate values of h under different levels of mobility is a subject of future work and is not discussed in this paper. Throughout the rest of this paper we choose $h = 0.5$.

3.3. Membership algorithm

The presence of drums induces a natural clustering of nodes. Each node *associates* with a drum of a level-1 greater than its own level, and selects this drum according to the contents of its own DART. Typically, a node associates with the one higher level drum that is the least number of hops away.

A node's association is made unilaterally and is not communicated back to the drum. A node invokes the membership algorithm after it has run the drum level selection algorithm. In its basic version, for a node of some level-n, this algorithm chooses the closest (in terms of hop count) of all drums of level $n+1$ for which this node has a DART entry that has not expired. However, this rule might lead nodes near the cell border to oscillate between drums. In order to prevent such oscillations, this rule is enhanced by assigning each DART entry a *weight* calculated from the frequency, the distance, and the number of beacons received. The node associates with the drum corresponding to the DART entry with the highest weight. In our design presented in this paper, we enforce that the new drum is at least 2 hops closer than the current one, and that at least 3 beacons have been received from the new drum. The membership algorithm cannot ensure that the node associates with a drum that is at most D_{n+1} hops away. This is the duty of the drum level selection algorithm.

The membership algorithm gives a unique ancestry for each node. Using this membership information, each node is assigned a hierarchical address based on the drum structure. This hierarchical address plays a vital role in routing.

The hierarchical membership structure can be viewed as a tree, and every node in the network is assigned a hierarchical address. The hierarchical address of a drum at some level-i is the concatenation of the hierarchical address of the level-$(i+1)$ drum with which it is associated and a randomly generated unique number. Thus, if ADDRESS(X_i) denotes the hierarchical address of some level-i drum, X_i, and PARENT(X_i) denotes the level-$(i+1)$ drum with which X_i has associated, then

$$\mathrm{ADDRESS}(X_i) = \mathrm{ADDRESS}(\mathrm{PARENT}(X_i)) \cdot \mathrm{RAND}(b)$$

where RAND(b) is a uniform random number of b bits, and "·" means concatenation. With a large value of b, the probability that two drums at the same level will chose the same random number can be made negligible. The hierarchical address of any leaf node, L, is given by

$$\mathrm{ADDRESS}(L) = \mathrm{ADDRESS}(\mathrm{PARENT}(L))$$

When a node powers on, it associates with a drum at level-1 and sets its hierarchical address to the hierarchical address of this drum. This implies that all nodes in the same fundamental cell have the same hierarchical address.

At start up of a node, the node has an empty hierarchical address and will forward every beacon. The node randomly chooses a timeout from a fixed window, and if no beacon is received before this timer expires, the node will increase its own level to become a drum and emit a beacon. If many nodes are simultaneously powered on, the spacing due to the random timeout should prevent all nodes from becoming drums. If more drums are chosen than necessary, extra drums will reduce their level as described earlier in Section 3.2. The beacons of the first few drums of a new ad hoc network will reach throughout the network. Once at least two of the nodes have increased their levels to become level-2 drums, the level-1 beacon scope will be confined by the cell structure.

When two ad hoc networks merge, i.e., when nodes of two networks with tree depth (hierarchy height) n and $m \leqslant n$ overhear each other's beacons, then these beacons are not stopped immediately and penetrate the other network. The level-k beacons with $k \geqslant m$, in particular, are forwarded throughout both networks as described in Section 3.1, for the reason that either the beacon or the forwarding node has a hierarchical address of length only m and thus cannot differ in the $k+1$ element of the hierarchical address. The smaller network quickly learns of the high level drum in the other network, and associates with it, updating its hierarchical address in the process. This corresponds to merging a smaller tree at the appropriate level into the larger tree. If the depths happen to be equal, the root of one of the trees will increase its drum level to become the new root, as discussed previously.

3.4. Scalable routing

Routing in the Masai realization of the Safari hierarchical architecture can be divided into two phases. Given the hierarchical address of the destination node, the first phase, called *inter-cell routing*, delivers the packet to the fundamental cell of the destination node. Once the packet reaches this cell, the second phase, called *intra-cell routing*, delivers the packet to the destination node within that fundamental cell.

Inter-cell routing is based on the destination node's hierarchical address and on the beacon records stored in the DART of each intermediate forwarding node. As mentioned previously, we assume the existence of bidirectional wireless links. Inter-cell routing operates by following the reverse paths of the beacons emitted by the drums of the cell at each level in which the destination node is located. Conversely, intra-cell routing can be based on any state-of-the-art on-demand routing protocol, since the size of the fundamental cell is kept small. We choose DSR [17,18] for the intra-cell routing protocol for its demonstrated stability with high performance in small ad hoc networks [7].

When a source node S with hierarchical address $S_n{\cdot}S_{n-1}, \ldots, S_1$ has a packet to send to node D, node S retrieves the hierarchical address $D_n{\cdot}D_{n-1}, \ldots, D_1$ of D using a lookup service, such as the one used in L+ [9]. The efficiency of the lookup service is not considered in this paper, but the achievable network performance is limited by the quality of the information that the service provides, especially when nodes are quickly changing hierarchical addresses due to mobility. If the nodes S and D belong to the same fundamental cell ($S_i = D_i$ for all $1 \leqslant i \leqslant n$), the lookup service is not used, and intra-cell routing is invoked immediately. Otherwise, inter-cell routing is invoked.

With inter-cell routing, the source S adds the hierarchical address of D to the packet header before sending the packet; the destination hierarchical address is thus available to the following intermediate forwarding nodes without a separate hierarchical address lookup. To forward a packet at an intermediate node, the node uses the same logic as the source node to determine whether to continue with inter-cell routing or to invoke intra-cell routing.

Fig. 1 shows an example of Masai routing in which a packet is originated from a source node S to a destination node D. Node S uses inter-cell routing to forward the packet to the first node along the path labeled by "a", which is the reverse path of the beacon originated by the level-2 drum X with which D is associated. Each node along this path likewise forwards the packet until it reaches a node that has a DART entry corresponding the level-1 drum Y with which D is associated; the packet then is routed along the path labeled by "b", which is the reverse path of the beacon originated by this level-1 drum Y. Once the packet reaches a node that is a member of the destination fundamental cell, intra-cell routing is used to deliver the packet to the destination node D along the route labeled by "c"; this path is dynamically discovered by the intra-cell on-demand routing protocol.

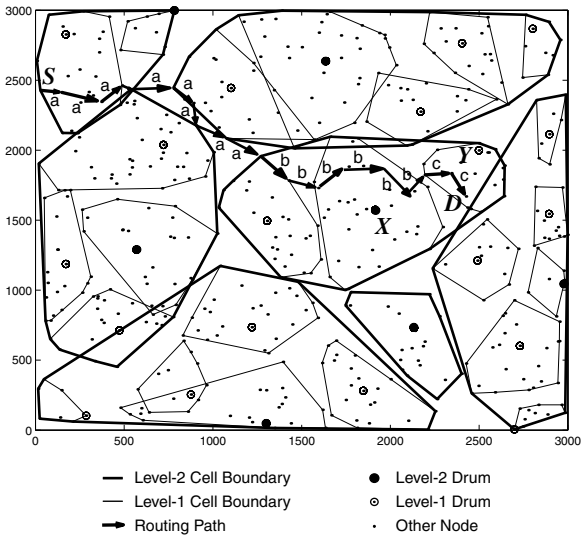


Fig. 1. Masai routing example: a packet from source node S is routed to destination node D.

3.4.1. Proactive inter-cell routing

The basic strategy of inter-cell routing is to follow the reverse path of the beacons that this node has received. When a node receives a beacon, it stores in its DART the node identifier of the transmitting neighbor node from which it received the beacon (along with the information contained in the beacon). When the node has a data packet to send or forward to a specific cell, it uses the DART to determine the next hop toward that cell.

As described in Section 3.1, each drum originates beacons that are forwarded to all nodes in its own cell and to nodes in the cells of its siblings (the same level cells that also share the same higher level drum). This mechanism ensures that any node that is in the same supercell will have the next-hop routing information to all fundamental cells in this supercell.

Unlike some clustered routing protocols in ad hoc networks that assume the existence of cluster heads with special routing functions, our drums are not necessarily part of the route taken for data transmission. As illustrated in Fig. 1, once the packet enters the level-2 cell of drum X, any node in that cell will have the routing information to the fundamental cell of drum Y, and the packet need not be routed through the drum X itself.

For inter-cell routing, each packet contains the following information in its header in addition to the hierarchical address of the destination; a forwarding node updates these fields with its DART entry used for forwarding this packet, until the packet reaches the fundamental cell of the destination node:

- *Prefix match length*: The *prefix match length* between the destination node's hierarchical address and the entry in the DART entry used for forwarding this packet. The prefix match length between two hierarchical addresses $B_n \cdot B_{n-1}, \ldots, B_1$ and $C_n \cdot C_{n-1} \ldots, C_1$ is the largest integer k such that $B_i = C_i$ for all $n - k < i \leqslant n$.
- *Sequence number*: The sequence number of the beacon from the DART entry used for forwarding this packet.
- *Hop count*: The distance in hop count of the current forwarding node to the drum of the beacon from the DART entry used for forwarding this packet.

During the inter-cell routing process, each forwarding node should use the best DART entry it has to deliver the packet to the next hop; this forwarding node must use the same DART entry to update the above three fields in the packet header. When a forwarding node searches its DART for a candidate DART entry to use in routing the packet, the node must guarantee that the candidate provides better routing information than the three fields currently listed in the packet's header: specifically, the prefix match length must be larger, or, if it is the same, the sequence number must be greater; if both prefix match length and sequence number are the same, the hop count in the candidate entry must be less than the hop count field in the packet header.

With the above requirements, inter-cell routing is guaranteed loop free. At any given time, a node can use only one unique DART entry to forward a given packet (its best entry, by the above selection algorithm). Moreover, DART entries reflect the reverse paths of the beacons, which are loop-free since any node forwards a given beacon only once. The reverse paths of beacons thus form tree branches originating at the respective drum. The paths traversed by data packets are composed of branch segments from different trees. When a packet is forwarded toward a drum, the packet always travels upward toward the root along the branches of the corresponding tree, which must be loop free. Finally, when the prefix match length increases during the packet delivery, the packet has jumped to another tree since now the packet is forwarded

toward a different, lower level drum. Because the inter-cell forwarding algorithm forwards a packet from a node with some prefix match length only to a node with greater than or equal prefix match length, the packet cannot jump back to a tree that was previously traversed. Therefore, the entire traversed path of a packet is loop-free.

3.4.2. Route repair in inter-cell routing

In the inter-cell routing algorithm as described above in Section 3.4.1, a node forwarding a packet follows the reverse path of the drum beacons. However, the corresponding DART entry at that node may have already expired, or due to partitions in the network or the unreliable wireless medium, some beacons might not have reached their intended scope; in these cases, the node attempting to forward the packet might not have any useful entry in its DART. Furthermore, even if the forwarding node does have a relevant DART entry, transmission of the packet to the indicated next-hop node might fail, for example due to node mobility or failure; we assume that a failure in transmitting a packet to the next-hop node can be detected after a limited number of retransmission attempts, for example through link-layer feedback as provided in the IEEE 802.11 MAC protocol [14].

When a forwarding node has no relevant DART entry for some packet, or when transmission of the packet to the next-hop node fails (after a limited number of retransmission attempts), the forwarding node invokes on-demand (reactive) *local route repair* to find an alternate route to continue forwarding the packet. After buffering any packets that could not be sent due to the failure, the node locally broadcasts a hop-limited Masai Local Route Request packet containing the following information derived from the undelivered packet: (1) the current value of the prefix match length, (2) the sequence number for the beacon being followed, (3) the hop count of the beacon being followed, and (4) the hierarchical address of the unreachable final destination node. In some other protocols, such as AODV [29], intermediate forwarding nodes may initiate route discovery for route repair. However, our mechanism for on-demand local route repair has unique requirements, as our DART data structure was originally obtained from the proactive beacon packets, not from a reactive route discovery process.

A node receiving a Local Route Request searches its DART for the hierarchical address of the destination node. If the node finds a longer prefix match for the destination hierarchical address or if the node finds a prefix match of the same length but with a greater sequence number, or if the prefix match length and the sequence number are both the same but the hop count in the DART entry is less than that in the Request, the node returns a Masai Local Route Reply containing the information from the matching DART entry, back to the originator of the Request. Once the Reply is received by the requester, the previously buffered packets are routed using the reverse path followed by the Reply just received. A node receiving multiple Local Route Replys chooses the Reply with the longest prefix match. If two Replys have the same length prefix match, the node chooses the Reply with the greater sequence number or lower hop count.

If a node receiving a Local Route Request cannot reply and if the Request is not a duplicate of one received earlier, the node forwards the Request by locally rebroadcasting it. The Request forwarder also must make sure that the Request is still within the transmission hop limit and that the Request generally travels "downhill" toward the destination, given the following definition of node "altitude." The "altitude" of a node with respect to a destination is defined by a combination of the prefix match length, the sequence number of the beacon, and the hop count to the relevant drum node, in that order. The longer the prefix match length, or the higher the sequence number, or the smaller the hop count, the "lower" is the "altitude." Similar to Gradient Routing [30], the state in each node's DART generally forms a downward gradient toward the respective drum node. The Local Route Request, therefore, can be forwarded with limited "uphill" hops other than the general transmission hop limit. For example, we can allow the transmission hop limit to be 4 and the "uphill" limit to be 2. In this way, the request packet can be forwarded "downhill" or "level" up to 4 hops to find better routing information in a node's DART; it can be forwarded no more than 2 hops "uphill." The local exploration of the Request is more efficient as the transmissions are predominantly "downhill" as expected.

When forwarding the Local Route Reply, nodes update their DART state as if forwarding a beacon packet, such that subsequent data packets can be forwarded normally using their updated DART entry.

3.4.3. Reactive intra-cell routing

When an intermediate node receives a data packet for forwarding, the node checks if it is in the same fundamental cell as the destination, i.e., the node's hierarchical address matches the hierarchical address of the destination. If so, the packet has reached the fundamental cell of the destination and intra-cell routing is used to further forward the packet to the destination. Although any ad hoc network routing protocol can be used as the basis for intra-cell routing, we choose to use DSR [17,18], as it is a purely reactive protocol that has been shown to perform well [7]. DSR is a source routing protocol, with each packet containing a source route (although the explicit source route can be removed from most data packets [13]). The DSR protocol consists of two mechanisms: *Route Discovery* and *Route Maintenance*. To perform a Route Discovery for a destination node D, a source node S broadcasts a DSR Route Request packet that is flooded through the network in a controlled manner. This Request is answered by a DSR Route Reply either from node D or from some other node that knows a route to D in its Route Cache. To reduce the frequency and propagation of Route Requests, each node aggressively caches source routes that it learns or overhears.

In traditional DSR, the flooding of a Route Request might be carried throughout the network, thus making Route Discovery increasingly expensive with increasing network size. Since in our case of intra-cell routing, it is already known that the destination exists in this fundamental cell, Masai intra-cell Route Discovery is limited to within that fundamental cell. Since different fundamental cells may have different sizes, the Route Discovery range is based on the dynamic membership of the nodes instead of on a predefined hop count limit. Specifically, whenever a node receives a Masai intra-cell Route Request, it compares it's own hierarchical address with the Route Request initiator's hierarchical address and forwards the packet only if the two hierarchical addresses match. This technique is scalable, as the fundamental cell size does not increase with the network size.

An originator node A may have the wrong hierarchical address of the destination node B, as B might have changed it's cell membership recently and this change is not yet known to node A. To take advantage of the high probability of the destination still remaining in the vicinity of its previous fundamental cell, a hop count threshold is introduced in the intra-cell Route Discovery that allows the intra-cell Route Request to additionally be forwarded one or two hops beyond the fundamental cell. Each node forwarding the Route Request thus checks if its own hierarchical address differs from that of the originator of the Route Request and if so, increments the hop count field in the packet. If the hop count is less than a threshold, the Request is forwarded, and otherwise it is dropped. This hop count threshold creates a fuzzy boundary for forwarding the Route Request beyond the cell, allowing routing with less overhead.

4. Models and analysis

In this section, we use analysis, based on models from statistical physics, to assess the performance of the drum level selection protocol and the overhead resulting from periodic beacons in the Masai realization of the Safari architecture. We validate the predictions made by these models through simulations.

4.1. RSA: a model for drum formation

The *random sequential adsorption (RSA)* model [34] describes molecular adsorption processes, which exhibit strong similarities to the drum formation in the Masai realization of the Safari. It was first studied by Renyi [31] and has gained great popularity known as the *car parking problem*, in which cars of unit length arrive sequentially at random locations along a street; each car attempts to "park" at its current location, or leaves if it would overlap with the space already occupied by another parked car. To apply this model to the drum formation process in Masai, we make two simplifying assumptions in this analysis.

4.1.1. The instantaneous propagation assumption

We use an instantaneous propagation assumption in this analysis to ensures with high probability the sequential character of drum formation, similar to the car parking problem (RSA). This assumption states that

$$T_{\max} - T_{\min} \gg T$$

where T is the time for a packet to traverse D_1 hops. We assume that each node, after it powers on, waits for a random time, uniformly distributed between $T_{\min}$ and $T_{\max}$, to receive a beacon from a drum, before becoming a level-1 drum itself.

Consider two nodes A and B, within D_1 hops of each other, which each just powered on and are waiting to receive a beacon from any level-1 drum. Under the instantaneous propagation assumption, with high probability, the first beacon of a node that just became a drum reaches the other node before its timer expires.

Therefore, with high probability, level-1 drums form or "arrive" sequentially, rather than simultaneously. In particular, they are separated by at least D_1 hops without need to resolve conflicts. The argument extends easily to all levels of the drum hierarchy.

4.1.2. The poisson arrivals assumption

Since we consider in this section the performance of the drum level selection from a "cold start" of the entire ad hoc network, it is natural to assume that all nodes arrive (power on) at the same time, with locations according to a (homogeneous) Poisson point process in the plane. Extensions to three-dimensional space are immediate. As a consequence of this Poisson arrivals assumption, the number of nodes arriving in disjoint regions of the plane are statistically independent of each other. The overall number of nodes in the network under this model is a Poisson random variable. More importantly, given the number of nodes in a region, their locations are statistically independent and uniformly distributed. This justifies the use of the RSA model in our analysis, given the number of nodes.

As we will study mainly the mean behavior of the protocol, it is useful to introduce the *node density*, ρ, as the expected overall number N of nodes in the network divided by the overall area A of the network.

4.1.3. Drum formation conforms to the RSA model

Under the above two assumptions on propagation and location in our analysis, the drum level selection can be thought of as nodes attempting to "park a disc-shaped car" in the following sense. A node arrives at its actual (physical) location at the time when its waiting timer expires after powering on. Under the Poisson arrivals assumption and given the number of participating nodes, this location is uniformly distributed and independent of other node locations. If the node has not received a beacon from any level-1 drum, the node will increase its level from 0 to 1, becoming a drum. Under the instantaneous propagation approximation, this new drum is at least D_1 hops away from any existing drums, which can be interpreted as parking a disc of radius $D_1/2$ hops. Otherwise, if the node did receive a beacon from a drum before expiration of its timer, it leaves its level set to 0 (the node is not a drum), which can be interpreted as a failed car parking: the node "leaves" the drum competition.

The drum formation process will proceed as long as a disc of radius $D_1/2$ hops can be parked without collision (i.e., without overlapping with the space already occupied by another parked car). Indeed, since we measure distance here in terms of number of hops, the presence of a disc of radius D_1 hops from every existing level-1 drum implies the presence of a node that will become a drum itself upon expiration of its timer. The formation stops at the so-called "jamming limit" when no further disc can be parked without colliding with existing discs.

Although equating in this analysis geometrical distance with hop distance introduces a distortion, this distortion is homogeneous over space, with high probability, if the density ρ is sufficiently large, and amounts to a change of units. For clarity, we denote by ρ_0 the node density in the hop metric sense. Its value depends on the transmission range of each radio and the spatial density of nodes. From simulations, we estimate a value of $\rho_0 \simeq 1.4$ for node density ρ of 10 nodes per radio range and for a radio range of 250 m.

The estimate of ρ_0 is obtained by simulating a packet broadcast initiated by a node that is situated near the center of the simulation area in a stationary network and recording the hop count at which this packet is received at all the other nodes in the network. The number of nodes that receive the packet within H hops is $\rho_0 \pi H^2$. Hence ρ_0 is obtained by dividing this number by πH^2.

To state the analytical result from applying the RSA model, we denote by $N_0 = N$ the total number of nodes in the network, and denote by N_1 the average number of level-1 drums that form. Then, using the fact that the parking density at the jamming limit [34,31] is 54.7%, the ratio of level-1 drums to the total number of nodes can be estimated as

$$\frac{N_1}{N_0} \simeq \frac{0.547}{\frac{\pi}{4} D_1^2 \rho_0}. \tag{2}$$

The graph in Fig. 2 was obtained by simulating the drum protocol in *ns*-2 for 100 s with 500 mobile

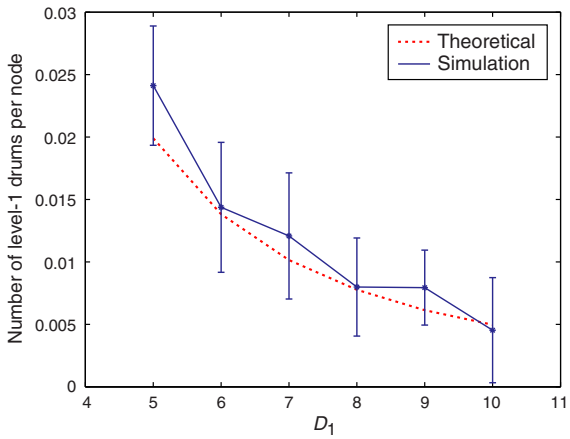


Fig. 2. Number of level-1 drums per node (*ns*-2 simulation).

nodes. Node speeds in these simulations are uniformly distributed between 5 and 15 m/s. Each simulation was run 10 times, and the error bars indicate the standard deviation from the mean. The mobility model here is a modified billiard ball model with random reflection angle upon hitting simulation area edges. The advantage of this model here is that it results in a uniform distribution of the nodes even during mobility, providing a closer match to the RSA model which assumes uniform arrival locations of cars. It is well known that in the Random Waypoint model [7], the nodes tend to show a larger concentration at the center; however, this bias towards the center is less pronounced at the beginning when starting the nodes in a uniform distribution.

If there is spatial non-uniformity in the node distribution, the value of ρ_0 will also vary spatially. Its value will be low at low node density regions and high at high node density regions. This variation may lead to deviation from what is predicted by Eq. (2) above. However the inverse square relation on D_1 in Eq. (2) will still hold.

The graph shows the number of level-1 drums per node at time 100 s, at which time the drum formation process is complete. The timer expires uniformly in [0,150 s]. Hence, the instantaneous propagation approximation holds. The graph also shows the number of level-1 drums based on the RSA model analysis. These results demonstrate that the RSA model successfully predicts the number of drums that have formed at the convergence of the drum level selection protocol.

For higher level drums, the car parking problem becomes a constrained parking problem, as the parking "substrate" consists now of lower level drums. Being less dense and quite discrete, this set of drum nodes provides a less ideal approximation of a continuum, and the appropriate model is the RSA-random sites (RSA-RS) model [16]. The model effectively reduces the packing density somewhat and results in the following generalization of Eq. (2):

$$\frac{N_i}{N_{i+1}} = \alpha_i \left(\frac{D_{i+1}}{D_i}\right)^2. \tag{3}$$

Here, N_i is the average number of level-i drums that form, and α_i is the ratio of the average packing densities of the RSA-RS process associated with level-i drums and the level-$(i+1)$ drums. As D_{i+1}/D_i is increased, the substrate becomes a closer approximation of a continuum, and α_i tends to unity. The RSA-RS model can also be used to deal with networks that are less dense than assumed in Eq. (2).

4.2. Collision model for drum retirement

When two level-n drums move to within $h \times D_n$ hops of each other, one of them will retire. Because the radii of the discs are the half of the distance between the two drums, this is analogous to a collision of two fictitious discs of radii $h \times D_n/2$ hops. Again, results from statistical physics of molecular collisions allow to compute the frequency of drum retirements [32] under the modified billiard ball mobility model above, corresponding to the motion of gas molecules. Adapting the theory for two-dimensional discs, we obtain the drum retirement rate λ, the number of level-n drum retirements per s in the network, to be

$$\lambda \simeq \frac{A D_n v_{\text{average}} \rho_n^2}{\sqrt{2}} \tag{4}$$

where v_{average} is the average velocity of the nodes, A is the area covered by the network, and ρ_n is the spatial density of level-n drums in terms of hop distance. To obtain a simpler functional form, we substitute $\rho_n = N_n/A$. However, from the RSA model described previously, N_n is proportional to N_0/D_n^2. In summary, we find that Eq. (4) takes the functional form $\lambda \simeq (N^2/AD_n^3)$. Thus, the stability of the drums increases with the level of the drums.

The drum formation rate at equilibrium must be equal to the retirement rate and is therefore also given by Eq. (4), if we assume a fairly stable population of drums.

4.3. Overhead characterization

This section characterizes some of the overhead caused by the beacons, leveraging the models and analysis described above.

4.3.1. Drum flooding overhead

From Eq. (3), for every level-$(i+1)$ drum, there are $\alpha_i(D_{i+1}/D_i)^2$ level-i drums, where α_i is a proportionality factor that is close to unity. Consider a network with l levels of drum hierarchy (l is $O(\log N_0)$). Let the level-i drum emit beacon packets once every T_i s. A level-i drum beacon reaches all nodes in the next higher level (level-$(i+1)$) cell. So a node will receive drum packets from all level-i drums that are within the same level-$(i+1)$ cell that contains this node ($\alpha_i(D_{i+1}/D_i)^2$ of them), for i running from 1 to $l-1$ and from the highest level drum.

Therefore, overhead due to drum beacon floods, X_{drum}, defined by the number of beacon packets forwarded per s per node, is

$$X_{\text{drum}} \approx \alpha_1 \frac{D_2^2}{D_1^2}\frac{1}{T_1} + \alpha_2 \frac{D_3^2}{D_2^2}\frac{1}{T_2} + \cdots + \alpha_{l-1}\frac{D_l^2}{D_{l-1}^2} \times \frac{1}{T_{l-1}} + \frac{1}{T_l}. \quad (5)$$

The actual overhead due to drum beacon floods is a little greater than that shown in Eq. (5), because beacon floods propagate a minimum number of hops (D_i). Therefore, a level-i drum's beacon packets reach some nodes of a neighboring level-$(i+1)$ cell if the drum is at the border of the level-$(i+1)$ cell. However, the error in Eq. (5) is negligible, since such overhearing nodes are at the border of the cell and hence are far fewer than interior nodes of a cell. Moreover such a node will receive such a "leaked" beacon packet only once for every, order of, $(D_{i+1}/D_i)^2$ broadcasts from its own level-$(i+1)$ cell. The D_i's and T_i's are geometrically increasing, as shown by Eq. (1). Hence, Eq. (5) implies that $X_{\text{drum}} = \Theta(1)$.

4.3.2. Frequency of cell changes for a node

A node's hierarchical address could change if the node moves into a new cell. Since the average interspacing (in hops) between two level-$(i+1)$ drums is proportional to D_{i+1}, if the average movement speed of the level-i drum node is v, it will cross the boundary of its level-$(i+1)$ cell in a time proportional to D_{i+1}/v. In particular, the frequency with which a node moves to an adjacent fundamental cell is proportional to v/D_1.

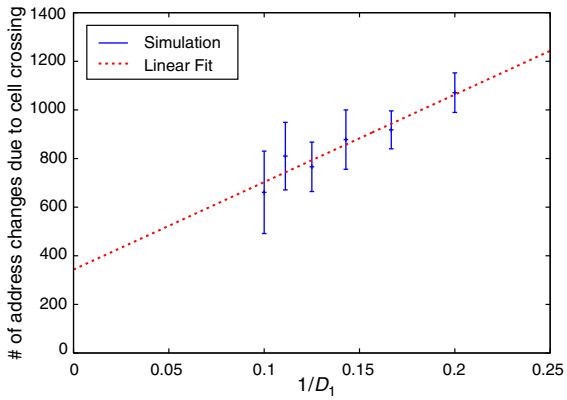


Fig. 3. Level-1 cell crossing frequency follows a $\frac{1}{D_1}$ law (*ns*-2 simulation).

Fig. 3 shows the result of an *ns*-2 simulation under the modified billiard ball mobility model described above. For each point, 10 simulations are run, and the variance is shown. Each simulation is run for 100 simulated seconds, and the data during the first 25 s was discarded to remove initial transient behavior. The x-axis is the inverse of D_1, and the y-axis is the total number over all 500 nodes of changes in the hierarchical address for a node due to level-1 cell crossings. The results show that the number of hierarchical address changes due to level-1 cell crossings is proportional to $1/D_1$.

5. Simulation evaluation

In addition to the analysis presented in Section 4, we have also evaluated the performance and scalability of the routing protocol in the Masai realization of the Safari architecture, using detailed *ns*-2 network simulations. We compare the performance of the Masai routing protocol with another landmark-based protocol, L+ [9], and with DSR [17,18], a well-known purely reactive routing protocol for ad hoc networks.

We used version 2.1b8a of *ns*-2, with the Monarch Project wireless and mobile extensions for *ns*-2 [7]. These extensions provide detailed modeling of an IEEE 802.11-based network with a wireless physical rate of 11 Mbps and a nominal wireless transmission range of 250 m. We implemented our Masai design by extending the existing DSR code distributed with *ns*-2, since we use DSR for routing within a fundamental Masai cell.

We set the beacon broadcast limit to 3 wireless hops ($D_1 = 3$), thus allowing the fundamental cell to be up to 6 wireless hops in diameter; DSR has

been shown to perform well, without significant overhead, in networks of this size [7]. Level-1 drums originate beacon packets every 1 s ($T_1 = 1$). We set γ in Eq. (1) (Section 3.1) to be 2 and set β in Eq. (1) also to be 2. For the *local route repair* mechanism in Masai (as in Section 3.4.2), we used 4 hops as the transmission limit and 2 hops as the "uphill" limit.

We simulated a large number of network topologies, with sizes ranging from 50 nodes to 1500 nodes and locations randomly distributed in a two-dimensional area. However, the x- and y-dimensions of the network area were modified so that the average network density was kept constant at the equivalent of 50 nodes in a 670 m × 670 m area; the average number of nodes per nominal wireless transmission area is thus approximately 20. This node density is the same as that used by Broch et al. [7] and has been used in many other ad hoc network simulations. We have found that this *average* density tends to avoid temporary network partitions when nodes randomly move around. Due to the very large memory consumption of *ns*-2 with large numbers of nodes, we had to limit our simulations to a maximum of 1500 network nodes. Nevertheless, these results, together with our analysis in Section 4, demonstrate the scaling potential and efficiency of routing in the Masai realization of Safari.

Nodes in these simulations move according to the Random Waypoint mobility model [7], with a maximum speed of 10 m/s (average of 5 m/s) and a Pause Time of 0. However, since as Yoon et al. [38] point out, the original Random Waypoint model suffers from a decaying average node speed over the life of the simulation, as suggested by them, we added a minimum speed limit of 1 m/s in our simulations to avoid this problem. Each simulation runs for 900 simulated seconds.

The network workload in these simulations consists of constant bit rate (CBR) flows, with each flow consisting of a randomly chosen source and destination node. Each flow lasts 90 s and generates 64-byte packets at constant rate of 4 packets/second. The first 350 s in each simulation run are used to observe the performance of the cell organization and drum selection; at time 350 s, data flows then begin arriving according to a Poisson distribution. This traffic pattern is more challenging to the routing protocol than the typical continuous long lifetime CBR flows, since it requires using new routes to many more unique destinations.

Each data point in our graphs for this performance evaluation represents the average of 16 individual simulations, created from the combination of 8 different randomly generated mobility patterns and 2 different randomly generated data traffic patterns. The error bars in these graphs are calculated as the sample standard deviation of the 16 runs for each data point; to more clearly show the error bars, some data points in some graphs have been shifted slightly along the x-axis for different curves.

Table 2 shows the header fields and sizes used in each type of packet used in the Masai realization of the Safari architecture. Each row of the table represents one type of packet and shows the size of the header fields used by that packet type, and each column shows a possible header field and the corresponding size (in bytes) of that field for that packet type (header fields not used in a given packet type are indicated as "N/A"). The total size of the Masai header for each packet type is thus the sum of the values in that corresponding row. Other parts of each packet, such as the standard MAC and IP headers, and the payload for DATA packets, are not shown in the table but are counted in the total packet size in our simulations. Since we base our intra-cell routing protocol on DSR, we use the same packet formats as used in the standard *ns*-2 implementation of DSR for all packets in the intra-cell phase in Masai.

Table 2
Header field sizes for each Masai packet type

Packet type	Header field and size (bytes)					
	Control	Hierarchical address	Beacon sequence	Hop count	Matched prefix length	Source route
BEACON	1	4 × level	4	1	N/A	N/A
LOCAL ROUTE REQUEST	1	4 × level	4	1	1	4 × length
LOCAL ROUTE REPLY	1	4 × level	4	1	1	4 × length
DATA in inter-cell phase	1	4 × level	4	1	1	N/A
DATA in intra-cell phase	4	N/A	N/A	N/A	N/A	4 × length

5.1. Routing scalability

An important goal of Safari is to provide routing in large ad hoc networks. We show that the design meets this goal by evaluating the packet delivery ratio (PDR), routing overhead, packet delivery latency, and routing path lengths used across different network sizes, up to 1500 nodes, in the Masai realization of the Safari architecture.

PDR is defined as the fraction of application data packets originated that are successfully received by the application layer at the respective destination node. Routing overhead per node is defined as the average network bandwidth consumed per node over all non-data packet transmissions. Overhead packets include beacon packets, Local Route Request and Local Route Reply packets for route repair, and all DSR routing packets. Every routing overhead packet contributes to this overhead each time it is transmitted (originated or forwarded). Furthermore, as data traffic does not start for the first 350 s, overhead bandwidth per node is calculated as an average over the period between time 350 and time 900 s (the end of the simulation). For the results shown in this section, *all* nodes in the network are mobile, with a speed between 1 m/s and 10 m/s, and the traffic in the network for each simulation is 100 CBR flows.

We compared Masai's performance against L+ and against DSR. For DSR, we used the version distributed with *ns*-2; we made only minor changes to it to make it support up to 32 hops in its source routing packet header, and likewise expanded its Route Discovery hop limit. For L+, we used the *ns*-2 code provided by the authors of L+, and we used its published default parameters [9]. Since we do not consider Masai's address lookup service in this paper, we similarly disabled the address update/query services of L+ and made the current hierarchical address of a packet's destination node available to the source node at no cost, in order to ensure a fair comparison between Masai and L+. Due to limitations in our experimental platform, we were unable to simulate DSR in networks of 1500 nodes and limited our DSR simulations to 1000 nodes.

Fig. 4 shows the comparative PDR performance of Masai, L+, and DSR. Masai delivers close to 100% of all data packets at all network sizes, despite the continuous mobility of all nodes in the network. At a network size of 1500 nodes, Masai achieves an average PDR of about 99.6%, with a very small standard deviation, whereas DSR and L+ achieved significantly lower PDR performance in very large networks. DSR slightly outperforms L+ in small networks (50 nodes), since it is entirely reactive whereas L+ is entirely proactive. Masai, with both proactive and reactive mechanisms, outperforms both L+ and DSR in these small networks. With increasing network size, the PDR of DSR drops significantly, due to the increasing length of routes that must be maintained.

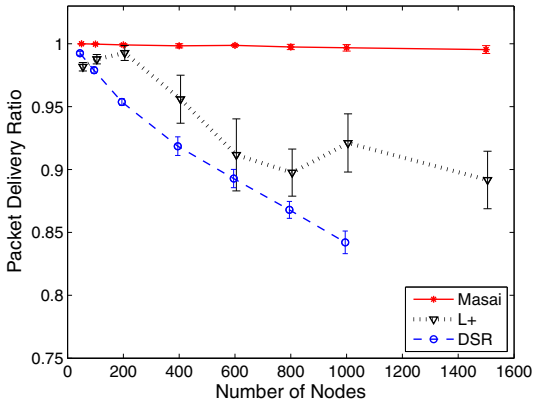


Fig. 4. Packet delivery ratio vs. network size, with all nodes mobile.

The PDR of L+ shows fluctuations as the network size increases. We are not sure of the cause of this fluctuation, but we conjecture that it is due to the interaction between the protocol's purely proactive routing and, as the network size increases, the discontinuous increases in the number of landmarks at each level and the total number of hierarchy levels. With increasing network size, the average size of each cell in L+ becomes larger and larger until a new landmark is created and if necessary a new level of the hierarchy forms. The landmarks become further and further apart in hop count (path length). As a result, due to the mobility of nodes in the network, packets being forwarded may more often encounter broken links on the path to a remote landmark; the likelihood of such a broken link increases as the path length increases. As the network size passes the point at which a new landmark or level of the hierarchy is created, the average size of each cell is reduced, as landmarks are now on average closer together in hop count; the path to the remote landmarks become shorter, and the chances that a packet being forward encounters a broken link on the way to the next landmark is reduced. Typical runs of L+ create a hierarchy of

2 levels for 50 and 100 nodes; 3 levels for 200, 400, and 600 nodes; and 4 levels for 800, 1000, and 1500 nodes. For 800- and 1000-node networks with L+, there is typically only one level-4 landmark, but 800-node networks have 4 level-3 landmarks while 1000-node networks have 7 level-3 landmarks. This difference in number of level-3 landmarks gives 1000-node networks a much smaller average size of the level-3 cells, accounting for the rise in PDR for L+ at a network size of 1000 nodes. A similar effect accounts for the rise at 200 nodes.

Masai experiences similar changes in its hierarchy with changing network size. For example, the average size of level-2 cells in Masai ranged from about 150 to 300 nodes, but the reactive local repair mechanism in Masai allows it to overcome any broken links (e.g., due to mobility) in following the reverse path of beacons from remote drums. The PDR of Masai thus remains almost constant as the network size increases.

To better understand the reasons behind each of the few dropped data packets experienced with Masai, we examined the *ns*-2 trace files to determine what caused each data packet loss. Fig. 5 shows the percentage of packet losses from each possible cause in the 1500-node networks using Masai. Of the total of 2434 lost data packets across the 16 simulation runs, more than 86% are due to failure in inter-cell routing, around 5% are due to failure in intra-cell routing, and around 7% of the lost packets occurred simply because the packets were still in transit when the simulation time ended. The remaining 35 dropped data packets (around 1.4%) were due to various reasons such as overflowed network interface queues.

Fig. 6 shows the routing overhead per node for Masai, L+, and DSR as the network size increases, with all nodes mobile as described above. Both Masai and L+ have a convex shape to their overhead bandwidth curves, becoming flatter as the network size increases, implying that their overhead scales logarithmically. To fully confirm our theoretical prediction for Masai's overhead from Section 4, simulations of larger networks must be done. However, this is currently limited by the inability of *ns*-2 to scale to such very large networks. L+ shows very small error bars for its routing overhead, which is expected for a purely proactive routing protocol. DSR shows very low overhead for small network sizes, but its overhead increases sharply as the network size increases, reaching approximately the same level as for Masai and L+ at 1000 nodes. Masai's overhead is higher than DSR's for networks less than 1000 nodes, but this overhead should be considered relative to Masai's significantly higher PDR (Fig. 4). Masai shows significantly lower overhead than does L+, due to factors such as the fact that individual nodes (at level-0) do not send beacons in Masai but do periodically advertise themselves in L+. Overall, since Masai delivers a much higher fraction of the data packets than does L+ or DSR, its advantage in efficiency is much greater than what is shown in this figure.

Fig. 7 shows the average packet delivery latency for the three protocols as the network size increases. In each of the protocols, average latency increases as the network size grows larger. The average delivery latency of Masai is higher than that of the other two protocols, though, due to a small number of delivered packets that required significantly longer

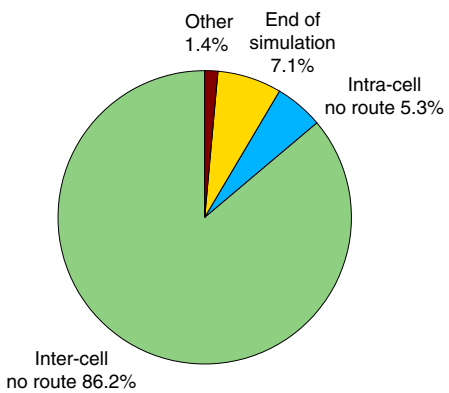


Fig. 5. Percentage of packet loss for each cause for Masai in 1500-node networks, with all nodes mobile.

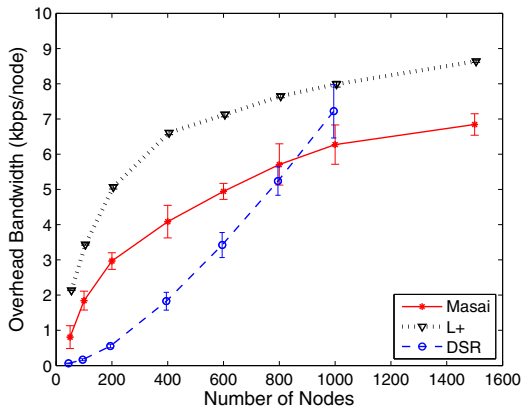


Fig. 6. Overhead bandwidth per node vs. network size, with all nodes mobile.

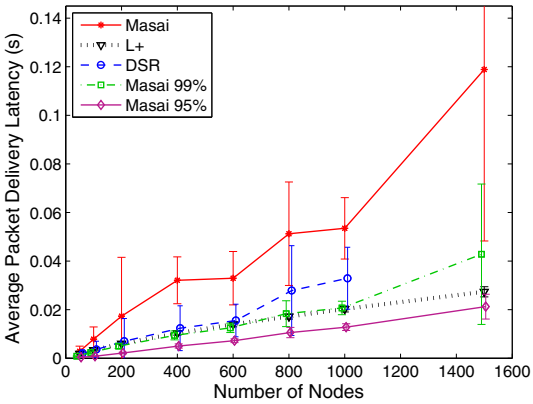


Fig. 7. Average packet delivery latency vs. network size, with all nodes mobile.

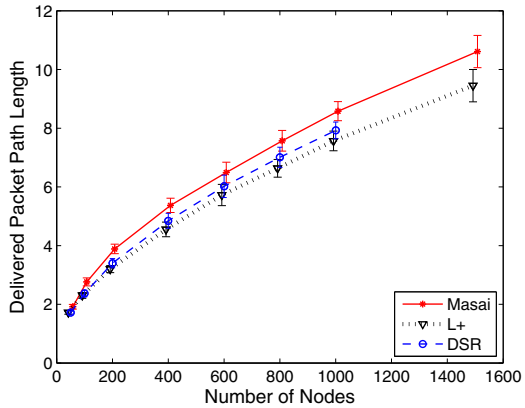


Fig. 8. Number of hops vs. network size, with all nodes mobile.

to be delivered. The longer delivery latency for these packets is caused by a combination of the time taken by the local route repair mechanism in Masai (Section 3.4.2) and the increased overall route length that may be created by this repair until the next BEACON packet passes through that part of the network. In contrast, L+ and DSR have no local route repair mechanism such as this and thus instead must drop packets that could have benefited from local repair; this difference is demonstrated by the substantially higher PDR for Masai than for L+ or DSR, as shown in Fig. 4. To further illustrate this point, we also show in Fig. 7 the average delivery latency of the 99% and 95% of delivered packets in Masai with the lowest delivery latency. When just 1% of the delivered packets are excluded, the average latency improves to roughly equal to that of L+ or DSR, and when 5% are excluded, Masai's average latency is clearly below the other two protocols, even though the difference in Masai's PDR over L+ or DSR (Fig. 4) is greater than 5%.

Another observation from Fig. 7 is that Masai and DSR have larger variance in their average delivery latency than does L+. This increased variance is due to the fact that L+ uses a purely proactive routing mechanism, whereas DSR is entirely reactive and Masai is a hybrid of reactive and proactive mechanisms (Masai intra-cell routing is entirely reactive, as is its local route repair in inter-cell routing). By its nature, any reactive routing mechanism may sometimes cause a packet to be delayed while the protocol searches the network to discover a new route, adding to delivery latency variation since only some packets require a new route discovery. Proactive routing, on the other hand, does not experience such delays but instead pays the cost of ongoing background overhead for exchanging routing information in order to attempt to always keep all routes up-to-date.

Fig. 8 shows the average path length (number of hops) used by delivered packets, with increasing network size, for Masai, L+, and DSR. Masai shows a small disadvantage compared to L+ and DSR; for example, in networks of 1000 nodes, Masai's average path length is about 1 hop longer than for L+ and one half hop longer than for DSR. This difference is primarily caused by the fact that Masai delivers more of the data packets than does L+ or DSR, as shown in Fig. 4. Whereas L+ or DSR must discard a data packet if it encounters a broken link along the packet's route (DSR's "packet salvaging" mechanism is able to avoid discarding some of these packets [17,18]), Masai uses its local route repair mechanism to find a new route in such circumstances and is thus able to successfully deliver the packet. However, the resulting total route used by the packet may be longer than optimal, as the broken link may be replaced by more than a single new hop; this slightly longer route persists until the next periodic beacon packet from the relevant drum reestablishes an optimal path back toward toward that drum.

5.2. Effect of mobility

The objective of Masai is to provide scalable routing for a large-scale ad hoc network environment. Whereas all results presented above are for *all* nodes being mobile, it is unlikely that all nodes in a real network will be mobile all the time. We thus study here scenarios using the Masai realization of Safari, with different fractions of the nodes

being mobile, ranging from all mobile to all stationary. Whether a node is mobile or stationary is not known to Masai in our simulations; if, for example, the stationary nodes were known, choosing them as drums in Masai would provide better performance due to fewer hierarchical address changes, but we do not explore such optimizations here.

In these simulations, we fixed the number of nodes in the network at 1000 nodes and the number of CBR flows at 100. We varied the percentage of mobile nodes from 0% nodes being mobile (all nodes are stationary) to 25%, 50%, 75%, and 100% nodes being mobile (all nodes are mobile). As above, mobile nodes move according to the Random Waypoint mobility model [7] with a speed between 1 m/s and 10 m/s.

Fig. 9 shows the change in PDR with the number of mobile nodes varying from 0% to 100%. Although there is a slight decrease in PDR with increasing mobility (the *y*-axis of the graph is magnified, ranging from 0.99 to 1.0 PDR), the PDR is around 99.7% even for 100% mobile nodes, showing that Masai reacts very well to mobility.

Fig. 10 shows the changes in routing overhead with this increase in percentage of mobile nodes, broken down by the overhead caused by Masai's proactive mechanism (beaconing) and reactive mechanisms (local route repair and intra-cell routing). The proactive overhead in Masai increases slowly as the mobility degree increases, due to changes to which nodes are drums. In particular, when an existing drum node moves away from the other nodes in its cell, another node there becomes a drum, while the first drum node may continue also as a drum in its new location or may take some time before deciding that it no longer needs to be a drum. As expected, reactive overhead grows as the percentage of mobile nodes increases.

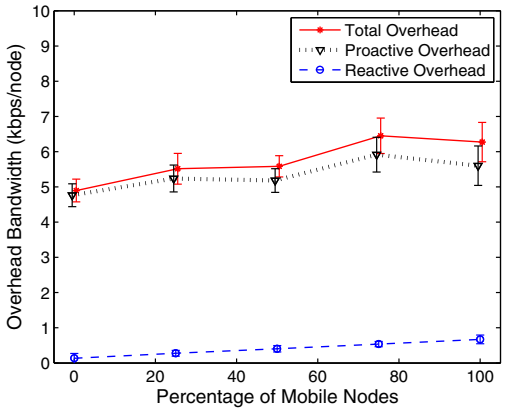


Fig. 10. Overhead bandwidth per node vs. percentage of mobile nodes in 1000-node networks.

Fig. 11 shows the changes in average packet delivery latency with the increase in percentage of mobile nodes; as in Fig. 7, we also show here the average delivery latency for the 99% and 95% of packets with the lowest latency. As the number of mobile nodes increases, the delay increases slightly, and the gap between the total average latency and the fastest 99% average latency grows larger. This widening gap suggests that although the latency every packet experiences grows, the largest impact is on the packets that require the most effort to deliver, experiencing perhaps more than one local route repair for a packet before delivery.

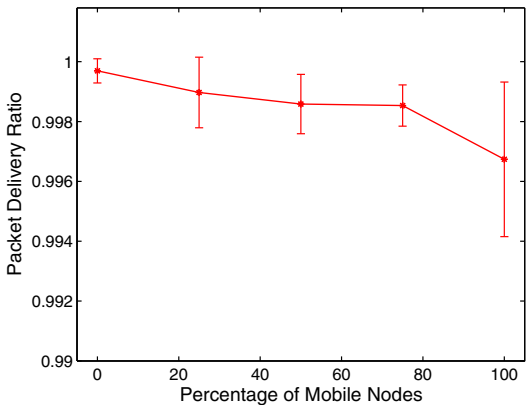


Fig. 9. Packet delivery ratio vs. percentage of mobile nodes in 1000-node networks (the *y*-axis ranges only from 0.99 to 1.0 PDR).

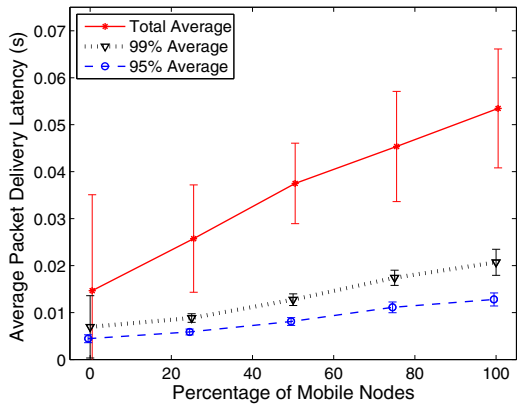


Fig. 11. Average packet delivery latency vs. percentage of mobile nodes in 1000-node networks.

5.3. Effect of traffic load

We now show how routing performance in the Masai realization of Safari varies with different levels of traffic load. The number of nodes here is constant at 1000, and all nodes are mobile with a speed between 1 m/s and 10 m/s. We vary the traffic load with 10 flows, 100 flows, 200 flows, and 300 flows.

Figs. 12 and 13, respectively, show the PDR and routing overhead at different levels of traffic load. The PDR decreases very slightly as the network becomes congested (the y-axis in Fig. 12 ranges only from 0.99 to 1.0 PDR). The total overhead also increases with the increase of traffic load from 100 to 300 CBR flows. This increase in overhead is because, as the number of destinations grow, more uses of DSR route discovery are needed to find intra-cell routes the final destinations. On the other hand, the proactive (beacon) overhead remains nearly constant.

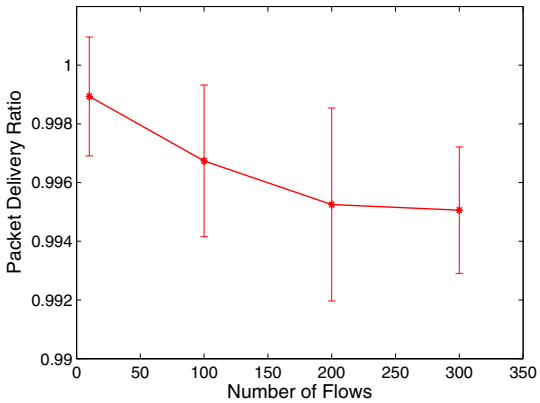


Fig. 12. Packet delivery ratio vs. traffic load in 1000-node networks, with all nodes mobile (the y-axis ranges only from 0.99 to 1.0 PDR).

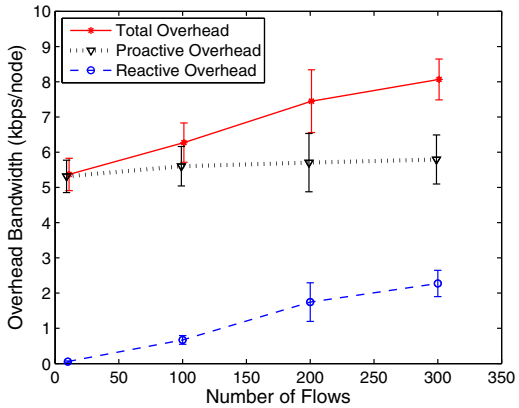


Fig. 13. Overhead bandwidth per node vs. traffic load in 1000-node networks, with all nodes mobile.

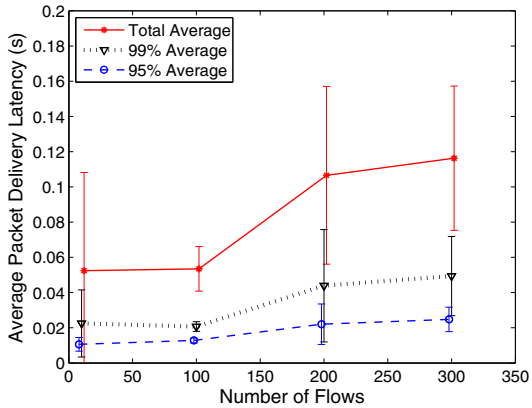


Fig. 14. Average packet delivery latency vs. traffic load in 1000-node networks, with all nodes mobile.

Fig. 14 shows the average packet delivery latency at different levels of traffic load. As the traffic load increases, the delay also increases slightly. The gap between the total average latency and the 99% average latency also increases as more flows are added to the network, due to an increase in number of packets affected by the need for local route repair.

5.4. Network bootstrapping

Finally, we evaluated the behavior of the Masai realization of Safari during network bootstrapping, when all nodes in the network power up and initialize at the same time. This study demonstrates the "worst case" behavior of node initialization using the Masai beaconing protocol (Section 3.1), drum level selection algorithm (Section 3.2), and membership algorithm (Section 3.3), as in most real networks, the individual nodes typically do not all power on simultaneously.

As described previously, when a node powers up, it initially waits for a period of time during which it forwards beacons from other nodes (drums) and attempts to choose some existing level-1 drum as its parent; the node waits until the expiration of this period before deciding, if necessary, to increase its own level and become a drum itself. Each node randomly selects this waiting period between 1 to 100 s in our simulations. This randomized waiting period avoids all of the nodes increasing their own level and becoming a drum at the same time.

Figs. 15 and 16 characterize the performance of Masai during network bootstrapping in one of our 1500-node simulations with 100 data flows; all nodes are mobile with a speed between 1 m/s and 10 m/s. Fig. 15 shows the changes in number of drums at different levels of the hierarchy over the duration of the simulation run. At time 100 s, the network has stabilized with a single level-4 drum, which remains the case throughout the remainder of the simulation. As all nodes are mobile, the number of drums at lower levels fluctuates somewhat, and the lower the drum level, the more fluctuation there is. Fig. 16 shows the changes in network overhead over the duration of the simulation. For the first data point (20 s), the overhead is initially high, primarily because, as described in Section 3.3, the nodes initially forward all beacons, until the cell structure of the network begins to form and there are at least two level-2 drums; after this point, the level-1 beacon scope will be confined by the cell structure. After the beacon overhead stabilizes, there is still some fluctuation in overhead, as different nodes become drums or cease being drums due to node mobility. After time 350 s, when the CBR flows begin, reactive overhead begins to contribute to the total overhead but remains low throughout.

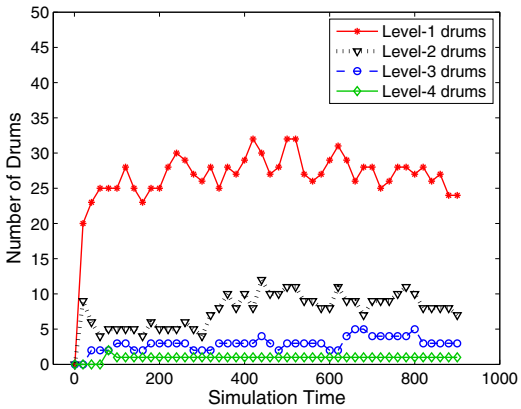


Fig. 15. Number of drums vs. simulation time in a 1500-node network, with all nodes mobile.

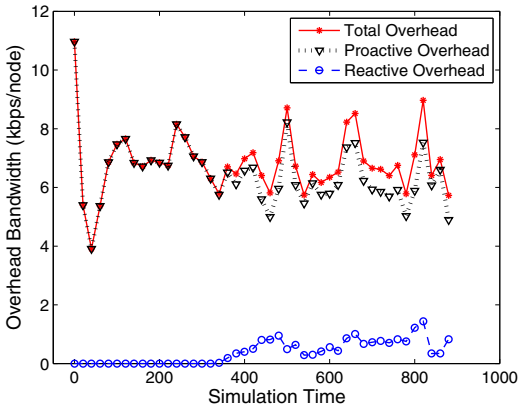


Fig. 16. Overhead bandwidth per node vs. simulation time in a 1500-node network, with all nodes mobile.

6. Related work

In this section, we discuss related work in scalable routing for ad hoc networks and how the Safari architecture and the Masai realization of Safari differ from existing approaches.

Ad hoc network routing protocols can generally be classified as either *proactive* (periodic) or *reactive* (on-demand). Proactive protocols (e.g., [28,15,25]) attempt to maintain up-to-date routes to all possible destinations at all times, whereas reactive protocols (e.g., [17,29]) attempt to discover or maintain routes only when needed to destinations for current communication. Reactive routing protocols have been shown to have generally lower overhead than proactive protocols, and they can react much more quickly as routes in the network change. However, for very large networks or very high rates of mobility, the overhead of current reactive protocols can grow quickly.

A number of approaches to scalable ad hoc network routing have been proposed. Geographical routing techniques (e.g., [6,19,20,22,3]) allow routing with state proportional only to the number of neighbors at each node, but they require GPS or other location techniques. Moreover, a source node must know the location of the destination before sending packets, thus requiring a location distribution and maintenance service. DREAM [3] and LAR [20], respectively, employ proactive or reactive flooding of the network and hence do not scale with the size of the network. GLS [21] is a scalable location service for geographical routing in ad hoc networks. However, even if the location service for geographical routing can be made scalable, GPS devices can be expensive and consume power, and do not function indoors, limiting the application of geographical routing techniques. Although Safari may be less scalable than geographical routing, Safari provides a practical, self-organizing hierarchy and is not dependent on GPS or other specialized devices. Also, geographic routing protocols generally

drop in performance in the presence of voids, as the protocol must backtrack to find a suitable next-hop node closer to the destination [19]; since Safari routing, instead, is based on following the reverse path of beacon packets, as long as the wireless links are bi-directional, such voids are not a problem for Safari.

Clustering techniques (e.g., [1,12,24,23,8,33,2,39]) can increase scalability, but existing active clustering mechanisms require periodic refreshing of neighborhood information and introduce significant maintenance overhead due to global query flooding. Another class of protocols use the idea of routing via dedicated fixed anchors (e.g., [4,5]), but such techniques depend on deployment of fixed anchor nodes. Gao et al. [11] propose a randomized kinetic clustering algorithm to create a set of clusters in a set of moving nodes; they also present a detailed evaluation of the properties of such clustering. Although this work can be used to create a collaborative and hierarchical ad hoc network, it is not clear how this algorithm would be implemented in a real network and how it would perform in such a real network. For example, the algorithm presented does not consider the inherent unreliability of wireless networks.

Techniques based on *landmark routing*, first proposed by Tsuchia [36,35,37], have also been proposed for scalable routing in ad hoc networks. Similar to our drum hierarchy in Safari, landmark nodes self-organize themselves into a hierarchy, such that landmarks at a given level in the hierarchy are an approximately equal number of network hops apart. The address of a node consists of the sequence of identifiers of the nearest landmarks, from highest to lowest level. During routing, a node extracts from the destination address the highest level landmark identifier that differs from its own node address, and forwards the packet towards the landmark with that identifier. The mapping from node identifiers to their current address is maintained in a distributed fashion. Landmark routing achieves scalability by dramatically reducing the size of per-node routing tables at the expense of somewhat longer routes. The original landmark scheme, designed for large wired networks such as the internet, had only routers as landmarks. End nodes did not participate in the hierarchy. LANMAR [27] attempts to scale mobile ad hoc networks by combining ideas from landmark routing and from Fisheye State Routing [26]. It specifically targets, however, ad hoc networks consisting of groups of nodes related in functionality and mobility.

The previous routing protocol that is most similar to Masai is L+ [9]. L+ modifies the lookup service of landmark routing to make it more scalable and modifies the routing to better handle mobile nodes. Although L+ has a number of similarities to our Masai realization of the Safari architecture, the fundamental differences between the two lie in how they perform routing. L+ uses a purely proactive approach to routing and is based on DSDV. Masai, on the other hand, employs a hybrid of proactive and reactive routing approaches. L+ keeps a list of routes to any destination and switches to the next route when the current best route breaks. It also needs to trigger a distance vector update whenever there is any change in connectivity due to mobility or channel loss. As the proactive part of routing, Masai uses reverse beacon paths, avoiding all per-destination overhead. When such a reverse route breaks, we perform on-demand local route repair, thereby switching to reactive routing to find a new route and repair the routing state. In addition, unlike L+, the hierarchy in Masai does not extend down to the lowest level but stops at fundamental cells. Within each fundamental cell, Masai uses a purely reactive protocol, thus reducing overhead and improving scalability. In summary, L+ inherits many of the problems associated with proactive ad hoc network routing protocols, which Masai avoids. With increasing mobility, the frequency of the proactive updates in L+ (or any proactive protocol) must increase proportionally, significantly increasing its overhead, a problem avoided in Masai; between these proactive updates, Masai can repair routes using reactive local route repair.

7. Conclusions

In this paper, we have given an overview of the Safari architecture for scalable routing in ad hoc networks. We have also presented the design and evaluation of a specific realization of the Safari architecture, which we call *Masai*. We focus in this work on the scalability of learning and maintaining the routing state necessary for a large ad hoc network. Masai includes a probabilistic, self-organizing network hierarchy formation protocol, complemented with a new hybrid routing protocol that uses this hierarchy. This hybrid routing protocol consists of both proactive and reactive components,

helping the routing to scale to a much larger numbers of nodes than previous ad hoc network routing protocols. Nodes in the Safari architecture are given hierarchical addresses, and each node's unique node identifier is mapped to its address using a distributed hash table (DHT) that leverages the hierarchical network structure. We have evaluated the Masai realization of the Safari architecture through analysis and simulations, under increasing network size, increasing fraction of mobile nodes, and increasing offered traffic load. Our simulation results demonstrate that the protocol is significantly more scalable than existing protocols. Compared to both the DSR and the L+ routing protocols, in particular, Safari has much higher packet delivery ratio (PDR) and lower overhead, successfully supporting routing in much larger ad hoc networks.

Acknowledgements

This work was supported in part by NSF under Grants CNS-0520280, CNS-0435425, CNS-0338856, and CNS-0325971; and by a gift from Schlumberger. The views and conclusions contained here are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, Schlumberger, Rice University, or the US Government or any of its agencies.

References

[1] Suman Banerjee, Samir Khuller, A clustering scheme for hierarchical control in multi-hop wireless networks, in: Proceedings of INFOCOM 2001, April 2001.

[2] Stefano Basagni, Imrich Chlamtac, Andras Farago, A generalized clustering algorithm for peer-to-peer networks, in: Proceedings of the Workshop on Algorithmic Aspects of Communication, held in conjunction with ICALP 1997, July 1997.

[3] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, Barry A. Woodward, A, Distance routing effect algorithm for mobility (DREAM), in: Proceedings of the Fourth ACM/IEEE Annual International Conference on Mobile Computing and Networking (MobiCom'98), October 1998, pp. 76–84.

[4] Ljubica Blažević, Silvia Giordano, Jean-Yves Le Boudec, Anchored path discovery in terminode routing, in: Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols (Networking 2002), May 2002, pp. 141–153.

[5] Ljubica Blažević, Silvia Giordano, Jean-Yves Le Boudec, Self-organized terminode routing, Cluster Computing 5 (2) (2002) 205–218. April.

[6] Prosenjit Bose, Pat Morin, Ivan Stojmenović, Jorge Urrutia, Routing with guaranteed delivery in ad hoc wireless network, Wireless Networks 7 (6) (2001) 609–616, November.

[7] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta G. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), October 1998, pp. 85–97.

[8] Mainak Chatterjee, Sajal K. Das, Damla Turgut, WCA: a weighted clustering algorithm for mobile ad hoc networks, Cluster Computing 5 (2) (2002) 193–204. April.

[9] Benjie Chen, Robert Morris, L+: scalable landmark routing and address lookup for multi-hop wireless network. Technical Report MIT-LCS-TR-837, Laboratory for Computer Science, Massachusetts Institute for Technology, Cambridge, Massachusetts, 2002.

[10] Saumitra M. Das, Himabindu Pucha, Y. Charlie Hu, Performance comparison of scalable location services for geographic ad hoc routing, in: Proceedings of INFOCOM 2005, March 2005.

[11] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, An Zhu, Discrete mobile centers, Discrete and Computational Geometry 30 (1) (2003) 45–65, May.

[12] Zygmunt J. Haas, Marc R. Pearlman, The performance of query control schemes for the zone routing protocol, in: Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 1998, pp. 167–177.

[13] Yih-Chun Hu, David B. Johnson, Implicit source routes for on-demand ad hoc network routing, in: Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001), October 2001, pp. 1–10.

[14] IEEE Computer Society LAN MAN Standards Committee, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, NY, 1997.

[15] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyumand, L. Viennot, Optimized link state routing protocol for ad hoc networks, in: Proceedings of the Fifth IEEE International Multi Topic Conference (INMIC 2001), December 2001.

[16] Xuezhi Jin, N.-H. Linda Wang, Gilles Tarjus, Julian Talbot, Irreversible adsorption on non-uniform surfaces: the random site model, Journal of Physical Chemistry 97 (17) (1993) 4256–4258.

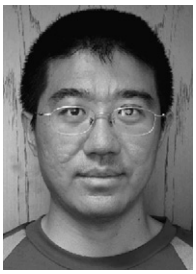
[17] David B. Johnson, David A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Tomasz Imielinski, Hank Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, 1996, pp. 153–181 (Chapter 5).

[18] David B. Johnson, David A. Maltz, Yih-Chun Hu, The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4. Internet Request for Comments RFC 4728, February 2007. <http://www.ietf.org/rfc/rfc4728.txt>.

[19] Brad Karp, H.T. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000), August 2000, pp. 243–254.

[20] Young-Bae Ko, Nitin H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: Proceedings of the Fourth ACM/IEEE Annual International Conference on Mobile Computing and Networking (MobiCom'98), October 1998, pp. 66–75.

[21] Jinyang Li, John Jannotti, Douglas S.J. De Couto, David R. Karger, Robert Morris, A scalable location service for geographic ad hoc routing, in: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000), August 2000, pp. 120–130.
[22] Wen-Hwa Liao, Yu-Chee Tseng, Jang-Ping Sheu, GRID: a fully location-aware protocol for mobile ad hoc networks, Telecommunication Systems 18 (1–3) (2001) 37–60.
[23] Chunhung Richard Lin, Mario Gerla, Adaptive clustering for mobile wireless networks, IEEE Journal on Selected Areas in Communications 15 (7) (1997) 1265–1275.
[24] A. Bruce McDonald, Taieb Znati, A mobility-based framework for adaptive clustering in wireless ad hoc networks, IEEE Journal on Selected Areas in Communications 17 (8) (1999). August.
[25] Richard Ogier, topology dissemination based on reverse-path forwarding (TBRPF): correctness and simulation evaluation. Technical report, SRI International, Menlo Park, California, October 2003.
[26] Guangyu Pei, Mario Gerla, Tsu-Wei Chen, Fisheye state routing in mobile ad hoc networks, in: Proceedings of the ICDCS Workshop on Wireless Networks and Mobile Computing, April 2000, pp. D71–D78.
[27] Guangyu Pei, Mario Gerla, Xiaoyan Hong, LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility, in: Proceedings of the First Annual Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc 2000), August 2000, pp. 11–18.
[28] Charles E. Perkins, Pravin Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in: Proceedings of the SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, August 1994, pp. 234–244.
[29] Charles E. Perkins, Elizabeth M. Royer, Ad-Hoc on-demand distance vector routing, in: Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99), February 1999, pp. 90–100.
[30] Robert D. Poor, Gradient Routing in Ad Hoc Networks, Media Laboratory, Massachusetts Institute of Technology, 2000. <http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf>.
[31] A. Renyi, On a one-dimensional problem concerning random space-filling, Publication of Mathematical Institute of Hungarian Academy of Science 3 (1958) 109–127.
[32] W.R. Salzman, Kinetic Molecular Theory: Knudsen Flow, Molecular Collisions, Mean Free Path, Department of Chemistry, University of Arizona, 2004. <http://www.chem.arizona.edu/ salzmanr/480a/480ants/knumolco/knumolco.html>.
[33] Jacob Sharony, An architecture for mobile radio networks with dynamically changing topology using virtual subnets, Mobile Networks and Applications 1 (1) (1996) 75–86.
[34] Robert H. Swendsen, Dynamics of random sequential adsorption, Physics Review A 24 (1981) 504–508, July.
[35] Paul F. Tsuchiya, The landmark hierarchy: description and analysis. Technical Report MTR-87W00152, MITRE Corporation, Bedford, Massachusetts, June 1987.
[36] Paul F. Tsuchiya, The landmark hierarchy: a new hierarchy for routing in very large networks, in: Proceedings of the SIGCOMM'88 Symposium: Communications Architectures and Protocols, August 1988, pp. 35–42.
[37] Paul F. Tsuchiya, Landmark routing: architecture, algorithms, and issues. Technical Report MTR-87W00174, MITRE Corporation, Bedford, Massachusetts, May 1988.
[38] Jungkeun Yoon, Mingyan Liu, Brian Noble, Random waypoint considered harmful, in: Proceedings of INFOCOM 2003, March 2003, pp. 1312–1321.
[39] Hongwei Zhang, Anish Arora, GS^3: scalable self-configuration and self-healing in wireless networks, in: Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing (PODC 2002), July 2002, pp. 58–67.



Shu Du is currently a graduate student in the Department of Computer Science at Rice University and is expecting his Ph.D. in Computer Science from Rice in 2007. His current research is in mobile and wireless systems, especially in routing and MAC protocols of multihop wireless systems. He received the M.S. degree in Computer Science from Rice in 2004. Before joining Rice, he received Bachelor and Master of Technology degrees in Computer Science and Engineering from Tsinghua University, Beijing, China, in 1999 and 2001, respectively.

Ahamed Khan received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology (IIT) Madras, India in 2001 and the M.S. degree in Electrical Engineering from Rice University in 2005. From 2005 to 2006 he worked at Qualcomm, San Diego in the mobile phone software division. Currently, he is working at Qualcomm, Hyderabad in the DSP group. His projects are in vocoding and echo cancellation domains. His current interests are in wireless communications, signal processing, and real time embedded systems.

Santashil PalChaudhuri received the B.Tech. degree from Indian Institute of Technology (IIT) Kharagpur in 2000, and M.S. and Ph.D. degrees from Rice University in 2002 and 2006, respectively, all in Computer Science. Since 2005, he has been associated with wireless networking startups, Meru Networks and Aruba Networks as a Member of Technical Staff. His Ph.D. thesis focused on a cross-layered architecture for wireless sensor networks, where he designed adaptive protocols for routing, scheduling and synchronization. His research interests lie in wireless networking protocols, systems, and architectures, particularly in mesh and sensor networks.

Ansley Post is pursuing his Ph.D. in Computer Science at Rice University. He is interested in self-organizing systems in general, and wireless systems specifically. He received the B.S. degree in Computer Science from Georgia Tech in 2002, and the M.S. degree in Computer Science from Rice University in 2005.

Amit Kumar Saha received the B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology (IIT) Kharagpur, India, in 1999, and the M.S. and Ph.D. degrees in Computer Science from Rice University in 2003 and 2007, respectively. He is currently with Tropos Networks, a leading company delivering metro-scale Wi-Fi mesh network systems. His research interests are in the area of mobile and wireless systems with a special emphasis on mesh networking.

Peter Druschel is the founding director of the Max Planck Institute for Software Systems. Previously, he was a Professor of Computer Science at Rice University, where he had been a member of the faculty since 1994. He does research in distributed systems, operating systems, and networks. He received the Dipl.-Ing. (FH) degree from Fachhochschule Munich, Germany, in 1986, and the Ph.D. degree from the University of Arizona in 1994. He received an NSF CAREER Award in 1995 and an Alfred P. Sloan Fellowship in 2000.

David B. Johnson is an Associate Professor of Computer Science and Electrical and Computer Engineering at Rice University. Prior to joining the faculty at Rice in 2000, he was an Associate Professor of Computer Science at Carnegie Mellon University, where he had been a member of the faculty for 8 years. He received the Ph.D. degree in Computer Science in 1990 from Rice University. Professor Johnson is leading the Monarch Project (Mobile Networking Architectures) research group at Rice and has also been very active since 1993 in the Internet Engineering Task Force (IETF), the principal protocol standards development body for the Internet. He was one of the main designers of the IETF Mobile IP protocol for IPv4 and is the primary designer of Mobile IP for IPv6, and his group's Dynamic Source Routing protocol (DSR) for ad hoc networks has been published by the IETF as an Experimental protocol for the Internet. Professor Johnson is currently the Chair of ACM SIGMOBILE, the Association for Computing Machinery's Special Interest Group on Mobility of Systems, Users, Data, and Computing. He received an NSF CAREER Award in 1995.

Rudolf H. Riedi received the M.Sc. degree in 1986 and the Ph.D. degree in 1993, both from ETH Zurich, Switzerland, in Mathematics. From 1993 to 1995, he was with B. Mandelbrot at the Mathematics Department of Yale University. He spent 1995–1997 with INRIA in Paris, France. In 1997, he joined the Electrical and Computer Engineering Department from where he moved to the Department of Statistics in 2003, both at Rice University. He currently holds a position as an Associate Professor. His research interests lie in the theory and practice of multifractals, the statistics of scaling processes, and multiscale analysis and synthesis, with applications in computer networking and computational finance. He won the ETHZ Polya prize in 1986. He has consulted with AT&T Labs and held long-term visits with the departments of ECE, Melbourne, Physics of ENS Lyon, and Computer Science of UFMG, Brazil.