Cross-Layer Discovery and Routing in Reconfigurable Wireless Networks

Christine Julien and Meenakshi Venkataraman
The Center for Excellence in Distributed Global Environments
The Department of Electrical and Computer Engineering
The University of Texas at Austin
c.julien@mail.utexas.edu, meens@ece.utexas.edu

***Abstract*— This work addresses the need for application-aware adaptive communication in mobile ad hoc networks that creates network routes based on applications' dynamic resource requests. We introduce an intuitive generalization to source routing which facilitates discovery of a resource in a mobile ad hoc network and the creation and maintenance of a route from the requesting host to the discovered destination. We thus eliminate the requirement that existing routing protocols be coupled with a name or resource resolution protocol, instead favoring an entirely reactive approach to accommodate significant degrees of mobility and uncertainty. We also present a performance evaluation and a comparison to existing alternatives.**

I. Introduction

Mobile ad hoc networks are created when mobile devices communicate directly without using an infrastructure. Applications for such networks are common when an infrastructure is unavailable (e.g., in disaster recovery situations when the infrastructure has been destroyed) or unusable (e.g., in military applications where the infrastructure belongs to the enemy). Mobile ad hoc networks form opportunistically and change rapidly in response to the movement of the connected devices, or mobile hosts. Such an environment presents a network topology that is both dynamic and unpredictable in which mobile hosts' interactions are inherently transient.

Such scenarios abound in a wide variety of application domains. In military scenarios, troops and their vehicles are becoming increasingly capable of both sophisticated data collection and dynamic wireless communication. In the field, a soldier may wish to locate mapping information, mine locations, or other data collected by his fellow soldiers. First responder applications require people with differing tasks, e.g., emergency medical technicians (EMTs), firemen, policemen, search and rescue officers, etc., to converge on a confined area and perform concurrent tasks. They collect information about the site (e.g., hot spots, smoke density, location of survivors, etc.) and benefit from accessing data collected by others' devices.

Much work on supporting applications in mobile networks builds routing protocols that maintain communication between senders and receivers. As the topology of the network changes, the protocols adjust routes to maintain end-to-end connectivity. These protocols are motivated by the desire to support the end-to-end communication common in Internet applications. These protocols require significant *a priori* knowledge. A host must know in advance the unique addresses of the other hosts with which it desires to communicate, which assumes the existence of well-known and available servers that cache resource availability. A host wishing to communicate with another host must first contact the server to resolve the host's name, following which the node must employ a routing algorithm to discover and maintain a communication path to the desired destination. These *two-phase* approaches have several drawbacks:

- Electing and maintaining a stable server set in an ad hoc network incurs a significant overhead in the highly dynamic scenarios we are targeting [19].
- The cost of advertisement becomes prohibitive as the number and variance of data sources increases.
- When resources are highly dynamic, maintaining a consistent registry requires significant numbers of control messages [10].
- In a purely ad hoc network, the servers may themselves be mobile and dynamic, requiring an initial discovery protocol targeted at finding the resolvers.

The novel contributions of this work are as follows. We identify a set of assumptions made by existing communication mechanisms that are limiting to the

protocols' applicability to supporting real mobile applications. Second, we present a protocol for communication that overcomes these assumptions, and we include an abstract model of the protocol's behavior. Finally, we present a performance evaluation that serves to not only compare our protocol to alternatives but to demonstrate the feasibility of incorporating non-fixed length addressing into a reactive mobile ad hoc routing protocol.

This paper is organized as follows. Section II presents our motivation. Section III describes our protocol in detail. Section IV analyzes our protocol's performance through simulation and compares it to a generalization of alternatives. In Section VI, we compare our approach to similar research, and Section V takes a critical look at the expressiveness and flexibility of our protocol. Finally, Section VII concludes.

II. Motivation and General Approach

In our evaluation of existing communication mechanisms and our examination of the needs of applications in mobile ad hoc networks, we identified a mismatch between existing protocols and the needs of emerging applications. Specifically, to successfully use existing mechanisms, applications must resolve names, intentions, or service descriptions into node addresses.

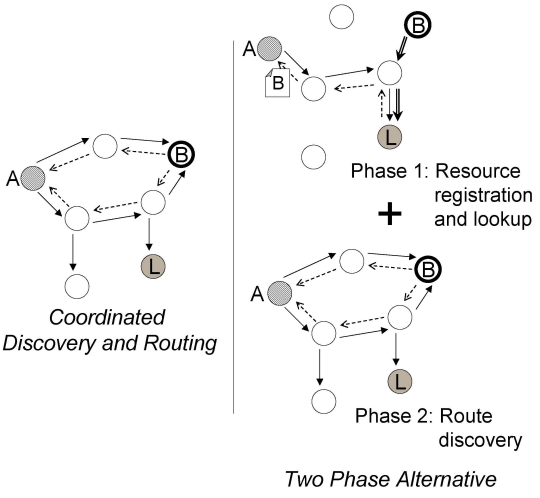

Fig. 1. Node **A** is the requester; node **B** is the destination that can provide the requested resource. Node **L** provides the lookup service. Solid black arrows indicate requests, dashed arrows represent replies, and double-lined arrows indicate service registrations.

Network overhead and message delivery latency must be of the utmost concern because hosts must take advantage of communication partners while they are connected. For this reason, we are motivated to avoid an approach which requires multiple phases of communication over the network. The alternatives are pictorially compared in Fig. 1 which shows the network traffic generated by a combined discovery and routing protocol on the left and a two-phase approach on the right. In highly dynamic networks, multi-phased interactions are more likely to cause failures as the services discovered may not actually be available when communication commences (e.g., as shown in Fig. 2).

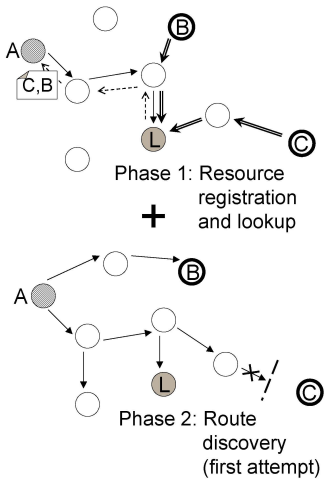

Fig. 2. The lookup service **L** returns both nodes **C** and **B** as resource providers. **A** cannot tell the difference between the two, so attempts to contact **C**, but **C** may have disappeared from the network. **A** must wait for the route discovery attempt to time out before attempting to contact **B**.

Our approach directly addresses the assumptions of existing approaches and constantly concerns itself with the performance implications of our design decisions. In the next section, we provide the details of a communication protocol that functions *without* a source having to know the unique address of the destination. Instead, route discovery is based solely on properties of the destination. To achieve this behavior, we introduce a level of indirection into a source routing protocol (e.g., Dynamic Source Routing (DSR) [17]). We selected source routing as a foundation because performance comparisons of DSR (a source routing protocol) and AODV (a distance vector routing protocol) have shown that, although the distance vector protocol achieves better performance on application level metrics like delay and throughput, source routing achieves a lower overhead in highly dynamic situations [3], [8]. The

Resource Discovery (RD):	$\langle seq\_num, source\_id, spec, route\_record \rangle$ contains a sequence number, the source id, the resource specification, and the route record
Route Reply (RR):	$\langle seq\_num, source\_id, route\_record \rangle$ contains the same sequence number and source id, but contains the complete path.
Route Error (RE):	$\langle link\_end1, link\_end2, reverse\_route \rangle$ contains the two hosts where the error occurred and the reverse of the original route.
Application Packet (P):	$\langle packet\_num, source\_id, spec, route\_record, application\_data \rangle$ contains the data, a unique packet number, the source id, the specification, and the route.

Fig. 3. CDR Packet Types

goal of this work is not to provide a highly-tuned protocol but to evaluate the feasibility and desirability of incorporating application-level information directly into a routing protocol.

III. A Protocol for Cross-Layer Discovery

The novelty of our approach lies in the fact that we achieve support for realistic applications, while having an acceptable impact on system performance, especially when compared with viable alternatives.

A. Request Specification Language

Our protocol's routing packets carry an application level specification of the destination instead of its fixed length address. A host may provide a number of capabilities, store different types of data, or satisfy varying requirements (e.g., it may be connected to a printer, it may function as an FTP server, or it may collect local traffic information). In the first responders application, a vital sign monitoring device provides information about an injured individual. In general, a mobile host that wishes to communicate with others does not know *a priori* which other host(s) will satisfy its needs.

Many possible solutions for providing resource descriptions and specifications exist [2], [13], [26]. In general, most approaches use *semi-structured data* [1] where attributes are related hierarchically. We assume that not only are the capabilities (i.e., resources, data, services, etc.) a host provides described in such a manner, but that application data is also structured so that it can be matched by similarly structured queries. The determination of such structures, especially in the case of data items, is likely to be application dependent and this is one of the major motivations for a *cross-layer* design, i.e., a design that performs actions not only at the network layer but also at the application layer. The specific description scheme used is not important, and a particular application may choose to swap out one specification language for another (we use a simple example scheme in Section IV).

B. CDR Protocol Fundamentals

Our protocol, *Cross-layer Discovery and Routing* (CDR) enables route discovery between two hosts based solely on attributes of the destination host, its resources, or its data. As part of discovery, a source route is generated that contains a list of the hosts connecting the source to each potential provider. Throughout our protocol description, we provide an abstract model of its behavior, an aspect often overlooked in communication protocol design. We view the elucidation of this model as essential to clearly stating the behavior of a protocol and the assumptions on which it relies. This makes explicit the state maintenance needs at each host and guides the careful design of our implementation.

CDR utilizes four packet types, described in Fig. 3. Each host also stores some state information, shown in Fig. 4, relating to its previous and pending requests, existing routes, and requests made by other sources.

C. Application Interaction

Fig. 5 shows the send action triggered by the application in I/O Automaton notation [20]. We show the behaviors only of host *A*, indicated by the subscript *A*. Each *action* (e.g., SendApplicationPacket$_A$) has an effect guarded by a precondition. Actions without preconditions are *input actions* triggered by another host. In the model, each action executes as one atomic step.

To abbreviate the formal description, we make two assumptions. We assume each host only attempts to send one application packet at a time and that a satisfactory destination exists and will be discovered. Both assumptions are removed in our implementation. We abuse I/O Automata notation slightly by using, for example "send *ResourceDiscovery*(RD) to *Neighbors*" to indicate a sequence of actions that triggers ResourceDiscoveryReceived (Fig. 7) on each neighbor.

When an application triggers SendApplicationPacket, the data and resource description are en-

Neighbors	the set of neighboring hosts (i.e., hosts to which this host is directly connected); Our implementation uses broadcast instead.
KnownRequests	a record of the (RD) packets this host has seen. This enables controlled flooding.
RouteCache(spec)	the routes that satisfy *spec*, sorted from lowest to highest latency.
PendingPacket	the packet waiting for a route discovery
seq_num	the sequence number for this host's resource discoveries; also enables controlled flooding.
packet_num	numbers packets sent by this host; used in resending packets that experience errors.
SentPackets(packet_num)	application packets sent by this host; used in resending packets that experience errors.
ResourceTable	semi-structured descriptions of this host's resources, matched against requests
Resends	application packets queued to be resent due to transmission failures.

Fig. 4. CDR State Information

```
SENDAPPLICATIONPACKET_A(P)
  Precondition:
    PendingPacket = NULL
  Effect:
    PendingPacket := P
    if RouteCache(P.spec) = ∅ then
      RD := ⟨ seq_num++, A, P.spec, {A} ⟩
      send ResourceDiscovery(RD) to Neighbors

TRANSMITAPPLICATIONPACKET_A(P)
  Precondition:
    PendingPacket = P
    RouteCache(P.spec) ≠ ∅
  Effect:
    route := RouteCache(PendingPacket.spec).head
    P.route_record := route
    P.packet_num := ++packet_num
    send ApplicationPacket(P)
        to P.route_record.successor(A)
    SentPackets(packet_num) := P
    PendingPacket := NULL
```

Fig. 5. Sending an Application Packet

capsulated in P. If no satisfactory route exists, the action creates a *Resource Discovery* (RD) and sends it to each neighbor. When discovery completes, the *RouteCache* will contain at least one satisfactory route. This enables the second action in Fig. 5, in which the host sends the packet on the best available route. The "send *ApplicationPacket*(P) to P.*route_record.successor*(A)" clause triggers APPLICATIONPACKETRECEIVED(P) on the the second host in the *route_record*. The host also stores a copy of the packet in *SentPackets* in case of a failure.

Fig. 6 shows APPLICATIONPACKETRECEIVED. If the host is the intended destination, the packet is passed to the application. Otherwise, the packet is propagated by selecting the next host in the route record and triggering that host's APPLICATIONPACKETRECEIVED(P) action. If the next link referred to in P's *route_record* no longer exists, an error message is

```
APPLICATIONPACKETRECEIVED_A(P)
  Effect:
    if P.route_record.tail = A then
      deliver application packet
    else
      if P.route_record.successor(A) ∈ neighbors then
        send ApplicationPacket(P)
            to P.route_record.successor(A)
      else
        reverse_route := reverse(P.route_record)
        RE := ⟨ A, P.route_record.successor(A),
                P.packet_num, reverse_route ⟩
        send RouteError(RE)
            to RE.route_record.successor(A)
```

Fig. 6. Propagating an Application Packet

generated. It uses the reverse of P's route record to target the source host.

```
RESOURCEDISCOVERYRECEIVED_A(RD)
  Effect:
    if A ∉ RD.route_record then
      if ResourceTable satisfies RD.spec then
        RR := RD
        RR.route_record := RD.route_record + A
        send RouteReply(RR)
            to RR.route_record.predecessor(A)
      else if ⟨ RD.source, RD.seq_num ⟩ ∉
          KnownRequests then
        KnownRequests :=
          KnownRequests ∪ {⟨ RD.source, RD.seq_num ⟩}
        RD′ := RD
        RD′.route_record := RD.route_record + A
        send RouteRequest(RD′) to Neighbors
```

Fig. 7. Propagating a Resource Discovery Packet

D. Resource Discovery

The above process triggers RESOURCEDISCOVERYRECEIVED, shown in Fig. 7. A receiver ensures there are no loops in the route and then determines whether or not it can act as a destination. While a

single line performs this check in our model, it uses application specific information, thus necessitating the protocol's cross-layer design, as application-level information must be accounted for in the resource discovery process. If this host does not satisfy the specification, it continues to propagate the resource discovery.

If this host can serve as a destination, the host generates a Route Reply (RR) that it returns to the source. The RR propagates using the reverse of the discovered route, triggering ROUTEREPLYRECEIVED on hosts in the *route_record* (shown in Fig. 8).

```
ROUTEREPLYRECEIVED_A(RR)
  Effect:
    if RR.source = A then
      RouteCache(RR.spec) :=
        RouteCache(RR.spec) + RR.route_record
    else
      send RouteReply(RR)
          to RR.route_record.predecessor(A)
```

Fig. 8. Propagating a Route Reply Packet

Unless the host is the source, ROUTEREPLYRECEIVED simply triggers the same action on its predecessor. If this host is the source, the route carried by the reply is stored in the *RouteCache* and associated with the appropriate application-level specification (RR.*spec*). For an initial route discovery, this insertion triggers TRANSMITAPPLICATIONPACKET shown in Fig. 5.

This description assumes that the network has symmetric links and that it is therefore possible for the packet to traverse the reverse route. If this is not the case, the destination must perform a reverse discovery using the source's unique network address. In CDR it is possible that multiple destinations will satisfy a request. A source simply selects the first host from which it receives a route reply. Section V discusses using context properties and context-sensitive application requirements to select the best path according to different metrics.

E. Route Error Propagation

When links break, transmissions encounter errors. The host detecting the broken link sends a *Route Error* (RE) to the source. On receiving an RE, if the host's route cache contains no additional routes for the desired specification, the source reinitiates route discovery.

Fig. 6 showed how the RE is generated by a host that detects a broken link. This host triggers ROUTEERRORRECEIVED on the previous host in the source route. Fig. 9 shows how this packet is propagated. The propagation of an RE does not guarantee that it reaches the original source; it may encounter link failures itself, and we do not attempt to recover from these. In such cases, the original source may not learn that its packet was not properly delivered. A host receiving an RE deletes any routes it stores that

```
ROUTEERRORRECEIVED_A(RE)
  Effect:
    for each route ∈ RouteCache do
      if RE.link_end1 ∈ route and
        RE.link_end2 = route.successor(link_end1) then
          RouteCache := RouteCache - route
    if RE.route_record.tail = A then
      Resends := Resends ∪ SentPackets(RE.packet_num)
    else
      send RouteError(RE) to RE.route_record.successor(A)
```

Fig. 9. Propagating a Route Error Packet

also use the broken link. When the packet reaches the source, it pulls a copy of the packet that experienced the transmission error from *SentPackets* and queues it for retransmission.

To complete our protocol's specification, we must also ensure that these packets are retransmitted. To this purpose, we add a RETRANSMITPACKET action which is the same as SENDAPPLICATIONPACKET in Fig. 5 except that the packet comes from *Resends* instead of directly from the application. This new action does not guarantee fairness between application sends and resends due to route errors; if the application continues to send packets, packets that need to be resent may never get the chance.

IV. ANALYSIS AND EVALUATION

In this section we provide a first step in demonstrating the feasibility of incorporating resource directed routing into highly dynamic mobile applications. We used the ns-2 network simulator to generate these results.

A. Simulation Settings

Our simulations utilized node mobility patterns based on *random waypoint* mobility [3] with 51 nodes in a rectangular field of size 1500m $\times$ 300m. Nodes' speeds were uniformly distributed between 0.01 and 20m/s, with the exception of one node, which was stationary in the center of the field. We varied the nodes' pause times from 0 seconds (for high mobility) to 900 seconds (for static networks). Our simulations

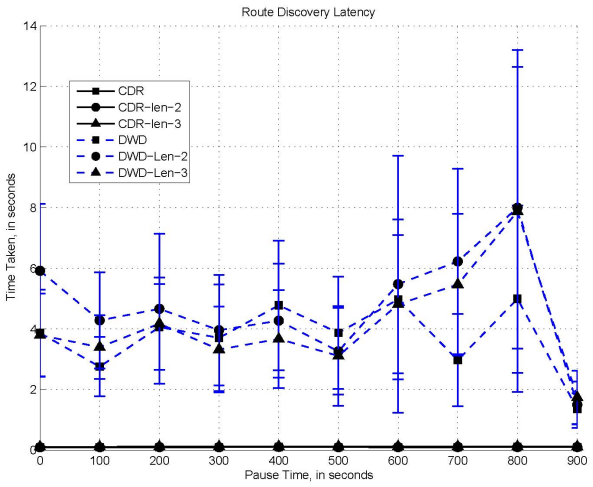

Fig. 10. Route discovery latency for one provider

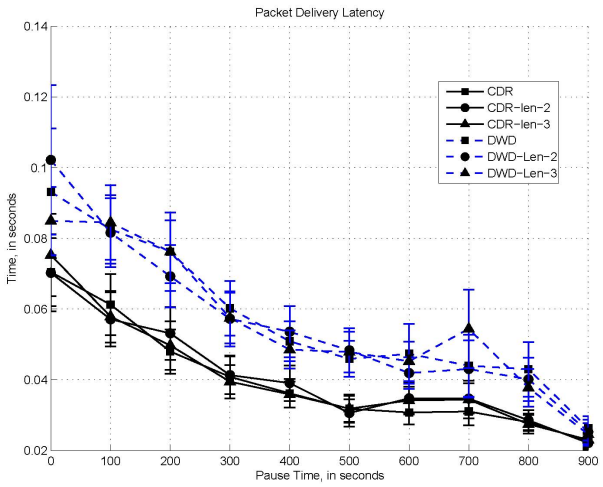

Fig. 11. Data delivery latency for one provider

used the 802.11 MAC. Each simulation is run for 900 simulation seconds, and each plotted point indicates an average over 200 samples. Traffic between sources and destinations was generated at the rate of 20 packets per second. All data packets were of size 512 bytes. Finally, we used resource descriptions of three different lengths: the basic description (17 bytes), "len-2" (45 bytes), and "len-3" (107 bytes). The descriptions searched for printers using increasingly specific descriptions.

B. Performance Metrics

Two important factors in CDR's design are the use of non-fixed length addressing and the incorporation of discovery into routing. We have chosen performance metrics specifically to measure the impact of these decisions on various aspects of the system's performance:

- *application packet delivery ratio:* the percentage of data packets successfully delivered.
- *discovery latency:* the amount of time it takes a source to know a satisfactory route.
- *data delivery latency:* the total time to deliver a data packet to a satisfactory destination.
- *normalized byte overhead:* the number of bytes of control data for each data packet delivered.
- *average route length:* the number of hops from the source to the selected destination.

For each metric, we report 95% confidence intervals.

C. Protocols

To gauge CDR's improvement over current approaches, we compared it to a protocol we call DWD (DSR with Discovery). Such a protocol is representative of those that utilize a lookup service to resolve the name or type of a service before subsequently contacting the node based on its id. We placed a lookup server on the center (stationary) node. Before creating a route, a source in DWD must contact the lookup server with the description of its desired resource. The lookup server responds with the node (or nodes) in the network that provide that service. We assumed that the lookup node had complete knowledge of the service providers in the system and therefore do not require service providers to register with the lookup service. As such, our measurements do not include the overhead associated with registrations and their renewals. We also assume that a source already knows the id of the lookup server and does not need to discover it. The ns-2 implementation of DSR was used to create routes between sources and the lookup server and between sources and destinations.

D. Results

We first compare CDR to DWD given a single available matching resource. This first set of figures shows results for each of the three description lengths for each protocol. We first compare the protocols' discovery latencies. Fig. 10 shows that, when it is necessary to discover a new route, it takes DWD (on average) much more time. This is due in part to the fact that in CDR, the route discovery time is correlated to the resource's proximity to the requestor, which is not the case in DWD. The wide confidence intervals on these measurements also indicate the wide variance of the results. Sometimes, DWD could return a route quite quickly; other times, however,

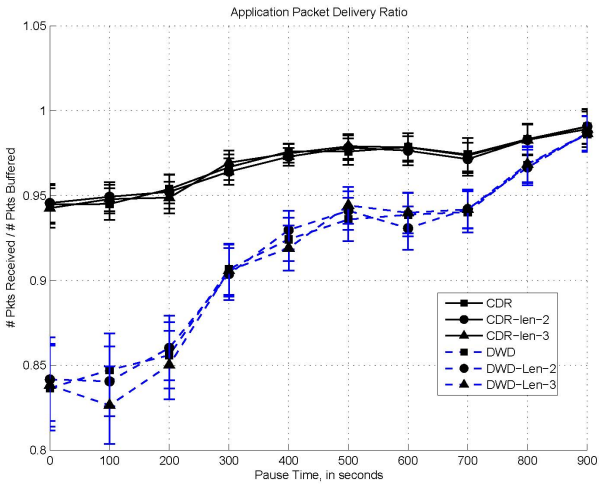

Fig. 12. Packet delivery ratio for one provider

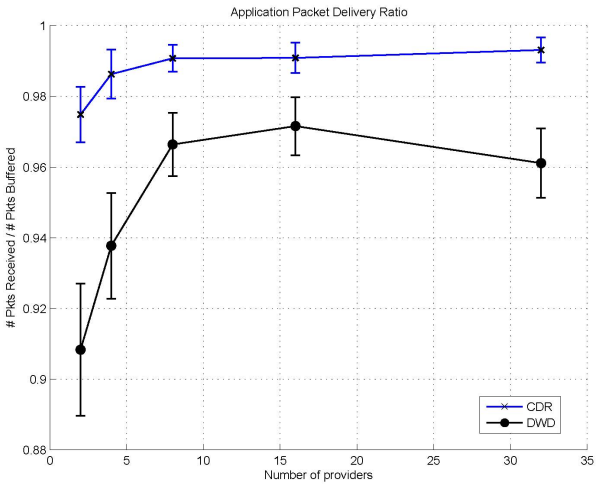

Fig. 14. Packet delivery ratio for multiple providers

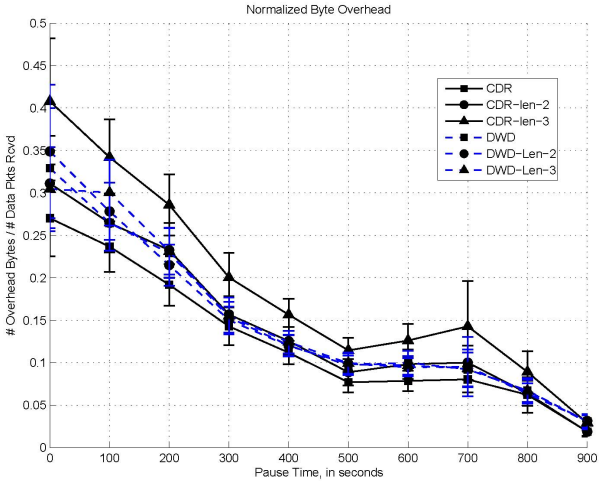

Fig. 13. Normalized byte overhead for one provider

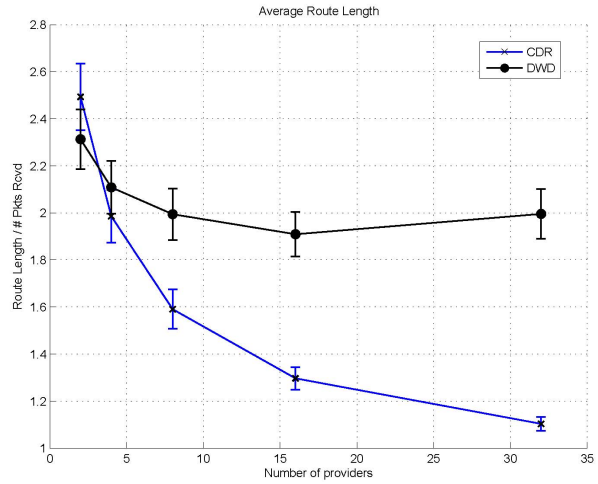

Fig. 15. Path length for multiple providers

the discovery server was out of range (as shown in Fig. 2) or took a long time to contact, even when the potential destination was quite close.

The discovery latency shown in Fig. 10 is only a portion of the total data latency applications perceive. The latter is shown in Fig. 11. When averaged over all application data packets (not just those that generated a discovery), the latency of DWD is still greater than for CDR. The difference is approximately a constant and represents the difference shown in Fig. 10 amortized over all successfully delivered application data packets.

Fig. 12 compares the application packet delivery ratio for CDR and DWD. The figure shows that, especially in situations of high mobility (low pause times), the CDR's delivery ratio is much higher than DWD's. This can be attributed to the fact that, in some situations in DWD, the discovery service was not contactable even though the potential destination was, but DWD was unable to deliver the application's message. In other cases, the destination was initially reachable, but the mobility that occurred during the time it took to discover the identity of the destination made it unreachable.

Finally, we compared the protocols' network overhead. We expected that, especially for larger descriptions, the overhead for CDR would be larger because the flooding portion of the protocol carries this larger description. In DWD, the flooding portion of the protocol carries only the fixed-length address of the destination. As Fig. 13 shows, for the shorter descriptions (len-1 and len-2), CDR's overhead is comparable to DWD's. As the description length increases, CDR's overhead also increases, while DWD's stays relatively unchanged. For the very expressive description (len-

3), the overhead of CDR is only marginally greater than for the other cases, and the previous results demonstrate that this increased overhead comes with a significant improvement to other application-level metrics like latency and delivery ratio.

The second set of experiments we performed placed multiple providers for the same resource within the network. We ran tests for 2, 4, 8, 16, and 32 potential providers. These scenarios are more likely representative of emerging mobile applications where multiple nodes can provide the same or similar functionality. Fig. 14 shows how the packet delivery ratio for the two protocols compares in this situation. CDR again outperforms DWD on this metric. The increase in success for CDR in comparison to DWD reflects the fact that CDR's single phase of communication quickly targets locally available resources. This argument is bolstered further by the results depicted in Fig. 15 which show the average path length for the routes selected by each of the protocols. CDR consistently selects a path of much shorter cost than DWD, indicating CDR's inherent preference for closer resources. DWD on the other hand, cannot tell from the list it receives from the lookup service which particular resource would be the optimal choice. In addition to benefiting our stated performance metrics, this also benefits applications in pervasive mobile environments that are often likely to prefer resources that are more "local," and, in wireless environments, the number of hops is often a decent measure of locality.

V. Discussion and Future Work

We have presented a novel model of communication that holds promise in supporting future mobile applications. This section examines the subsequent steps that must be taken to build on these results to create a deployable, usable, and expressive resource-directed discovery and routing protocol in highly dynamic environments.

1) Reactive versus Proactive Approaches: Section II outlined our rationale for an entirely reactive protocol. Briefly, our decision is motivated by the fact that our target applications operate in highly dynamic and data rich environments where advertising *all* of the available data proves too costly in terms of communication overhead. This is in stark contrast to service provision, which operates under the assumption that a widely-used set of services will be desired by multiple applications which therefore benefit from distributed advertisement of the services. Given the potential for success of our resource-directed protocol described in this paper, further extensions may include a limited proactive behavior based not on the nature of the data or resource but on the nature of the requests for it. That is, once the frequency of requests or number of requesters reaches a certain (adaptive) threshold, it may make sense to proactively distribute data in a limited local region (depending on the extensiveness of the dynamics of the environment). Future work will investigate the feasibility of such modifications with respect to metrics for measuring when to adapt and the degree to which proactive behavior should be used. This type of adaptive protocol differs from the Zone Routing Protocol (ZRP)'s [14] use of hybrid proactive/reactive behavior in that their scheme uses only network topology information to adapt, while we promote using network and application context.

2) Multiple Route Caching and Updating: A resource request can generate multiple routes to the same destination. In addition, because we do not use a unique identifier to specify the destination, multiple distinct destinations may satisfy the request. For now, we simply choose the one with the lowest latency. Additional metrics can be easily incorporated, e.g., relative location or load. New issues arise, however, because some interactions between a source and a destination, once initiated, may have long-lived state that impacts future interactions. This state may have to be maintained as the connection switches from one destination to another. For the moment, this concern is ignored in our protocol, and there are many cases when this is acceptable or even desirable. For example, if the data resource is local temperature information, it is desirable, that, as the device moves, a more local resource is selected in preference to an old connection to a more distant resource. On the other hand, if the interaction is a bidding negotiation between a buyer and a seller, automatically switching to a new seller would disrupt any ongoing transactions. Additional protocols can be integrated with CDR for transparently migrating existing state information from one resource to another when acceptable or, in the worst case, ensuring clean and announced disconnection from disappearing resources [15].

VI. Related Work

Routing protocols for mobile ad hoc networks can generally be divided into two categories: *proactive* and *reactive*. Proactive protocols (e.g., DSDV [23]) maintain routes between each pair of hosts in the

network. Reactive, or on-demand protocols (e.g., AODV [24] and DSR [17]) create routes only when requested by a particular source and maintain them only until they are no longer used. The Zone Routing Protocol [14] is a hybrid that leverages proactive behavior within a local "zone" surrounding a node and switches to reactive behavior outside of that zone. These routing protocols require the application to provide the unique address of the destination to create a route instead of specifying properties of a desired resource.

Service discovery approaches, e.g., [19], [9], [12], [7], allow applications to query resource resolvers based on descriptions instead of names. It is this style of approach that the DWD protocol in the previous section mimics. *Publish-subscribe* systems provide a service similar to our goals, and the concept has been applied successfully in infrastructure mobile networks [4] and even in mobile ad hoc networks [29]. The philosophical bases of name resolution, service discovery, and publish-subscribe approaches assume that multiple subscribers will be simultaneously interested in the same publication. As a result, the architectures use varying degrees of proactive behavior for resources or data to announce or advertise their presence. In highly dynamic networks, this generates significant overhead that is often not necessary given the expected behavior of applications.

Our work is not the first to propose application-level or content-directed communication. Content Based Multicast (CBM) [30] pushes messages to receivers based on the message's content. This is complementary to our approach in that it supports push interactions. Network Abstractions [25] uses a multicast to collect and maintain a set of the identities of hosts that satisfy an application level property. Messages are subsequently sent only to the collected set of nodes. Application-oriented routing [21] extends TORA [22] and uses a combination of proactive and reactive behavior, requiring hosts to perform topologically limited advertisements of their services. Such an approach is targeted towards scenarios in which applications share common interests and are therefore often looking for similar things.

Work more closely aligned with our goals integrates resource discovery and route construction in mobile ad hoc networks [10], providing an implementation of the architecture requirements first elucidated in [18]. This work enhances AODV [24] to simultaneously discover services and routes to them, but the approach assumes a predefined and well-known mapping of service descriptions to fixed length integers. This significantly limits the protocol's flexibility. Similarly, EDSR, part of the MPP protocol suite for providing peer-to-peer interactions in mobile ad hoc networks [11], [27] extends the DSR protocol [17] to allow route requests to allow hosts to be addressed by content instead of unique address. However, the approach uses a rigid naming scheme based on hash functions and no evaluations are provided that directly measure the impact of the new naming scheme on routing performance. The Group-based Service Routing Protocol (GSR) [6] does incorporate an expressive naming scheme for describing resources but relies on the cooperation of an advertisement scheme [5] to aid in the resource discovery process. In a similar vein to our work, [28] augments both DSR and DSDV with service discovery capabilities and demonstrates that the service selection algorithm has a significant impact on the network's throughput.

Directed diffusion [16] is an attribute based routing scheme targeted directly for sensor networks. The communication occurs in two "phases;" the exploratory phase creates a network of gradients (and floods responses back to the requester). The "best" gradients are subsequently selected through reinforcement. This protocol operates in environments where nodes commonly coordinate to perform a specific sensing task and can take advantage of this cooperation to aggregate messages destined for a sink node. We target a drastically different communication environment, where the traffic flows are neither predictable, persistent, nor deterministic and many nodes serve as "sinks." In addition, we focus on the feasibility of non-fixed length addressing in a dynamic scheme, where the presented directed diffusion implementation makes strong assumptions regarding an agreed-upon naming representation. None of these existing pieces of work combines a flexible naming scheme with a completely distributed discovery environment or evaluates the impact of routing with non-fixed length addressing on the overhead of communication in highly mobile environments, the two most significant contributions of the presented work.

VII. Conclusion

This paper has presented a novel communication protocol, *Cross-layer Discovery and Routing* (CDR) that alleviates the need for applications in a mobile ad hoc network to contact a well-known repository to create routes among mobile hosts. As we set out to bridge the gap between existing communications

approaches and applications' requirements, we started with the motivation that the combination of a reactive protocol with the source routing paradigm holds the most promise for a responsive and flexible mechanism (Section II). We presented CDR, providing a formal abstract characterization of the protocol (Section III). To examine the feasibility of incorporating non-fixed length data and resource descriptions into a reactive routing protocol, we performed a simulation analysis of our protocol and compared it with alternatives (Section IV). Finally, we examined the implications of the most fundamentally unique aspects of our protocol and identified areas for enhancements (Section V). The work presented in this paper provides a necessary and significant first step in supporting real-world dynamic and adaptive applications for emerging mobile ad hoc network scenarios.

ACKNOWLEDGEMENTS

The authors would like to thank the Center for Excellence in Distributed Global Environments for providing research facilities and the collaborative environment. This research was funded, in part, by the National Science Foundation (NSF), Grant # CNS-0620245. The conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

[1] S. Abiteboul. Querying semi-structured data. In *Proc. of the 6^{th} Int'l. Conf. on Database Theory*, pages 1–18, 1997.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[3] J. Broch, D. Maltz, D. Jounson, Y.-C. Hy, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of Mobicom*, pages 85–97, 1998.

[4] M. Caparuscio, A. Carzaniga, and A. Wolf. Design and evaluation of a support service for mobile, wireless publish/subscribe applications. *IEEE Trans. on Software Engg.*, 29(12):1059–1071, 2003.

[5] D. Chakraborty and A. Joshi. GSD: A novel group-based service discovery protocol for MANETs. In *Proc. of MWCN*, 2002.

[6] D. Chakraborty, A. Joshi, and Y. Yesha. Integrating service discovery with routing and session management for ad hoc networks. *Ad Hoc Networks Journal*, 2004.

[7] S. Czerwinski, B. Zhao, T. Hodes, A. Joseh, and R. Katz. An architecture for a secure service discovery service. In *Proc. of Mobicom*, pages 24–35, 1999.

[8] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of INFOCOM*, pages 3–12, 2000.

[9] P. Engelstad, Y. Zheng, T. Jonvik, and D. V. Thanh. Service discovery and name resolution architectures for on-demand MANETs. In *Proc. of the Int'l. Wkshp. on Mobile and Wireless Networks*, pages 736–742, 2003.

[10] C. Frank and H. Karl. Consistency challenges of service discovery in mobile ad hoc networks. In *Proc. of the 7^{th} Int'l. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Sys.*, pages 105–114, 2004.

[11] I. Gruber, R. Schollmeier, and W. Kellerer. Performance evaluation of the mobile peer-to-peer service. In *Proc. of the 4^{th} IEEE Int'l. Symp. on Cluster Comput. and the Grid*, pages 363–371, 2004.

[12] E. Guttman. Service location protocol: Automatic discovery of IP network services. *IEEE Internet Comput.*, pages 71–80, July-August 1999.

[13] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. IETF, RFC 2608, June 1999.

[14] Z. Haas, M. Pearlman, and P. Samar. The zone routing (ZRP) for ad hoc networks. IETF MANET Working Group Internet Draft, July 2002.

[15] Q. Huang, C. Julien, and G.-C. Roman. Relying on safe distance to achieve strong partitionable group membership in ad hoc networks. *IEEE Trans. on Mobile Comput.*, 3(2):192–205, 2004.

[16] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heideman, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. on Networking*, 11(1):2–16, February 2003.

[17] D. Johnson, D. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad Hoc Networking*, pages 139–172, 2001.

[18] R. Koodli and C. Perkins. Service discovery in on-demand ad hoc networks. Internet Draft, October 2002.

[19] U. Kozat and L. Tassiulas. Service discovery in ad hoc networks: An overall perspective on architectural choices and network layer support issues. *Ad Hoc Mobile Networks*, 2:23–44, 2004.

[20] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, 1989.

[21] M. Matthes, F. Apitzsch, M. Lauer, and O. Drobnik. Application-oriented routing for mobile ad hoc networks. In *Proc. of the European Wireless Conf.*, 2004.

[22] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of INFOCOM*, pages 1405–1413, 1997.

[23] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers. In *Proc. of SIGCOMM*, pages 234–244, 1994.

[24] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proc. of WMCSA*, pages 90–100, 1999.

[25] G.-C. Roman, C. Julien, and Q. Huang. Network abstractions for context-aware mobile computing. In *Proc. of the 24^{th} Int'l. Conf. on Software Engg.*, pages 363–373, 2002.

[26] Salutation Consortium. The salutation webpage, 2004.

[27] R. Schollmeier, I. Gruber, and F. Niethammer. Protocol for peer-to-peer networking in mobile environments. In *Proc. of ICCCN*, pages 121–127, 2003.

[28] A. Varshavsky, B. Reid, and E. de Lara. A cross-layer approach to service discovery and selection in manets. In *Proc. of MASS*, pages 459–466, 2005.

[29] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *Proc. of the 1^{st} Int'l. Wkshp. on Mobile Peer-to-Peer Comput.*, pages 92–97, 2004.

[30] H. Zhou and S. Singh. Content based multicast (CBM) in ad hoc networks. In *Proc. of Mobihoc*, pages 51–60, 2000.