Efficient Content Routing in MANETs Using Distances to Directories

J.J. Garcia-Luna-Aceves[†*], Qian Li[*]
[*]University of California, Santa Cruz, CA 95064
[†]PARC, Palo Alto, CA 94304
{jj, liqian}@soe.ucsc.edu

***Abstract*—The content routing protocols for MANETs proposed to date require the flooding of content requests or link-state advertisements. The Adaptive Publish-subscribe Distance Vector (APDV) protocol is presented as an alternative. APDV combines routing to well-known directories using distance-vector signaling with publish-subscribe mechanisms to discover content. Named data objects (NDO) are published with directories, and consumers of content obtain routes to NDOs from those directories. Each node is covered dynamically by a minimum number of directories within a few hops. APDV is shown to be loop-free at every instant, to provide correct routes within a finite time, and to have a small route stretch. APDV is compared with representative protocols for routing to NDOs using link-state information and on-demand routing; the results show that APDV incurs orders of magnitude less control overhead in large wireless networks.**

I. Introduction

Given the limitations of traditional routing to destination addresses, many proposals have been made for information centric networking (ICN) [1] in which content is routed by names and routers cache content opportunistically. However, relatively few results have been reported on routing of named data objects (NDO) in mobile ad hoc networks (MANET). The challenges posed by node mobility have been considered [15]; however, as Section II summarizes, few proposals have been made to support name-based routing of content in wireless networks. Interestingly, the protocols proposed to date for routing to NDOs in MANETs rely on the establishment of overlays, the flooding of link-state information, or network-wide dissemination of content requests. These approaches incur excessive signaling overhead as the number of network nodes or NDOs increase. We introduce the *Adaptive Publish-subscribe Distance Vector* (APDV) protocol for routing of NDOs in MANETs. APDV is a new approach to routing [14] in MANETs that eliminates most of the flooding needed for signaling.

Section III describes APDV as it applies to routing of NDOs. It consists of three components: (a) electing a subset of nodes to serve as directories that maintain routes to nearby destinations, (b) maintaining routes to all known directories using distance vectors, and (c) using publish-subscribe mechanisms with which directories are informed of routes to NDOs and consumers obtain routes to NDOs from directories. Section IV addresses the correctness of APDV and the route stretch it incurs. Section V describes the results of a simulation experiment comparing the performance of APDV to that of on-demand and proactive protocols for name-based routing of content. The impact of the number of NDOs requested on the signaling overhead incurred by the protocols is examined for the case of a 100-node network. The results from this experiment shows that APDV incurs orders of magnitude less signaling overhead than on-demand and proactive routing to NDOs.

This work was sponsored in part by the Baskin Chair of Computer Engineering at UC Santa Cruz, and by the US Army Research Office under Grant SUB0700155/SC20070363.

II. Related Work

There is a large body of work on resource and service discovery in ad hoc networks [17], [6]. What is striking about this prior work is that all proposals either assume that names are mapped to addresses and routing to those addresses is done independently (e.g., ADNS [4]), or augment existing routing protocols with service discovery functionality [7].

Varvello et. al. [16] proposed the use of a geographic hash table to allow routing to content rather than destinations. The limitation of this approach is that GPS is needed at each node, and a DHT is used in which the directory entries of NDOs may be far away from their producers and consumers. Geo-routing is needed to guide the subscriptions to content, which often requires the use of flooding to cope with dead-ends, and routes to content can be much longer than shortest paths.

Many schemes have been proposed based on establishing a distributed hash table (DHT) over a virtual topology defined on top of the physical network (e.g., [19], [2]). The advantage of this approach is that the DHT size grows only logarithmically with the number of intended destinations. However, a virtual link in the virtual topology can correspond to a multi-hop path in the physical network topology, and signaling overhead must be incurred to maintain such links, which becomes excessive in large MANETs. AIR [3] uses a DHT based on prefix labels of nodes that runs directly on the physical topology, rather than a virtual topology, and supports routing to content by name. The limitation of AIR is that the routes to content or destinations can be much longer than shortest paths.

There are many proposals for routing to NDOs in MANETs based on the flooding of requests for content. DIRECT [13] supports name-based routing of content in disrupted networks by the persistent flooding of content interest requests within

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

and across connected network components and opportunistic caching of content. LFBL [10] is a similar approach aimed at connected MANETs and based on the flooding of content requests on demand and opportunistic caching.

III. APDV

APDV (Adaptive Publish-Subscribe Distance Vector) assumes that each network node has a network-wide unique name, and each piece of content is an NDO. The names assigned to NDOs are structured, with each name of an NDO containing a *publisher name* component that uniquely denotes a publisher, and an *object name* component that is unique for a given publisher. For example, "soe.ucsc.edu" in the name "soe.ucsc.edu/papers/apdv-NOM13 paper.pdf" is the name of the publisher of the NDO.

A subset of routers are selected dynamically to serve as *directories*. A directory maintains the routes to nodes nearby, as well as some mappings of NDO names to the names of nodes. The distributed algorithm used to select directories ensures that each non-directory node is within a maximum distance r from a minimum number k of *local directories* for the node. Fig. 1 illustrates the basic operation of APDV assuming that each node has at least one local directory within two hops. In the example of Fig. 1, nodes *a, k, m, x* and *r* are the elected directories of the network.

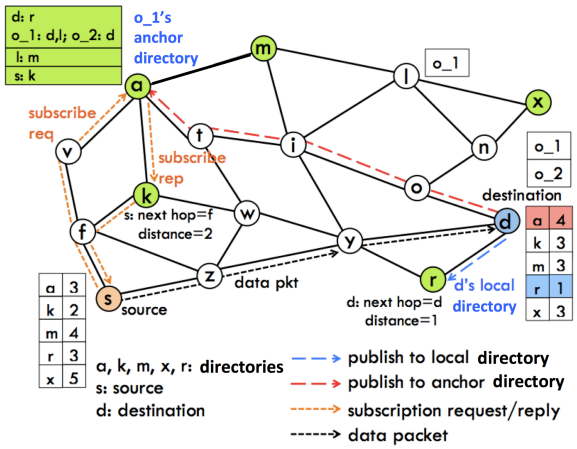

Fig. 1: Example of APDV operation

All network nodes maintain loop-free routes to all directories using sequence-numbered distances. For simplicity, we assume that a node maintains a single route to each directory. Each node contacts each of its local directories to publish its presence. To do this, node d sends a publish message to each of its local directories with the mapping $(d, \{l_d^1, ..., l_d^k\})$, where l_d^i $(1 \leq i \leq k)$ is a local directory for node d. Each local directory l_d^i of node d and each relay between d and the directory receiving the publish request from d stores a tuple stating d, the next hop to d, and $\{l_d^1, ..., l_d^k\}$.

A node d uses a *common hash function* on the publisher portion of the name of an NDO it stores to publish it with an anchor directory. The anchor directory is the directory whose name is the closest match to the hash of the NDO name among all directories. A node may publish an NDO individually or may publish all NDOs sharing a given name prefix using a single entry. For convenience, we use the term NDO to denote an individual NDO or a name prefix.

Node d provides the anchor directory of the NDO it publishes using name ndo with the tuple $(ndo : d, \{l_d^1, ..., l_d^k\})$, which maps the name of the NDO to the name of a node storing it and the local directories near the storage node. Relay nodes between d and anchor directories may cache the information published by d for a period of time. In the example of Fig. 1, node d has two NDOs, it publishes its presence with its local directory (node r), and publishes NDOs o_1 and o_2 with node a, which is the anchor directory for both NDOs. Directory r has a route to node d, while directory a has a route to directory r and the mappings $(o_1 : \{(d, r), (l, \{m, x\})\})$, and $(o_2 : (d, r))$. A node requiring NDO o_i uses the common hash function on the publisher's portion of the name of o_i, and sends a subscription request to the resulting anchor directory. In turn, the anchor directory for NDO o_i replies with the mapping of o_i to the nodes that store it and the local directories near those nodes. In the example of Fig. 1, a node requesting NDO o_1 would contact node a and receive the mapping $(o_1 : \{(d, r), (l, \{m, x\})\})$.

A. Information Stored and Exchanged

Node i maintains two tables with information about directories. A *directory table* (DT^i) states, for each directory c in the network, its name (n_c^i); the distance from i to c (d_c^i); the next hops from i to c (s_c^i); and a sequence number (sn_c^i) creed by c and used to avoid routing loops. Node i also maintains a *neighbor directory table* (NDT^i), which stores the directory tables reported by each neighbor of node i. The entry for directory c reported by neighbor j and stored in NDT^i is denoted by $\{n_{cj}^i, d_{cj}^i, sn_{cj}^i\}$.

Node i maintains a *neighbor table* (NT^i) with information about neighboring nodes and used to select those nodes that should be directories. For each neighbor j of i, NT^i stores: the name of the node (n_j^i); a sequence number (sn_j^i) created by j and used to determine that the entry is the most recent from node j; a *directory status flag* (ds_j^i) stating whether node j is a directory; the *directory counter* (k_j^i) stating the number of directories within r hops of node j; and the *local directory list* (LDL_j^i) consisting of the names of all directories within r hops of node j.

A node transmits HELLOs periodically every few seconds and a HELLO includes some or all the updates made to its tables. A node stores all the information from the HELLOs it receives from its neighbors, and also caches information it receives in subscription or publication requests from neighbors. Node i maintains routing information about NDOs and nodes in a *routing table* (RT^i) based on HELLOs containing publication and subscription requests it receives from neighbors. NT^i, DT^i, NDT^i, LDL^i, and RT^i are updated by the exchange of HELLOs among neighbors.

An entry for node j listed in RT^i specifies: the name of the node (n_j^i); a sequence number (sn_j^i) created by j and used to

avoid routing loops; the distance from i to j (d_j^i); the successor in the route to j (s_j^i); and the local directory list of node j (LDL_j^i), which may be a link to NT^i if the node is within two hops. An entry for NDO o_j listed in RT^i specifies: the name of the NDO, the name of the node that publishes it (j); and the local directory list of node j (LDL_j^i), which may be a link to NT^i if the node is within two hops.

Node i includes its own information in NT^i, i.e., it stores an entry corresponding to n_i^i, and uses the information in its HELLOs. A HELLO from node i contains: n_i^i, sn_i^i, ds_i^i, k_i^i, and updates to NT^i and DT^i. An update to NT^i regarding neighbor j consists of the tuple $\{n_j^i, sn_j^i, ds_j^i, k_j^i, LDL_j^i\}$. An update to DT^i regarding directory c consists of the tuple $\{n_c^i, d_c^i, sn_c^i\}$. An entry for neighbor v in NT^j sent in a HELLO to node i is denoted by $\{n_v^j, sn_v^j, ds_v^j, k_v^j, LDL_v^j\}$, and the same entry stored in NT^i is denoted $\{n_{vj}^i, sn_{vj}^i, ds_{vj}^i, k_{vj}^i, LDL_{vj}^i\}$.

B. Selecting and Routing to Directories

Nodes select a dominating set C of nodes that serve as directories, such that every node $u \notin C$ (called *simple node*) is at a distance smaller than or equal to r hops from at least k directory nodes in C. A node u is said to be (k, r) dominated (or *covered*) if there are at least k directories in C within r hops from u. The directory selection scheme is based on HELLO messages exchanged among one-hop neighbors. To keep the selection algorithm and signaling simple, only distances to directories and node names are used as the basis for the selection of directories.

1) Selecting Directories: Nodes self-select themselves to become or stop being directories. A given node i determines to add or delete its own entry in DT^i according to the Directory Addition Rule (DAR) and directory Deletion Rule (DDR) defined below.

Node i is initialized with $DT^i = \phi$ and $NT^i = \phi$, and waits for a few seconds to start receiving HELLOs from nearby nodes before selecting directories. Hence, according to DAR, node i selects itself as a directory when it is first initialized, unless it receives HELLOs from neighbors that prompt it not to include itself as a directory based on DDR.

Let the number of entries in a list L be donated by $|L|$, and the lexicographic value of a name n be donated by $|n|$.

Once node i has updated NT^i and DT^i by processing the HELLOs from its neighbors, it computes LDL^i from DT^i, such that $v \in LDL^i$ if $d_v^i \leq r$, and sets $k_i^i = |LDL^i|$.

DAR (Directory Addition Rule):
Node i adds itself to DT^i if
$(k_i^i < k) \wedge [\ |i| = Min\{|n_j^i| \mid \forall j \in NT^i \mid (j \notin DT^i) \wedge (k_j^i < k)\}\ ]$

DDR (Directory Deletion Rule):
Node i deletes itself from DT^i if
$(k_i^i > k) \wedge [\ \forall\ j \in NT^i\ [\ (\ |LDL_j^i - \{i\}| \geq k\ \forall j \notin LDL_i^i\ ) \wedge (|n_j^i| < |n_i^i|\ \forall j \in LDL_i^i)\ ]\ ]$

2) Routing to Directories: For simplicity, we assume that each node maintains a single route to each directory selected in the network using the updates to directory tables included in HELLOs.

To guarantee loop-free routes, APDV uses sequence numbers that restrict the selection of next hops towards a given directory by any node. Only those neighbors with shorter distances to a directory or with a more recent sequence number reported by it can be considered as next hops to the directory. An important aspect of APDV is that entries for directories can be deleted on purpose as a result of DDR, rather than only as rare occurrences due to failures or network partitions. Together with the transmission of periodic HELLOs, the Reset Directory Rule (RDR) and the Update Directory Rule (UDR) discussed below address this functionality.

Let N^i be the set of one-hop neighbors of node i. Node i updates DT_j^i as a result of HELLOs from neighbor $j \in N^i$ or the loss of connectivity to neighbor j. If node i loses connectivity to node j, the entries in DT_j^i are deleted. Once node i is selected as a directory, it is the only node that can change the sequence number for its own entry in directory-table updates sent in HELLOs.

When node i decides to delete itself as a directory based on DDR, its entry must be deleted in the rest of the network. Node i uses RDR to set its self-entry with an infinite distance and an up-to-date sequence number for a finite period of time T, before deleting its self-entry from DT^i to ensure that the rest of the nodes delete the entry for i in their directory tables. If node i receives a HELLO from j or experiences a link failure that makes it update DT_j^i for entry $c \neq i : \{n_{cj}^i, d_{cj}^i, sn_{cj}^i\}$, node i updates its entry for c in DT^i according to UDR, which forces node i to propagate a reset update or to select a successor to directory c that is either closer to c or has reported a more recent sequence number from c.

RDR (Reset Directory Rule):
If node i must delete itself from DT^i using DDR then
  set $d_i^i = \infty$; $sn_i^i = sn_i^i + 1$; *reset-timer*$^i = T$

UDR (Update Directory Rule):
If ($\exists q \in NT^i \mid sn_{cq}^i > sn_c^i$)
  then begin
    if ($(v = s_c^i) \wedge (sn_{cv}^i > sn_c^i) \wedge (d_{cv}^i = \infty)$)
      then set $sn_c^i = sn_{cv}^i$; $d_c^i = \infty$
      else begin
        set $d_c^i = Min\{d_{cf}^i + 1 \mid (f \in NT^i) \wedge (sn_{cf}^i = Max\{sn_{cv}^i \mid v \in NT^i)\}\}$
        set $s_c^i = j \mid (j \in NT^i) \wedge (d_{cj}^i = d_c^i - 1)$
        set $sn_c^i = Max\{sn_{cv}^i \mid v \in NT^i\}$;
  else begin
    set $d_c^i = Min\{d_{cf}^i + 1 \mid (f \in NT^i) \wedge (sn_{cf}^i = sn_c^i) \wedge (d_{cf}^i < d_c^i)\}$;
    set $s_c^i = j \mid (j \in NT^i) \wedge (d_{cj}^i = d_c^i - 1)$

C. Publish-Subscribe Mechanisms in APDV

Nodes learn about routes to directories and to one- and two-hop neighbors, but have no routes to NDOs. To allow nodes to obtain routes to arbitrary NDOs without incurring network-wide dissemination of signaling messages, APDV uses a publish-subscribe mechanism for name-to-route resolution.

1) Publishing NDOs: Publishing an NDO in APDV consists of making one or more local directories know the route to the node storing the NDO and having an anchor directory know the mapping from the name of the NDO to the name of the storing node and its associated local directories.

Node i publishes itself with the k directories listed in LDL^i. The local directories in LDL^i are within r hops of node i and serve as the "landmarks" for other nodes to communicate with node i, given that nodes far away from node i do not have routes to node i. Accordingly, a local directory for node i must maintain a route to node i, and it also maintains the mapping (i, LDL^i), so that it can find alternate ways to reach node i if its route to i fails.

The anchor directory for NDO o_i (denoted a_{o_i}) is obtained by using a network-wide consistent hash function that maps the *publisher name* component of the name of the NDO into the name of one of the directories selected in the network, which are listed in the directory table. Directory a_{o_i} must store the mapping (o_i, $LNDO_i$), where $LNDO_i$ is a list of tuples corresponding to the nodes that have published NDO o_i. Each tuple specifies the name of a node storing the NDO and the local directory list (LDL) of the node. This allows a node v that obtains $LNDO_i$ to request NDO o_i by sending its request towards the nearest local directory (according to its directory table) of a node storing NDO o_i.

Nodes can publish themselves by using their names to select the anchor directories using the network-wide consistent hashing function. The anchor directory for node i (a_i) stores the mapping (i, LDL^i), so that it can provide any node v far away from node i the list LDL^i, with which node v can send data packets towards a directory in LDL^i that is nearest to node v.

The forwarding of a publication request from a node to its local directories is done by the exchange of HELLOs. Given that nodes maintain loop-free routes to all directories, publication requests are forwarded over the reverse loop-free routes already established from directories to nodes. The routes maintained by local directories to nearby nodes are refreshed periodically; each node creates a new publication request by increasing the sequence number of its own HELLO.

If node i receives a HELLO from neighbor j with a publication request originated by node v, which consists of update to LRT^j for destination v ($\{nid_v^j, sn_v^j, d_v^j, LDL_v^j\}$), then node i forwards the request (i.e., it includes the LRT^i entry $\{nid_v^i, sn_v^i, d_v^i, LDL_v^i\}$ in its own HELLO) if it is the successor for node j to any of the directories listed in LDL_v^j.

Once a local directory c receives an entry for destination v and $c \in LDL^v$, then c publishes (i.e., stores) the entry $\{n_v^c, sn_v^c, d_v^c, s_v^c, LDL_v^c\}$, where s_v^c is the neighbor from which it received the publication request. Directory c may also forward it if it is the successor to another directory in LDL^v for the neighbor from which it received the publication request.

Node i can publish itself as a destination with its anchor directory a_i by node i using the network-wide consistent hash function to map its name to the name of one of the directories in DT^i to obtain $hash(i) = a_i$, where $a_i \in DT^i$. After that, node i sends a publication request to its successor towards its anchor directory a_i with the tuple $\{n_i^i, sn_i^i, d_i^i, LDL_i^i\}$. Each node v in the route from node i to directory a_i forwards the publication request towards a_i and caches the tuple $\{n_i^v, sn_i^v, d_i^v, s_i^v, LDL_i^v\}$. Once directory a_i receives the request, it stores the tuple $\{n_i^{a_i}, sn_i^{a_i}, d_i^{a_i}, s_i^{a_i}, LDL_i^{a_i}\}$. Hence, each node processing a publication request learns the route to the node issuing the request, and the anchor directory is able to obtain the mapping needed to redirect nodes sending subscription requests to the local directories of node i.

Similarly, node i publishes NDO o_k with an anchor directory using the network-wide consistent hash function on the set of directories in DT^i to obtain $hash(o_k) = a_{o_k}$, where $a_{o_k} \in DT^i$. Node i then sends a publication request to its successor towards a_{o_k} with the tuple $\{o_k, n_i^i, LDL_i^i\}$, where n_i^i is node i's name. Each node v in the route from node i to directory a_{o_k} forwards the publication request towards a_{o_k} and caches the tuple. Once directory a_i receives the request, it adds the mapping (n_i^i, LDL_i^i) and adds n_i^i to the list of nodes storing o_k. Hence, a_{o_k} learns over time the nodes that store NDO o_k and the local directories of those nodes.

2) Subscribing and Routing to NDOs: Subscribing to an NDO in APDV consists of a node requesting a way to reach an NDO by contacting the anchor directory of the NDO, and the anchor directory providing the requesting node with a list of nodes storing the NDO and the local directory lists for those nodes. The forwarding of subscription requests is handled in much the same way described above for the case of publication requests.

Node p requests NDO o_k by computing $hash(o_k) = a_{o_k}$, where $a_{o_k} \in DT^p$ and sends its subscription request towards a_{o_k}. The subscription request states the name of NDO (o_k), the anchor directory (a_{o_k}), and the list of local directories for node p (LDL^p). Directory a_{o_k} responds to node p with the mapping of o_k to the list of names of nodes storing the NDO and their associated local directory lists. The response is sent to the nearest directory it finds in LDL^p.

Similarly, if node p has data for destination node $j \notin DT^p$, it computes $hash(j) = a_j$, where $a_j \in DT^p$ and sends its subscription request towards a_j. The subscription request states the identifier of node j, its anchor directory a_j, and LDL^p. Directory a_j responds with the tuple $\{n_j^{a_j}, sn_j^{a_j}, LDL_j^{a_j}\}$ and sends the response to the nearest directory it finds in LDL^p. Once node p receives the reply to its subscription, it stores the tuple $\{n_j^p, sn_j^p, LDL_j^p\}$ in RT^p. Data packets from p are then sent towards the directories in LDL_j^p that are the closest to node o. A data packet must specify the sender, the destination, and the selected local directory of the destination. This can be done by encapsulating the header of the packet stating the origin and the destination with a header stating the origin and the selected local directory of the destination. Once the packet reaches a relay node y with an active route for the destination, the packet is forwarded directly to the destination itself, as long as the distance from node y to the destination is at most r hops.

IV. APDV Correctness

The following theorems demonstrate that APDV is loop free at every instant, that all non-directory nodes have at least k local directories within r hops within a finite time after the topology of the network is connected and stable, and that the the route stretch incurred by APDV is small.

Theorem 1: Nodes using UDR to update routes to directories yields loop-free routes to directories at every instant.

Proof: The proof is by contradiction. Assume that a routing loop L_c for directory c consisting of h hops is created when nodes in L_c change successors according to UDR. Let $L_c = (n_1, n_2, ..., n_h)$, with $n_{i+1} = s_c^{n_i}$ for $1 \leq i \leq h-1$ and $n_1 = s_c^{n_h}$.

According to UDR, for each hop $n_i \in L_c$ $(1 \leq i \leq h)$ it must be true that $sn_c^{n_i} \leq sn_c^{n_{i+1}}$ for $1 \leq i \leq h-1$ and $sn_c^{n_h} \leq sn_c^{n_1}$. This result implies that $sn_c^{n_i} = sn_c^{n_{i+1}}$ for $1 \leq i \leq h-1$ and $sn_c^{n_h} = sn_c^{n_1}$. Because of UDR, this implies that $d_c^{n_i} < d_c^{n_{i+1}}$ for $1 \leq i \leq h-1$ and $d_c^{n_h} < d_c^{n_1}$, which implies that $d_c^{n_i} < d_c^{n_i}$ for $1 \leq i \leq h$. This is a contradiction and hence the theorem is true. *Q.E.D.*

Theorem 2: APDV is loop-free at every instant.

Proof: From Theorem 1, it follows that all routes to nodes that are selected as directories are loop free. Nodes that serve as local directories for nodes nearby obtain their routes by the propagation of publication requests from those nodes along the reverse loop-free paths to local directories established following UDR. Hence, the routes maintained at local directories to simple nodes, and the routes stored by simple nodes along reverse loop-free paths to local directories must be free of loops. On the other hand, the route from a given source node to a destination far away consists of two concatenated components. The first component is a loop-free route to a local directory, which is loop free. The second component is a route from a relay node within r hops to the destination that was obtained by the propagation of publication requests from the destination along the reverse loop-free routes to a local directory, which is also loop-free. Therefore, the theorem is true. *Q.E.D.*

Assume that APDV is executed in a connected network G with a node set N, and further assume that topological changes stop taking place after a given time t_T.

Theorem 3: Each simple node in G must be covered by at least k directories located within r hops of the node.

Proof: Because nodes communicate their directory lists persistently with every HELLO they transmit, all nodes must have a consistent view of their one- and two-hop neighbors within a finite time $t_H \geq t_T$, for otherwise at least one node is unable to receive the HELLOs from a neighbor. Because all nodes have loop-free routes to any selected directory (Theorem 2), $CT = DT^i \ \forall i \in N$ within a finite time after any node v changes DT^v using DAR or DDR.

Assume that a given node u is uncovered and has only $k_u^u < k$ directories within r hops at time t_H, while the rest of the nodes are covered. Node u can only be a simple node or a directory at time t_H.

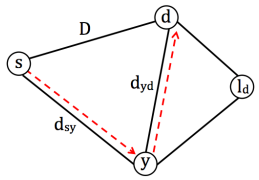

Fig. 2: Route stretch in APDV

TABLE I: Signaling Overhead for Routing to NDOs

Protocol	10%	20%	40%
APDV	162.73	176.04	199.18
AODV-NDO	157.85	10412.90	26591.93
OLSR-NDO	698.68	1441.54	2268.49

Consider the case that node u is a directory at time t_H. Node u cannot use DDR to stop being a directory at time t_H, because then node u itself and all its one- and two-hop neighbors who are simple nodes must be covered by at least k directories other than node u. Hence, if node u is a directory at time t_H, it must remain being a directory indefinitely. This implies that the theorem is true in this case, because APDV ensures only that simple nodes are covered by k directories within r hops each.

Consider the case that node u is a simple node at time t_H. It follows from DAR that node u must become a directory, because $u = Min\{nid_j^u \ \forall j \in NT^u \mid (j \notin DT^u) \wedge (k_j^u < k)\}$, given that u is the only node in its neighborhood that is not covered. Furthermore, any node v using DDR at time $t_S > t_H$ as a result of node u becoming a directory must be covered when it becomes a simple node. Hence, the theorem is also true in this case. *Q.E.D.*

Let r represent the maximum distance between a simple node and any of its local directories, and D be the shortest distance between a node and an NDO it requests. The following theorem states an upper bound on the route stretch that can be incurred by APDV, which is defined as the ratio of the distance attained by APDV between a node and a remote NDO divided by the shortest distance D.

Theorem 4: APDV incurs a route stretch smaller than or equal to $1 + 2r/D$.

Proof: Consider Fig. 2, which shows the case of a node s requesting an NDO from node d by using the route provided by APDV pointing to a local directory of d, l_d. Node y is the first node along the path to l_d with a valid route to node d, and it may be the case that y is in fact l_d. The route attained with APDV is the concatenation of the route from s to y with the route from y to d, with lengths d_{sy} and d_{yd}, respectively. The length of the shortest path from s to d is D. Accordingly, the route stretch in APDV (S) equals $S = D_A/D = (d_{sy} + d_{yd})/D$. Because y must have a route to d that is at most r hops, $d_{yd} \leq r$. Hence, by the triangle inequality it follows that $d_{sy} \leq (d_{yd} + d_{sd})$. This implies that $d_{sy} \leq (r + D)$ and hence $d_{sy} + d_{yd} \leq 2r + D$. The proof of the theorem follows by substituting this result in the expression for S. *Q.E.D.*

From the above, it is clear that the length of the routes attained in APDV tends to be the shortest-path length as $\frac{r}{D}$ becomes smaller, which is the case of very large networks.

V. Performance Comparison

We compare APDV against representative approaches of content routing protocols proposed for MANETs, with all the protocols using opportunistic content caching at each router. To isolate the impact of node mobility from content mobility (i.e., caching), and due to space limitations, we assume static network topologies in this paper. We made this choice given that prior results [14] show that APDV performs far better than traditional routing protocols (AODV [11] and OLSR [5]) in large MANETs.

We implemented AODV-NDO by modifying the AODV QualNet implementation [12] as an example of on-demand content routing. AODV-NDO operates like DIRECT [13] and LFBL [10], with routers flooding requests for NDOs on demand, followed by the delivery of NDOs from routers with local copies of the NDOs. We also implemented OLSR-NDO by modifying the OLSR QualNet implementation [12] as an example of proactive content routing. OLSR-NDO operates like NLSR [8], with routers flooding link state advertisements for physical links as well as for NDOs stored locally.

The data plane used for both OLSR-NDO and APDV is the same as the data plane advocated in recent ICN architectures [1]. An NDO request is forwarded towards the nearest router that has advertised the NDO, and the NDO is sent back over the reverse path by the first router with a local copy of the NDO being requested.

We focused on the average number of control packets generated by the routing protocols as the performance metric. All protocols use the same time period to refresh their routing structures. For APDV we used $k = 2$ and $r = 3$ to select directories. Each simulation ran for 10 different seed values. We used static networks of 100 nodes, with nodes being uniformly distributed in the network to avoid disconnected nodes. The routing protocols were tested using the IEEE 802.11 DCF as the underlying MAC protocol, and all signaling packets are sent in broadcast mode. We increased the number of nodes requesting NDOs from 10% to 40% of the 100 nodes of a static network. Nodes are uniformly distributed in a simulation area of 1800×1800 m^2. Each node originally publishes 10 NDOs, with the name of each NDO consisting of a hierarchical name including the name of its publisher and the hashing to find anchor directories is based on the publisher name of the NDO name.

Table I shows the results of the simulation experiment. It is clear that, as the rate of content requests increases, the signaling incurred in the flooding of content requests (AODV-NDO) or the updating of all nodes on the locations of new copies of NDOs (OLSR-NDO) becomes excessive. On the other hand, APDV signaling overhead increases sub-linearly with the rate of content requests because the use of directories eliminates most of the flooding in the MANET.

VI. Conclusion

We described how the *Adaptive Publish-subscribe Distance Vector* (APDV) protocol can provide efficient routing to named data objects in MANETs using distance vectors. We showed the correctness of APDV and used a simulation experiment to compare its performance with the performance of on-demand and proactive approaches to name-based routing in wireless networks. The key reason why APDV outperforms the other protocols is that it eliminates most flooding by using directories. More work is needed to fully understand and exploit the approach advocated in APDV. One important aspect is the fact that many NDOs need to be known only within a given horizon, and hence a hierarchy of directories can be used to allow NDOs to be known locally, regionally or globally. The combined effect of node and content mobility must also be analyzed.

References

[1] B. Ahlgren et. al., "A Survey of Information-centric Networking," *IEEE Commun. Magazine*, July 2012, pp. 26–36.

[2] M. Caesar et al.,"Virtual ring routing: network routing inspired by DHTs," *SIGCOMM '06*, 2006.

[3] J.J. Garcia-Luna-Aceves and D. Sampath,"Scalable Integrated Routing Using Prefix Labels and Distributed Hash Tables for MANETs," *Proc. IEEE MASS 2009*, Oct. 2009.

[4] X. Hong et al., "Distributed Naming System for Mobile Ad-Hoc Networks," *Proc. ICWN '05*, June 2005.

[5] P. Jacquet et al., "Optimized Link State Routing Protocol for Ad Hoc Networks," Optimized Link State Routing Protocol," *Proc. IEEE INMIC '01*, Dec. 2001.

[6] U. Kozat and L. Tassiulas, "Service Discovery in Mobile Ad hoc Networks: An Overall Perspective on Architectural Choices and Network Layer Support Issues," *Ad Hoc Networks*, Vol. 2, No. 1, pp. 23–44, Jan. 2004.

[7] L. Li and L. Lamont, "A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-layer Design," *Proc. IEEE PerCom '05 Workshop*, 2005.

[8] A.K.M. Mahmudul-Hoque et al., "NSLR: Named-Data Link State Routing Protocol," *Proc. ACM ICN '13*, Aug. 2013.

[9] Y. Mao et al, "S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks," *Proc. NSDI 07*, April 2007.

[10] M. Meisel, V. Pappas, and L. Zhang, "Ad Hoc Networking via Named Data," *Proc. ACM MobiArch '10*, Sept. 2010.

[11] C.E. Perkins and E.M. Royer, "Ad-hoc On-Demand Distance Vector Routing," *Proc. 2nd IEEE WMCSA*, Feb.1999.

[12] Scalable Network Technologies, *Qualnet*, http://web.scalable-networks.com/content/qualnet.

[13] I. Solis and J.J. Garcia-Luna-Aceves, "Robust Content Dissemination in Disrupted Environments," *Proc. CHANTS 08: ACM MobiCom 2008 Workshop on Challenged Networks*, Sept. 2008.

[14] Q. Li, J.J. Garcia-Luna-Aceves, "APDV: making distance vector routing scale using adaptive publish-subscribe mechanisms," *Proc. ACM MSWiM '13*, 2013.

[15] G. Tyson et. al., "A Survey of Mobility in Information Centric Networks: Challenges and Research Directions," *Proc. ACM NoM '12*, June 2012.

[16] M. Varvello et. al., "On The Design of Content-centric MANETs," *Proc. IEEE WONDS '11*, Jan. 2011.

[17] C.N. Ververidis and G.C. Polyzos, "Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques," *IEEE Comm. Surveys and Tutorials*, Vol. 10, No. 3, 2008.

[18] Y. Wang and J.J. Garcia-Luna-Aceves, "Collision Avoidance in Multi-hop Ad Hoc Networks," *Proc. IEEE MASCOTS '02*, 2002.

[19] B. Zhao et al., "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Report No. UCB/CSD-01-1141, University of California, Berkeley, April 2001.