VINE: Zero-Control-Packet Routing for Ultra-Low-Capacity Mobile Ad Hoc Networks

Ayush Dusia*, Ram Ramanathan†, Warren Ramanathan†, Christophe Servaes†, and Adarshpal S. Sethi*
*Department of Computer and Information Sciences, University of Delaware, Newark, Delaware, 19716, USA
†goTenna Inc., 81 Willoughby Street, Brooklyn, New York, 11201, USA
Email: adusia@udel.edu, ram@gotenna.com, warren@gotenna.com, christophe@gotenna.com, sethi@udel.edu

***Abstract*—We consider the problem of routing short-burst data in Mobile Ad Hoc Networks (MANETs) characterized by ultra-low data rates. Existing routing protocols exhibit poor scalability in such low-capacity regimes due to their use of control packets. We present a novel on-demand zero-control-packet routing protocol called VINE that computes cost gradients to nodes by inspecting packet headers of the received data packets, which are then used to forward the future data packets. VINE provides data reliability via per-hop implicit acknowledgments and end-to-end acknowledgments.**

We describe VINE and derive an expression for its communication complexity. We present ns3 simulation results across a wide range of network sizes, densities, and traffic that show that VINE significantly outperforms AODV across all of these scenarios, with up to ∼2.5x higher delivery ratio. VINE also provides better security by eliminating scope for control attacks. VINE has been implemented on the goTenna Pro mesh networking device for the military and public safety markets.

***Index Terms*—Mobile Ad Hoc Networks, Routing Protocol, Performance Analysis, Mobile Mesh Networks**

I. Introduction

Narrowband tactical communications (e.g., NATO NBWF [1]), off-grid disaster relief, long-range outdoor Internet-of-Things (IoT), and other contexts are characterized by very low bit-rates. For example, the bit-rate for NBWF is 20-82 Kbps [1], and for the long-range IoT standard (LoRA) is 0.3-50 Kbps [2]. Further, these technologies are required to or are envisioned to operate in a mobile multi-hop context.

The literature is replete with thousands of MANET broadcast and unicast routing protocols, and several standards exist, such as AODV [3] and OLSR [4]. However, most if not all such protocols use dedicated control packets such as Hellos, Link-State Update, Route Request/Response, etc. Ultra-low capacity MANETs cannot support this control packet overhead even for modestly-sized networks (see Section V). Routing control packets also pose a security vulnerability [5] (see section III-C).

We present VINE: a novel routing protocol for MANETs that does not utilize any routing control packets. Instead, VINE builds routing state by inspecting headers of data packets that it then uses for forwarding future data packets. Specifically, VINE uses three fields in the header, namely the sender, the previous sender, and a hop count to build routing state to 1-hop neighbors, 2-hop neighbors, and origin of the packet, respectively. Over time as traffic flows, an increasingly rich sink tree toward each node is created, resembling the growth of a "vine" in a "grove". VINE also eliminates the additional MAC- and PHY-layer header overhead on each control packet.

Packets are forwarded along non-increasing cost gradients (like water flowing downhill), until the destination is reached. If there is no fresh-enough gradient state, the packet is broadcast. This decision is taken independently at each hop – thus, a packet may alternate between broadcast (when no state exists) and unicast (when state exists) en route to its destination. Consequently, when the rate of topology change is so high that routing state cannot keep up with it, VINE automatically falls back to flooding, which ensures high delivery ratio. VINE provides per-hop reliability via *implicit acknowledgments*, that is, retransmissions based on overheard forwarded packets, and delivery notification via *end-to-end acknowledgments* – features that are not present in most routing protocols. Finally, by eliminating control packets altogether, VINE is immune to control-based routing attacks such as those mentioned in [5].

We derive an expression for the communication complexity of VINE and show that it tends to stabilize quickly. We compare the performance of VINE with that of AODV using the ns3 simulation tool. We have chosen AODV as a point of comparison because it is the basis of many standards including RPL [6], LOADng [7], and IEEE 802.11s [8]. Our simulations on low-capacity (25 Kbps) networks show that as size increases or density decreases, the packet delivery ratio of AODV falls steeply (below 40% for large networks), whereas VINE's does not (98%). VINE's total communication load (i.e., total bytes transmitted) is also significantly lower than that of AODV. Over high-capacity (1 Mbps) networks the gap narrows considerably, but VINE still provides about 10% better PDR and incurs lower routing overhead.

VINE has been implemented on the goTenna Pro [9] – a small handheld device for first-responders, military, and other professionals. VINE is targeted for applications such as 1-1 texting, unicasting low-fidelity images, and other low-bandwidth delay-tolerant and short-burst applications [10].

In summary, VINE provides a novel zero-control-packet alternative to existing protocols and is more scalable, secure, and reliable for a wide range of military and other applications.

II. Related Work

Reactive routing protocols (e.g., AODV [3], DSR [11]) exchange route request and route reply control packets with other nodes. AODV periodically exchanges Hello messages,

whereas DSR uses source routing, both resulting in increased network overhead. Proactive protocols (e.g., DSDV [12], OLSR [4]) select routes in advance by periodically exchanging routing information, either link-state or distance vector, resulting in high overhead, especially in large and dense networks, draining energy even when there is no traffic. Optimizations [13] and hybrid approaches [14] reduce the network overhead but at the risk of keeping stale routing information. Location-based approaches [15] route packets using GPS coordinates but face difficulties when the actual topologies based on path loss and connectivity are different from the constructed ones.

The gradient-routing schemes have their roots in the directed diffusion approach [16], where the sink node floods control packets, acting as queries for collecting sensed data and building reverse paths to the sink. Nodes unicast packets to the next hop on the path to the sink node. Gradient broadcast approaches [17] use a cost field in the header to forward packets in non-increasing cost direction. These approaches mostly target multi-source single-sink scenarios.

The works in [18] and [19] (called EP-RPL) describe the latest developments in LoRa [2] and RPL [6] technologies, respectively. A proof-of-concept implementation in [18] describes an extension to the single-hop LoRa networks. The work in [19] highlights the shortcomings of the traditional RPL protocols (e.g., (LOADng) [7] and P2P-RPL [20]) and describes an energy-efficient hybrid protocol that leverages the regional information in selecting a subset of nodes that participate in route discovery. However, both these works are adaptations of AODV, applying route discovery procedures with extensive use of control packets.

NBWF's evaluations in [21] identify all existing routing protocols to be inadequate for low-capacity networks and propose using link metrics in conjunction with the low-overhead Hello schemes in the original standard [1]. The delay tolerant protocols (MaxProp [22] and PRoPHET [23]) relying on the neighbor discovery and store-and-forward schemes are also inadequate for disaster relief, emergency, and public safety situations.

In contrast to all the above approaches, VINE learns cost gradients by inspecting packet headers and uses them for routing future packets, eliminating the need for any control packets. It accommodates arbitrary source-sink pairs, and has a constant size header with only two extra fields compared to a typical stack, which we show to be insignificant compared to the control packet sizes. There is no periodic exchange of routing information, and hence, it incurs low overhead and little energy drain. Finally, the non-existence of explicit control packets or GPS information renders it immune to a wide-range of control and GPS attacks respectively.

III. The VINE Protocol

VINE is a subnet-layer protocol (below the IP layer) that efficiently delivers an application-layer message to the specified destination.

VINE learns routes by inspecting packet headers without using any control packets. It opportunistically sets up *gradient state* at each node determining the next hop and the cost for reaching a destination node. The header fields in every received packet[1] are utilized by every node to create gradient state for the source (origin) of the packet, allowing the node to forward packets to the source along a reverse-path *sink-tree* rooted at the source. State is also created for nodes within a 2-hop neighborhood.

When gradient state for a destination is not available, or it is deemed outdated, the packet is *broadcast*, that is, all recipients are targets. When state is available, the packet is *unicast*, that is, only the specified next hop processes the packet. This decision is taken independently at every hop. So the forwarding of a packet may alternate between flooding (in regions where no state exists) and unicast forwarding (in regions where state exists) en route to its destination. At network start-up time, when no state is available, the packets are flooded, but as traffic flows, nodes quickly create states and the need for flooding packets rapidly reduces (see Section IV). At the same time, under highly dynamic topology when nodes cannot keep up with the changes, VINE continues to provide a high delivery ratio by flooding packets.

Since many applications require a delivery notification, VINE incorporates an End-to-End Acknowledgment (E2E-A) scheme. Alternately, VINE could leverage a similar Transport Layer function. The E2E-A is handled like any other packet, and is also used to build gradients. In particular, the E2E-A allows a source to limit itself to a single flood per destination.

We first describe the procedure for building gradients, and then describe how the gradients are used for forwarding.

A. Gradient Establishment

A node maintains a list of gradients, one for each destination. Each entry includes the *destination*, *next hop*, *cost*, and *timestamp*. Currently, *cost* is the number of hops to the destination, but it can be generalized to any metric; *next hop* is the neighbor node to which the packet should be forwarded; and *timestamp* is the update time. If an entry does not get updated for a period (*GradientStateExpiry*), then it is deemed *expired* and not used. A single lowest cost entry is maintained for a given *destination* and *next hop* pair, but there can potentially be an entry for every *destination* and *next hop* pair. To limit gradient entries, each node maintains only a fixed number per destination (*MaxGradsPerDest*).

VINE packets contain the following header fields:

- *source*: The packet originator.
- *destination*: The packet destination.
- *sender*: The node forwarding the packet (same as the source if the sender is the originator).
- *prevSender*: The node that the sender first received the packet from (same as the source if sender is originator).
- *targetReceiver*: The intended next hop.
- *seqNum*: A sequence number unique at the source.
- *costFromSource*: the number of hops (cost) to the source.
- *ttl*: time-to-live (maximum forwarding count).

[1]Going forward a packet refers to a data packet unless stated otherwise.

On receiving a packet, the node updates (or creates) a 1-hop gradient towards the sender. In addition to that, the node creates a 2-hop gradient towards the previous sender with the sender as the next hop; and a k-hop gradient towards the source with the sender as the next hop, where k is the *costFromSource* field in the header. In the case of duplicate information (e.g., the source is the sender), only one gradient is created.

B. Packet Forwarding

A node updates all necessary header fields before forwarding the packet. In particular, the node makes the sender as the previous sender, itself as the sender, and increments the cost by one. Then, the node searches the gradient table for a suitable next hop having the minimum cost to the destination with ties broken according to whether the next hop was already unsuccessfully used or *timestamp*. We note that this is but one of several possible heuristics for selecting the next hop. If no gradient entry is available for the destination, or all available entries are deemed expired based on the *timestamp*, then the node broadcasts the packet, else it uses a minimum cost gradient.

All VINE packets are sent using MAC-level broadcasts. Thus, unlike WiFi MAC for example, no RTS, CTS or ACK packets are introduced, further eliminating control packets. Instead, VINE provides per-hop reliability based on overhearing the transmission of the next hop, considering it as an *implicit acknowledgment (IA)* and retransmitting if necessary[2]. A retransmission is scheduled using an IA timer only for the unicast packets. In the case of retransmission, the node attempts to select another min cost gradient, but if not available, then the same one is used, and the IA timer is reset. The *MaxRetransmissions* parameter determines the maximum attempts before the broadcast.

VINE also uses *end-to-end acknowledgments (E2E-A)* sent by the destination to the source, expressing a successful delivery of the packet. An E2E-A prevents nodes (e.g., last hop) from retransmitting in case of no IA. The E2E-A also allows the source and the intermediate nodes to learn gradients towards the destination. Nodes forward an E2E-A in the same way as any other data packet.

C. Discussion: Flooding, Control Packets, Security

The use of flooding as an initial or default mode may raise concerns about an excessive load. However, VINE achieves a natural balance, flooding a packet only when there is no or expired gradient, but that implies no or low traffic, so flooding can be accommodated. On the other hand, when traffic is high, nodes maintain up-to-date gradients, reducing the need for flooding. We show in Section IV that the load quickly stabilizes as a result of gradients learned from the previous sender information, IA, and E2E-A. Further, we note that traditional routing protocols also use flooding for disseminating control packets.

[2]We recognize that implicit acknowledgments may not work in networks using directional transmission, but given the near-ubiquity of omnidirectional antennas and the significant control savings, we have decided to employ it.

It may be argued that while VINE does not have dedicated control packets, there is control information in the header. However, we note that all of the fields except *prevSender* and *costFromSource* are present at some layer of any MANET stack – for instance, the MAC header typically contains the *sender* and the *targetReciever*. Further, *costFromSource* is strictly not necessary as it can be derived as (MAX_TTL - *ttl*) where MAX_TTL is initial value of *ttl*. Thus, the total impact is just the length of *prevSender*, which is 2 bytes ($\sim$2%) in our goTenna implementation. Moreover, it does not incur the per-packet MAC- and PHY-layer header and MAC contention penalties that dedicated additional control packets do. These points are confirmed by the simulation results in Section V.

As discussed in [5], a variety of attacks such as blackhole, link withholding, spoofing and wormhole attacks – all of which exploit and manipulate control packets (e.g RREQ in AODV) – can easily disrupt existing MANET protocols. Although researchers have proposed solutions to these attacks (e.g., [24], [25]), all of them introduce significant system complexity and have other drawbacks [5]. VINE does not have any dedicated control packets, and hence these attacks cannot be launched as-is. While it is conceivable that the attack ideas may be adapted to work on the control information in the VINE header, this can be likely be countered by commonplace network encryption/authentication (similar to IPSec).

IV. Communication Complexity Analysis

We now analyze the communication complexity (CC) of VINE. We define CC as the total bytes transmitted in the network for a packet originated at source S.

Let N and d be the network size and the network diameter, respectively. The message generated (payload) at the node is considered to be of size B bytes. The VINE header added to the packet is of size b bytes. The E2E-A acknowledgment packet sent by destination node D is of size E bytes.

The packet originated at node S is unicast to the next hop, if node S has a valid gradient to node D, otherwise, the packet is broadcast. Node S has a valid gradient to node D if a packet was sent to node D in the last T seconds and its E2E-A was received. Here, T is the *GradientStateExpiry* period. We consider all transmissions to be error-free.

We assume that node S generates packets for the destinations selected uniformly at random. Thus, the probability of not generating a packet for a particular destination is $(N-2)/(N-1)$.

Let R be the packet generation rate. If node D is the destination of a packet generated at node S, the probability of broadcast is at most the probability of not sending any of the previous RT packets to node D. Thus, the probability of sending n^{th} packet as a broadcast is:

$$p_n \leq \left(\frac{N-2}{N-1}\right)^{min(RT,n-1)} \tag{1}$$

The exponent uses min to capture the fact that if T is very high, say infinity, then the probability of broadcast depends on the previous $n-1$ packets, else it depends on the previous

RT packets. None of these previous RT packets must be sent to node D for the n^{th} packet to be broadcast. From Equation 1, as $RT \rightarrow \infty$, $p_n \rightarrow 0$. However, for practical values of RT, the probability of broadcast will be infinitesimally small.

$$CC_n \leq p_n[(N-1)(B+b)+dE]+(1-p_n)[d(B+b)+dE] \quad (2)$$

The first term of Equation 2 represents the worst case complexity of broadcast. Here we assume that the packet is flooded. When the packet reaches node D, all the intermediate nodes and node D will have routes to node S. So E2E-A will be forwarded maximum d times (since d is the network diameter).

The second term represents the complexity when node S has a valid gradient to node D. If node S has a valid gradient, then all the intermediate nodes between S and D would also have valid gradients because the E2E-A of the previous packet would have updated them. Thus, the packet from node S will be forwarded maximum d times, and so will be the E2E-A.

We compare the average CC obtained from Equation 2 to the average CC of an ns3 experiment, in which a single node sends packets to destinations selected uniformly at random in a 6x6 Manhattan grid. The experiment uses wireless links working in ad hoc mode and susceptible to interference and packet collisions. So, unlike the theoretical analysis, the nodes may retransmit packets based on the explanation in Section III. The average CC from Equation 2 (Numerical) is calculated using the ns3 experiment values (e.g., network size, payload). A random variable selects the destination and accordingly determines whether the node unicasts or broadcasts the packet.

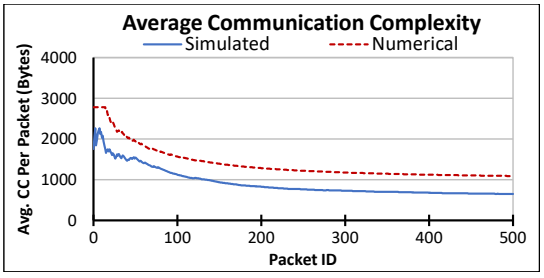


Fig. 1. Average communication complexity

Figure 1 shows the average CC of the first 500 packets. The simulated CC is lower than the numerical CC because, in addition to the source and the sender, nodes also learn gradients towards the previous sender, reducing the need for flooding. The average CC is high for the first few packets due to initial flooding, but it quickly converges as nodes start unicasting packets. Although a certain amount of traffic churn is required to maintain state, the "sweet spot" is very low – e.g., in a 6x6 grid, with T=1 min, VINE attains lower CC than Flooding for $R >= 4$ packets/min. In this experiment, only a single node generates packets, but when all nodes start generating, then the average CC would converge quicker.

V. Simulation Results

We have implemented and evaluated VINE in ns3 and compared the results to that of AODV. Table I lists the scenarios used for evaluation. In these scenarios, nodes transmit at 25 Kbps data rate to model low-capacity networks. We have also evaluated VINE at a higher data rate by repeating the increasing size experiments and configuring nodes to transmit at 1 Mbps data rate. Table II lists all the simulation parameters.

TABLE I
Network Scenarios

Network Scenario	Size (nodes)	Density (nodes/km^2)	Data Packet Interval (seconds)
Increasing Size	[10, 50]	3.3	30
Increasing Density	30	[0.83, 7.5]	30
Increasing Load	30	3.3	[10, 30]

We have found that AODV's performance improves in our settings if we use large Hello intervals and other associated parameters (shown in Table II). So, we included AODV results for two different Hello intervals: 1 (RFC [3] and ns3 recommended) and 30 seconds; and refer to them in the graph plots and the description as AODV-default and AODV-modified, respectively.

TABLE II
Simulation Parameters

Parameter	Value	Parameter	Value
Simulation Time	60 minutes	Node Speed	4 m/s
Data Packet Size	50 Bytes	Node Mobility	Rand. Waypoint
Propagation Loss	Friis Model	MAC	802.11b
VINE			
GradientStateExpiry	60 secs	IATimer	0.5 secs
MaxRetransmissions	2	MaxGradsPerDest	2 secs
AODV-default		**AODV-modified**	
Hello Interval	1 sec	Hello Interval	30 secs
Node Traversal Time	40 ms	Node Traversal	0.25 secs
Next Hop Wait	50 ms	Next Hop Wait	0.25 secs
Active Route	3 secs	Active Route	90 secs
MyRoute Timeout	6 secs	MyRoute Timeout	180 secs

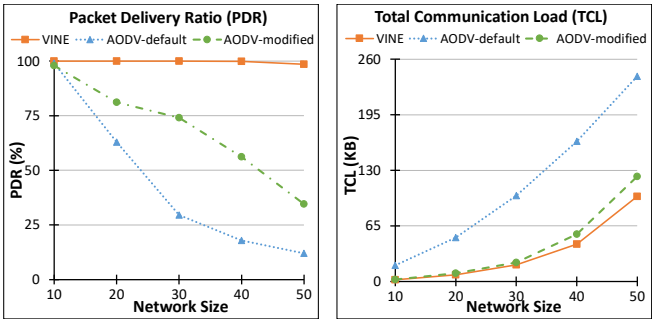
The simulation results are compared using the following two metrics: (1) Packet Delivery Ratio (PDR) is the ratio of the total data packets received and transmitted; and (2) Total Communication Load (TCL) is the total bytes transmitted per minute, comprising of the data packet headers, E2E-A (in VINE), control packets (in AODV), and the payload sizes.

All numerical comparisons below are with the better-performing AODV-modified unless mentioned otherwise.

A. Increasing Network Size

Figure 2(a) shows VINE having a significantly higher PDR than AODV for all network sizes. In fact, the PDR is close to 100% for all network sizes and ~2.5x better than AODV for size 50 nodes. AODV's PDR is remarkably low at all but very small sizes showing its inadequacy over low data rates. Nodes transmitting at low data rates experience long transmission delays, making the network susceptible to packet losses due to increased congestion, interference, and collision. AODV suffers the most due to its extensive use of route discovery, route maintenance, and local connectivity procedures. The network that was already congested and experienced a packet loss becomes overwhelmed with the control packets, resulting in a considerably low overall PDR. VINE uses data packets for learning routes, so the absence of control packets reduces

the possibility of collision. Although a route's absence or expiry results in broadcast, the resulting flooding updates routes in several nodes. The E2E-A feature unique to VINE also helps update routes frequently, making VINE resilient against dynamic topology changes and achieve a high PDR.



(a) Packet Delivery Ratio (b) Total Communication Load

Fig. 2. Simulation results for the increasing network size scenario where the size ranges from 10 to 50 nodes but the density remains constant.

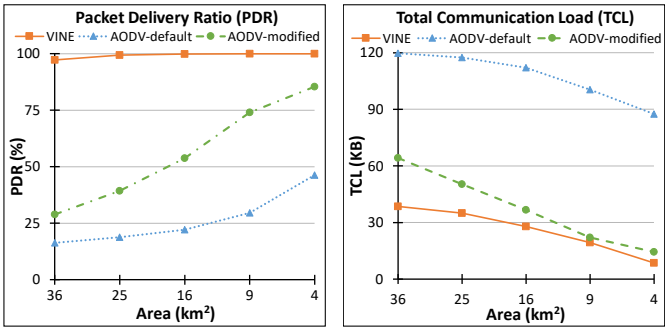
Despite flooding packets when necessary and using E2E-As, the TCL in VINE is ∼1.2x better (lower) than in AODV (shown in Figure 2(b)), proving that the sizes of extra fields in the header and the E2E-As are insignificant compared to the control packet sizes. AODV's control packets overwhelm the network and occupy a majority of its TCL.

B. Increasing Network Density

Figure 3(a) shows VINE having ∼3x and ∼1.2x better PDR than AODV in sparse (36 km^2) and dense (4 km^2) networks, respectively. In dense networks, the nodes are in close proximity to each other and frequently update their routes. The absence of control packets in VINE keeps the interference low. Sparse networks are more susceptible to link breaks, but VINE benefits from network layer retransmission, either to a different neighbor or broadcast, ensuring resiliency and a high PDR. Moreover, the previous sender information, IA, and E2E-A help frequently update the routes and minimize the need for flooding packets.

AODV's low PDR in sparse networks is due to two reasons. First, unlike VINE, there is no network layer retransmission to another neighbor, so a transmission over a broken link always results in a packet loss. Secondly, every packet loss triggers the route maintenance procedure, followed by the route discovery procedure, increasing interference for other ongoing transmissions. Although the local connectivity procedure detects link breaks, it takes two failed Hello messages to confirm one [3]. The PDR improves in the denser networks because network experiences fewer link breaks, and hence, less frequent route discovery and route maintenance instantiations.

Figure 3(b) shows TCL reducing for both AODV and VINE with the increasing network density because packets get forwarded fewer times. Also, fewer link breaks reduce route maintenance procedure calls in AODV and flooding in VINE. In sparse networks, VINE has ∼1.6x better (lower) TCL than AODV, but the gap narrows in dense networks.

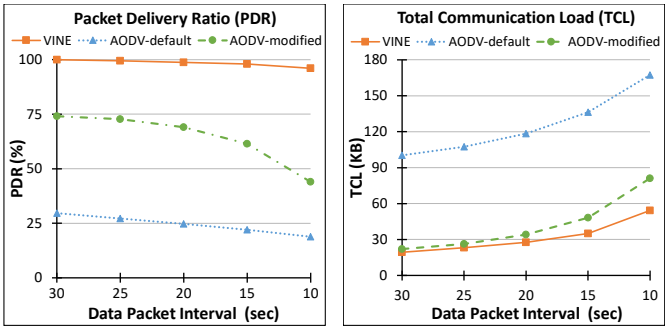


(a) Packet Delivery Ratio (b) Total Communication Load

Fig. 3. Simulation results for the increasing density scenario where the network size is 30 nodes but density ranges from 0.83 to 7.5 nodes/km^2 (i.e., simulation area ranges from 36 to 4 km^2).

C. Increasing Network Load

Increasing the network load (i.e., decreasing the data packet interval) is likely to increase packet losses due to growing interference and collisions. However, VINE's zero-control characteristic helps achieve up to 2x better PDR than AODV (shown in Figure 4(a)). On the other hand, AODV experiences a steep drop in the PDR because of its vicious circle, in which every packet loss due to a collision triggers the route maintenance and route discovery procedures, increasing the control packets and interference even further.



(a) Packet Delivery Ratio (b) Total Communication Load

Fig. 4. Simulation results for the increasing network load scenario where the network size is 30 nodes but packet interval ranges from 10 to 30 secs.

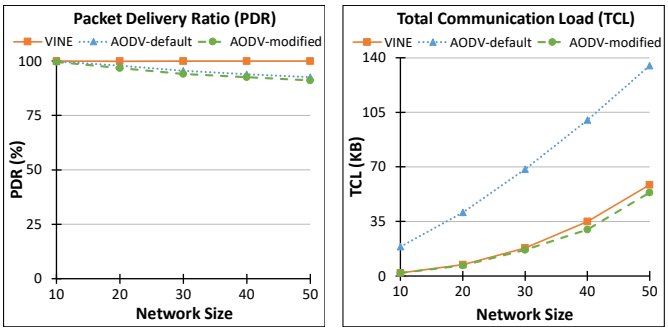
Despite the decreasing PDR and the same number of Hello messages, AODV's TCL increases (shown in Figure 4(b)), proving that the route maintenance and route discovery procedures get triggered frequently. VINE's TCL is up to 1.5x lower than that of AODV. It increases with the load because more packets get forwarded at delivered.

D. Increasing Network Size (1 Mbps Data Rate)

We now investigate if VINE's significant performance advantage extends to higher data rates, as might be expected in WiFi-based networks, by repeating the increasing network size experiments with a data rate of 1 Mbps. Figure 5(a) shows that both AODV and VINE have high PDRs, but VINE has 10% better PDR than AODV. The high PDR indicates that there is little interference at high data rates and the given network loads. Similar to the previous results, VINE has nearly 100%

PDR for all network sizes, whereas AODV's PDR reduces marginally in large networks. The fact that AODV-default and AODV-modified have similar PDRs proves that the 30 seconds Hello interval is suitable for all considered network scenarios and that AODV-modified provides a fair comparison to VINE.

Despite having similar PDRs, there is a significant difference between the TCLs of AODV-default and AODV-modified (shown in Figure 5(b)). The difference is mainly due to 30x more Hello messages sent by AODV-default. On the other hand, VINE's TCL is similar to that of AODV-modified because VINE leverages flooding for ensuring a high PDR at the expense of a few extra data packets. The overhead of these packets reflects in the TCL results, but it is significantly lower compared to that of AODV-default.



(a) Packet Delivery Ratio (b) Total Communication Load

Fig. 5. Simulation results for the increasing network size where nodes are configured to transmit at 1 Mbps data rate.

VINE outperforms AODV in all scenarios, ensuring versatility, reliability, and resilience against dynamic topology changes, proving that it is a better protocol across the board.

VI. Conclusions

Existing routing protocols such as AODV and OLSR for multi-hop wireless networks are universally based on disseminating explicit control packets. In low-capacity low-traffic MANETs such as those based on NBWF [1] and LoRA [2], the use of control-packet-based protocols such as AODV causes premature congestion collapse as described in Section V.

We have presented a novel routing protocol called VINE that does not use dedicated control packets. Instead, routes are progressively built and refined based on observations of data packets. Our study shows that while VINE floods the first few packets, it rapidly builds sufficient states to enable reverse-path forwarding and stabilizes the communication complexity. Our simulation results across a wide range of network size, density, traffic load over low capacity networks show that VINE is substantially more reliable than AODV (e.g., PDR of 98% vs AODV's 40% for 50 nodes). The Total Communication Load (TCL), which is a rough measure of battery energy consumption, is also lower for VINE.

VINE has been implemented on the goTenna Pro mesh networking device, which is currently being used for emergency and public safety. Its applicability extends more generally to any MANET as a reliable and scalable protocol that is not vulnerable to control packet attacks. Future work includes generalizing the hop metric to a cost vector, adapting the expiry time in accordance with the system dynamics, further reducing the amount of flooding while maintaining reliability, and studying security attacks and solutions.

References

[1] "North Atlantic Treaty Organization (NATO) Standardization Aggrement (STANAG) 5631/AComP-5631, Narrowband Waveform Physical Layer, Ratification Draft, Edition 1," 2015.

[2] LoRaWAN Certified Products. https://lora-alliance.org/. [Apr-2019].

[3] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Jul. 2003, RFC 3561.

[4] T. H. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," https://tools.ietf.org/html/rfc3626, Oct. 2003, RFC 3626.

[5] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, and A. Jamalipour, "A survey of routing attacks in mobile ad hoc networks," *IEEE Wireless Commun.*, vol. 14, no. 5, pp. 85–91, Dec. 2007.

[6] T. Winter *et al.*, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," https://tools.ietf.org/html/rfc6550, Mar. 2012, RFC 6550.

[7] M. Vučinić, B. Tourancheau, and A. Duda, "Performance comparison of the RPL and LOADng routing protocols in a Home Automation scenario," in *WCNC 2013*, Shanghai, China, Apr. 2013, pp. 1974–1979.

[8] G. R. Hiertz *et al.*, "IEEE 802.11s: The WLAN mesh standard," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 104–111, 2010.

[9] goTenna Pro. https://gotennapro.com/products/gotenna-pro. [Apr-2019].

[10] R.Ramanathan, C.Servaes, W.Ramanathan, A.Dusia, and A.Sethi, "Long-range short-burst mobile mesh networking: Architecture and evaluation," in *Proc. IEEE SECON*, 2019.

[11] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking*, 2001, pp. 139–172.

[12] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, Oct. 1994.

[13] A. Boukerche, S. K. Das, and A. Fabbri, "Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks," *J. Parallel Distrib. Comput.*, vol. 61, no. 7, pp. 967–995, Jul. 2001.

[14] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Proc. Int. Conf. on Universal Personal Communications*, vol. 2, San Diego, CA, USA, Oct. 1997, pp. 562–566.

[15] Y.-B. Ko and N. H. Vaidya, "Location-aided Routing (LAR) in Mobile Ad Hoc Networks," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw.*, Dallas, Texas, USA, Oct. 1998, pp. 66–75.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. Mobile Comput. Netw.*, Boston, MA, USA, Aug. 2000, pp. 56–67.

[17] Robert D. Poor, "Gradient Routing in Ad Hoc Networks," https://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf, 2000, MIT Media Laboratory.

[18] D. Lundell, A. Hedberg, C. Nyberg, and E. Fitzgerald, "A Routing Protocol for LoRA Mesh Networks," in *Proc. 19th Int. Symp. WoWMoM*, Chania, Greece, Jun. 2018, pp. 14–19.

[19] M. Zhao, I. W. Ho, and P. H. J. Chong, "An Energy-Efficient Region-Based RPL Routing Protocol for Low-Power and Lossy Networks," *Internet of Things Journal*, vol. 3, no. 6, pp. 1319–1333, Dec. 2016.

[20] E. Baccelli, M. Philipp, and M. Goyal, "The P2P-RPL routing protocol for IPv6 sensor networks: Testbed experiments," in *Proc. SoftCOM*, Split, Croatia, Sep. 2011, pp. 1–6.

[21] L. Li *et al.*, "Networking for next generation NBWF radios," in *Proc. MILCOM 2015*, Oct. 2015, pp. 121–126.

[22] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. INFOCOM 2006*, Barcelona, Spain, Apr. 2006, pp. 1–11.

[23] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks," Aug. 2012, RFC 6693.

[24] S. Desilva and R. V. Boppana, "Mitigating malicious control packet floods in ad hoc networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, vol. 4, New Orleans, LA, USA, Mar. 2005, pp. 2112–2117.

[25] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE J. Selected Areas in Commun.*, vol. 24, no. 2, pp. 370–380, Feb. 2006.