Proc. IEEE Military Communication Conf., MILCOM, Baltimore, MD, Oct. 23-25, 2017, pp. 103-108

Multilayer MANET Routing with Social-Cognitive Learning

Yalin E. Sagduyu*, Yi Shi*, Tugba Erpek*, Sohraab Soltani*, Sharon J. Mackey†, Derya H. Cansever†, Mitesh P. Patel†, Bart F. Panettieri†, Boleslaw K. Szymanski‡, and Guohong Cao§
*Intelligent Automation, Inc., Rockville, MD 20855, USA, †U.S. Army CERDEC, APG, MD 21005, USA,
‡Rensselaer Polytechnic Institute, Troy, NY 12189, USA, §Pennsylvania State University, University Park, PA 16802, USA
Email: {ysagduyu, yshi, terpek, ssoltani}@i-a-i.com, {sharon.j.mackey.civ, derya.h.cansever.civ, mitesh.p.patel.civ, bart.f.panettieri.civ}@mail.mil, szymab@rpi.edu, gcao@cse.psu.edu

***Abstract*—This paper studies the problem of routing in a multilayer (communication and social) network. Network protocols, such as link state routing and its variants, heavily used in mobile ad hoc networks (MANETs) cannot sustain robustness and efficiency as the topological information becomes easily stale with fast network dynamics. Attempts to collect and exchange excessive network information would result in significant overhead and would degrade the overall network performance. This paper presents the SCATE (Social-Cognitive Advancement at Tactical Edge) routing protocol that applies social-cognitive techniques to improve robustness and efficiency of a multilayer network with MANET communication and social links. In a distributed and decentralized setting with local information, nodes learn and update their distances to destinations using social-cognitive metrics and make routing decisions to minimize the end-to-end delay. The SCATE protocol is compared with Optimized Link State Routing (OLSR) with and without social links. Stand-alone computer simulations and high fidelity simulation/emulation tests with CORE and EMANE are used to evaluate SCATE under different communication network (traffic and mobility) and social network effects. Results show that the SCATE protocol is a viable solution to MANET routing by substantially reducing the overhead and the end-to-end delay, and increasing the end-to-end delivery ratio for both unicast and multicast traffic.**

***Index Terms*—MANET; routing; multilayer network; communication network; social network; social-cognitive metrics; delay; delivery ratio; overhead; CORE; EMANE; unicast; multicast.**

I. Introduction

Multiple networks may interact with each other and thus form a *multilayer interdependent network* [1], [2]. In this paper, we consider a *communication* network and a *social* network, where routing through the combined social and communication network structures along with social-cognitive learning has the potential to improve the overall network performance such as end-to-end delay. Robust and efficient routing protocols are needed for a mobile ad hoc network (MANET), where the network topology changes quickly and the overhead incurred to track these changes is excessive. *Social-aware routing* [3]–[5] leverages socially-connected user pairs to improve the network performance. This paradigm has been applied largely to delay-tolerant networks (DTNs), where the delay is not the primary concern and the mobility is heavily relied on for packets to reach destinations.

In this paper, a MANET environment is considered with a distributed and decentralized setting, where the goal is to minimize the delay without any explicit reliance on mobility (i.e., the DTN concept does not apply). The *SCATE (Social-Cognitive Advancement at Tactical Edge) routing protocol* is introduced to utilize relationships/interactions across social and communication network dimensions that combine the information that is known *a priori* with the *local information* that is learned on-the-fly. The SCATE protocol is designed to route both *unicast* and *multicast* traffic, and uses *social-cognitive metrics* (e.g., frequency of encounters, social pressure metric [4], and frequency of interactions) to learn the distances to the destinations. Using these estimates, each node individually selects the next hop with the shortest distance (normalized with link delay) to the destination of its packet traffic.

As an extension to routing protocols (e.g., [6]) for a single-layer network, this paper considers a *multilayer network* that consists of MANET communication links and additional social links that are enabled by other communication means such as high power, (aerial) relay (e.g., UAV), and directional transmissions to extend the transmission range. Routing has been studied in a combined communication-social network in [7]–[10] without online learning. The SCATE protocol supports nodes with *social-cognitive learning* to reliably determine routing metrics in a multilayer network, where nodes can choose each link (for transmitting and forwarding data in the routes to their destinations) from a MANET communication link or social link (social links are enabled by using higher high power, relay, or directional transmissions). SCATE operates with local information in a distributed/decentralized setting without any centralized controller.

The SCATE protocol is compared with Optimized Link State Routing (OLSR) [11] and OLSR-SL (the proposed extension of OLSR with social links) in extensive simulations with various traffic (unicast or multicast), mobility (random or group) and social network models (simulated or real data). Evaluation is extended to *Common Open Research Emulator (CORE)* [12] simulations (also referred to as emulations) using *Extendable Mobile Ad-hoc Network Emulator (EMANE)* [13] models for PHY and MAC. Results show that the SCATE

This work was partially supported by the ARL under CA No. W911NF-09-2-0053 (the ARL NS CTA).

protocol significantly reduces the overhead and the end-to-end delay, and increases the end-to-end delivery ratio compared to OLSR and OLSR-SL.

The contributions of this paper are summarized as follows:

1) An introduction of a novel routing protocol, SCATE, developed with social-cognitive learning to minimize the delay in a multilayer network.
2) The application of SCATE to unicast and multicast traffic.
3) The demonstration via extensive simulations (including CORE and EMANE) that SCATE achieves major gains relative to OLSR and OLSR-SL.

The rest of the paper is organized as follows. Section II presents the network model and the SCATE routing protocol. Section III extends the SCATE routing to multicast traffic. Section IV describes the simulation setting. Section V presents the simulation results and compares SCATE with OLSR and OSLR-SL. Section VI extends the simulations to CORE and EMANE. Section VII concludes the paper.

II. Network Model and SCATE Routing

We consider a multilayer network that consists of *wireless MANET communication* and *social links*. In the underlying social network, a predefined social distance d_{uv}^S is assigned between any two nodes u and v. Social links are enabled via different communication means, such as higher transmit power, relay (e.g., UAV) or directional transmissions, to extend the range of socially-connected transmitter-receiver pairs. In SCATE routing, each node uses social-cognitive metrics to estimate the distance from the destination and computes the routing metric that includes the link delay and the decrease in the estimated distance to the destination. Then, each node selects the next-hop node to minimize this routing metric.

Assuming a slotted time, node u computes $\hat{d}_{uv}(t)$ as an estimation on the communication distance $d_{u,v}(t)$ to another node v at any time (slot) t. In *unicast* traffic with one destination per packet, each node u selects the next hop neighbor v to minimize the expected end-to-end delay to destination $\mathcal{D}$. At time t, node u selects the next-hop node $v^*(t)$ as

$$v^*(t) = \underset{v}{\operatorname{argmin}} \left\{ \frac{D_{uv}}{\hat{d}_{u\mathcal{D}}(t) - \hat{d}_{v\mathcal{D}}(t)} \right\}, \tag{1}$$

where $\hat{d}_{u\mathcal{D}}(t) - \hat{d}_{v\mathcal{D}}(t)$ is the reduction in estimated distance to $\mathcal{D}$ and D_{uv} is the delay of using link from node u to v.

Node u estimates the communication distance $d_{uv}(t)$ to another node v as a general function f of d_{uv}^S and some social-cognitive distance $s_{uv}(t)$ that is learned on-the-fly $s_{uv}(t)$. The estimated distance is given by

$$\hat{d}_{uv}(t) = f(d_{uv}^S, s_{uv}(t)). \tag{2}$$

For numerical results, we specialize (2) to

$$\hat{d}_{uv}(t) = \alpha d_{uv}^S + (1-\alpha) s_{uv}(t), \tag{3}$$

where the weight α satisfies $0 \leq \alpha \leq 1$. Each node u computes $s_{uv}(t)$ with local information. We consider three social-cognitive metrics $m_{uv}(t)$ to compute $s_{uv}(t)$:

1) *Frequency of encounters (FE)*: percentage of time (measured at time t) nodes u and v are within their communication range r_C, i.e.,

$$m_{uv}(t) = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{1}(d_{uv}(\tau) \leq r_C), \tag{4}$$

where $\mathbf{1}(e)$ is the indicator function (equal to 1 if event e happens, and 0 otherwise). $s_{uv}(t)$ is inversely proportional to $m_{uv}(t)$.

2) *Social pressure metric (SPM)* [4]: remaining time (measured at time t) for node u to encounter node v, i.e.,

$$m_{uv}(t) = \frac{1}{t} \sum_{\tau=1}^{t} f_{uv}(\tau), \tag{5}$$

where $f_{u,v}(\tau)$ is the time remaining to the next encounter of u and v at time τ (note that $f_{u,v}(\tau) = 0$ if u and v meet at time τ). $m_{uv}(t)$ was calculated in [4]. In this formulation $s_{uv}(t)$ is proportional to $m_{uv}(t)$.

3) *Frequency of interactions (FI)*: percentage of time (measured at time t) when node u transmits to node v, i.e.,

$$m_{uv}(t) = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{1}(u \text{ transmits to } v \text{ at time } t). \tag{6}$$

$s_{uv}(t)$ is inversely proportional to $m_{uv}(t)$.

In Eqs. (4)–(6), the average can also be computed over $[t-w, t]$ for a suitable window size w to consider the recent status only. At time $t+1$, each node u updates s_{uv} based on $s_{rv}(t)$ from every node r in the discovered neighbor set $\mathcal{N}_u(t)$:

$$s_{uv}(t+1) = \min\left\{ s_{uv}(t), d_{ur}^S + s_{rv}(t), \forall r \in \mathcal{N}_u(t) \right\}. \tag{7}$$

The steps of the SCATE protocol at time t are given by

1) Any node u computes $s_{uv}(t)$ for any other node v with a social cognitive metric, FE (Eq. (4)), SPM (Eq. (5)) or FI (Eq. 6), and updates $\hat{d}_{uv}(t)$ (Eq. (3)).
2) Any node u broadcasts HELLO packets with s_{uv}, collects $s_{rv}(t)$ from $r \in \mathcal{N}_u(t)$, and updates $\mathcal{N}_u(t)$.
3) Any node u updates $s_{uv}(t)$ with Eq. (7).
4) Any node u decides on the next hop $v^*(t)$ with Eq. (1).

This formulation for delay minimization can be also extended to maximizing the end-to-end delivery ratio.[1]

III. Extension to Multicast Routing

There is one destination of each source S in *unicast* traffic. In the *multicast* case, traffic of a source S is destined to multiple destinations $D_1, D_2, \cdots, D_M$. Then, the routing problem is to find a multicast tree T that is rooted at node S and includes all destinations. Each leaf node of the multicast tree must be a destination, whereas a destination may be a leaf node or a non-leaf node in the tree. The latter case means that this destination needs to forward data for other destinations.

[1]If p_{uv} is the delivery ratio on a link, $p(\mathcal{P}) = \prod_{(u,v)\in\mathcal{P}} p_{uv}$ is the end-to-end delivery ratio on path $\mathcal{P}$. Taking the logarithm of both sides, the same additive structure of delay optimization follows such that Eq. (1) is replaced with $v^*(t) = \operatorname{argmax}_v \left\{ \frac{\log(p_{uv})}{\hat{d}_{u\mathcal{D}}(t) - \hat{d}_{v\mathcal{D}}(t)} \right\}$.

We define the performance metric γ_n (e.g., delay d_n) for each $(S, \mathcal{D}_n)$ of M pairs. The performance metric is $\gamma = h(\gamma_1, \gamma_2, \ldots, \gamma_M)$ for the multicast tree, where $h(\cdot)$ is monotone for each γ_n. For example, the average delay $d = \frac{1}{M}\sum_{n=1}^{M} d_n$. Such a multicast performance metric can be regarded as an extension of unicast performance metric. We extend unicast routing of SCATE to multicast as follows.

1) For each destination $\mathcal{D}_n$, apply unicast routing to obtain a path from S to $\mathcal{D}_n$.
2) Merge paths for all destinations to obtain a multicast tree.

For step 2, it is possible that some paths share common nodes. Assuming a centralized approach (extended later to a distributed protocol) and a static network, Property 1 follows.

Property 1. *If the optimal path $\mathcal{P}_i$ from source S to destination $\mathcal{D}_i$ and the optimal path $\mathcal{P}_j$ from source S to destination $\mathcal{D}_j$ have a common node r, then the subpath $\mathcal{P}_i^r$ from source S to common node r of path $\mathcal{P}_i$ and the subpath $\mathcal{P}_j^r$ from source S to common node r of path $\mathcal{P}_j$ must have the same performance (or these two subpaths are identical).*

To prove Property 1, we need the following fact, which holds under various performance metrics, e.g., the average delay.

Fact 1. *If we replace a subpath by a better subpath, the new path is also better than the original path.*

Proof to Property 1. Suppose that $\mathcal{P}_i^r$ and $\mathcal{P}_j^r$ do not have the same performance. Without loss of generality, assume $\mathcal{P}_i^r$ is better. Replace $\mathcal{P}_j^r$ by $\mathcal{P}_i^r$ and obtain a new path from source S to destination $\mathcal{D}_j$. From Fact 1, this new path is better and this contradicts the optimality of the original path for $\mathcal{D}_j$. ■

From Property 1, in step 2, if some paths share a common node r, their subpaths from source S to r have the same performance. If these subpaths are not identical, any one of them can replace others without performance change. This leads to a tree instead of multiple paths. A parent node always transmits one copy to a child node independent of its offsprings. From the monotonicity of $h(\cdot)$, Theorem 1 follows.

Theorem 1. *If the optimal paths to each destination are combined to obtain a tree, this is the optimal multicast tree.*

Theorem 1 enables us to extend unicast routing to multicast.

Implementation issues. Next, we cast multicast routing as a *distributed protocol*. A multicast packet header includes a set of all destinations. Since a packet will be duplicated when a node has multiple children, one packet for a subset of destinations, a set of intended destinations is included in the header. Each node node chooses the next hop node for each intended destination and specifies intended destinations for each selected node. If a next hop node is selected for multiple intended destinations, only one packet is transmitted to this node. The distributed protocol incorporates the following.

- The first time a destination receives a packet, even if it is not an intended destination (due to mobility), it is counted as an end-to-end delivered packet and the delay is computed based on this delivery time. If there are other intended destinations, it needs to forward this packet.
- Under mobility, a node may receive a packet multiple times, each for a different set of intended destinations, and will forward this packet multiple times.

Based on these insights from combining unicast paths, the following *distributed multicast protocol* is developed:

1) A multicast packet header includes all destinations.
2) Each node i selects the best next-hop $N_i(\mathcal{D}_j)$ for each destination $\mathcal{D}_j$.
3) If next-hops $N_i(\mathcal{D}_j)$ overlap, node i transmits one packet to $N_i(\mathcal{D}_j)$ and specify multiple destinations in the header.
4) The first time a destination receives a packet, even if it is not an intended destination (due to mobility),
   a) it counts this packet as end-to-end delivered;
   b) it removes its ID as a potential destination from the packet header; and
   c) if there are other intended destinations, the current destination needs to forward the received packet.
5) If a node receives a packet multiple times (each for a different set of destinations), the node forwards this packet multiple times.

Multicast destinations may not be given or fixed, i.e., nodes may *subscribe to* or *unsubscribe from* a multicast. For such scenarios, there are the following steps:

1) A destination $\mathcal{D}_i$ subscribes/unsubscribes via a unicast with source being $\mathcal{D}_i$ and destination being S.
2) Multicast source S keeps a list of subscribed users, and
   a) adds a node if "subscribe" packet is received;
   b) removes a node if "unsubscribe" packet is received.
3) With a multicast packet, S uses the list of subscribers as destinations and initializes the multicast protocol.

IV. Simulation Setting

The following three routing protocols are compared in a discrete-event simulator written in Python.

1) *The SCATE routing*: A node selects the next-hop node according to Eq. (1).
2) *OLSR*: A node selects the communication link on the minimum delay path.
3) *OLSR-SL*: A node selects the communication link or social link (SL) on the minimum delay path.

There are N nodes and F source-destination pairs. Time is slotted and each slot corresponds to one packet transmission. Each source generates traffic with rate R_T packets per slot. The number of destinations for each packet flow is chosen uniformly randomly from 1 to M (where $M = 1$ for unicast).

A. MANET communication network

MANET communication network is generated by distributing nodes uniformly randomly in the network of area A and let them move at every T_C time slots according to either one of the following two models:

1) *Random waypoint mobility*: Each node individually makes a random movement with the same length l_I in a random direction.
2) *Group mobility*: Nodes may belong to different groups. Nodes in the same group first move along the same direction with the same length l_G. Then, each node makes an individual movement with a smaller length l_I.

TABLE I
SYSTEM PARAMETERS.

N, number of nodes	100
A, network area	$[0,1] \times [0,1]$
F, number of flows	3
R_T, packet generation rate	0.1 packet/slot
M, multicast group size	3
T_C, period of movements	5 time slots
r_C, MANET transmission range	0.1
D_C, MANET link delay	1 slot
l_I, length of individual movement	0.01
l_G, length of group movement	0.04
r_S, range for social links	0.1
D_S, social short range link delay	2 slots
D_L, social long range link delay	4 slots
n_L, number of long range social links	up to 4
$r_{C,S}$, extended transmission range	0.6
α, weight for social distance	0.5

The delay on a communication link is denoted by D_C (the queueing delay is separately considered). Each transmission is successful if the transmitter-receiver pair is within a fixed communication range r_C. This model captures the successful packet reception if the signal-to-noise-ratio (SNR) is greater than a fixed threshold. For channel access, TDMA is used to avoid collisions.

B. Social network

The underlying social network is either *simulated* or obtained from a *real data set*.

1) Social network is simulated according to the generative Octopus model [14]. Nodes in the communication network are randomly deployed on a disk with unit radius representing the social distances. A node has a social link to another node if the social distance between them is less than a range r_S (the link delay is D_S). In addition, each node has n_L social links to nodes outside the range r_S to generate the small-world effect (the link delay is D_L).
2) Real data set from a real-word experiment [15] is used to obtain the social network connections and the distances between users. Details are given in Sec. V-C.

There are various communication means to enable social links (by providing the transmission range extension denoted by $r_{S,C}$)) other than the default MANET communications:

1) Higher transmit power.
2) Relay (e.g., UAV) with both uplink and downlink.
3) Directional transmissions.

In this paper, we assume that nodes apply higher transmit power as means to enable long-range communications. Notation is summarized in Table I with default values.

V. COMPARISON OF SCATE, OLSR AND OLSR-SL

A. Performance Metrics and Configurations

Three performance metrics (all averaged over nodes and time for delivered packets) are evaluated.

1) *Overhead*: exchanged control information (kb).
2) *Delay*: end-to-end packet delay (slot).
3) *Delivery ratio*: end-to-end success rate.

TABLE II
COMPARISON OF SCATE, OLSR, AND OLSR-SL.

Configuration $\mathcal{C}_1$	Overhead	Delay	Delivery ratio
OLSR (Test 1)	6.35	2.62	0.31
OLSR (Test 2)	3.23	2.20	0.07
OLSR-SL (Test 1)	18.64	2.19	0.45
OLSR-SL (Test 2)	1.03	1.44	0.07
SCATE with FE	0.11	1.90	0.34
SCATE with SPM	0.12	1.65	0.40
SCATE with FI	0.04	0.99	0.03

Different system settings are considered:

1) *Social network*: simulated (from Octopus model [14]) (Ns) or real data (from [15]) (Nr).
2) *Traffic*: unicast (Tu) or multicast (Tm).
3) *Mobility*: random (Mr) or group (Mg).

B. Performance Evaluation

SCATE (with social-cognitive metrics of FE, SPM or FI), OLSR, and OLSR-SL are compared in Table II for configuration $\mathcal{C}_1 =$ (Ns, Tu, Mr). SCATE reduces the overhead and the delay, and increases the delivery ratio. SPM achieves a good performance for all metrics. The ratio of social links (among all links used in routing) is 0.32, 0.15, and 1.00 for FE, SPM and FI, respectively. Comparing with OLSR, the overhead of OLSR-SL increases excessively due to the use of social links.

OLSR and OLSR-SL incur high overhead, when the frequency of the HELLO messages (to sense links and detect neighbors) is set the same in OLSR, OLSR-SL and SCATE. We call this case as Test 1. Next, we run an additional Test 2, where the overhead of the OLSR and OLSR-SL protocols is made smaller by increasing the intervals of HELLO messages and topology control (TC) messages (to declare topology by advertising link states). The purpose of Test 2 is to translate the overhead gains to the delivery ratio. In Test 2, we reduce the OLSR overhead roughly to half by sending control packets less frequently. The same control packet frequency is used for OLSR-SL. As shown in Table II, the delivery ratio of OLSR drops from 0.31 (Test 1) to 0.07 (Test 2). If we further reduce control packets, very few data packets will be completed (due to the timeout mechanism on keeping neighbor information, a node unlikely finds a neighbor to transmit and holds packet) and the average overhead increases. OLSR-SL achieves better overhead gain but the delivery ratio drops excessively.

C. Performance Evaluation under Various Configurations

Next, we compare SCATE with OLSR and OLSR-SL under various configurations on network, communication and traffic models. Note that a comparison of results across configurations cannot provide meaningful insights due to different settings.

Real data set for social network and distance. In configuration (Ns), we simulated the social network with the Octopus model. Next, we use a real data set, the Reality Mining Data [15], to build the social network in configuration (Nr). Table III shows the performance under configuration $\mathcal{C}_2$ = (Nr, Tu, Mr), i.e., when real data set is used for social network, unicast traffic and random mobility are applied.

TABLE III
COMPARISON OF SCATE, OLSR, AND OLSR-SL FOR $\mathcal{C}_2$-$\mathcal{C}_4$.

	Overhead	Delay	Delivery ratio
Configuration $\mathcal{C}_2$ = (Nr, Tu, Mr)			
OLSR	5.13	2.32	0.35
OLSR-SL	14.33	0.32	0.07
SCATE with FE	0.14	2.14	0.39
SCATE with SPM	0.18	2.18	0.48
SCATE with FI	0.24	1.78	0.42
Configuration $\mathcal{C}_3$ = (Ns, Tm, Mr)			
OLSR	17.38	1.74	0.09
OLSR-SL	24.86	2.27	0.36
SCATE with FE	0.04	0.78	0.23
SCATE with SPM	0.07	0.88	0.17
SCATE with FI	0.02	1.01	0.18
Configuration $\mathcal{C}_4$ = (Ns, Tm, Mg)			
OLSR	18.23	1.31	0.10
OLSR-SL	19.92	1.19	0.44
SCATE with FE	0.08	0.83	0.16
SCATE with SPM	0.08	1.04	0.19
SCATE with FI	0.03	0.30	0.12

- *To generate a social network (or social links)*, we use the friend matrix that indicates whether two users are friends and the lab (or outlab) matrix that indicates the time two users closely stay in (or outside) lab. We add a social link between two nodes if they are friends, or closely stay in or outside lab for certain hours. As a result, we obtain a social graph with 93 users (nodes) and 666 links.
- *To generate social distance*, we use the survey data. Each user answers 25 questions. One question and its possible answers are as follows: Q: *Have you travelled recently?* A: *Very often - more than a week/month*; *Often - a week/month*; etc. We measure the difference between answers of two users on a question and map it to a value in $[0, \sqrt{2}]$. The average of differences between answers of the two users (aggregated on all questions) is defined as the social distance between these two users.

Multicast traffic. In configuration (Tu), we considered unicast traffic with one destination per source. Next, we consider multicast in (Tm) with multiple destinations per source and apply the multicast extension of routing protocols. Table III shows the performance under configuration $\mathcal{C}_3$ = (Ns, Tm, Mr), i.e., when simulated data set is used for social network, multicast traffic and random mobility are applied.

Group mobility model. In configuration (Mr), random waypoint mobility is used. Next, we use group mobility in (Mg). Table III shows the performance under configuration $\mathcal{C}_4$ = (Ns, Tm, Mg), i.e., when simulated data set is used for social network, multicast traffic and group mobility are applied.

In all these configurations, SCATE significantly reduces the overhead, reduces the end-to-end delay, and increases the end-to-end delivery rate relative to OLSR and OLSR-SL.

VI. EVALUATION WITH CORE AND EMANE

We compare SCATE and OLSR under CORE [12] simulations using EMANE [13] PHY/MAC models. CORE is an open-source software for building virtual networks by using the representation of a real computer network that runs abstract models in real time (such as generating and sending real packets among nodes). CORE runs with EMANE that emulates MAC and PHY layers. There are host machine, controller laptops (virtual) and radios (virtual) in our configuration. The simulation tests are run on the host machine. IP settings are configured in CORE for radios and controllers (laptops). Each radio has a dedicated controller. Initially, the test scenario is specified by determining the number of radios, their positions and mobility patterns and the propagation path loss model. Next, the software to be run on the host machine, controllers and radios are specified. The radios run the SCATE protocol and determine the next-hop radio. The controllers serve two purposes: 1) convey the throughput, delay and overhead statistics to the host machine, 2) act as a social link (Ethernet connection for this case) when the "distance" associated with sending the data through the RF communication link is larger compared to using the social link. The test results include dynamic updates for packet exchanges among radios and network statistics such as throughput and overhead. These results are displayed and recorded on the host machine.

SCATE protocol is implemented as Python modules simulating a stand-alone radio software that is associated with each virtual node in CORE. Fig. 1 shows the workflow of SCATE modules described below:

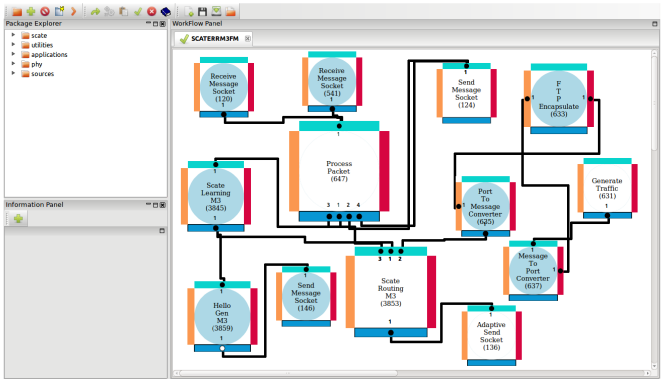

Fig. 1. The workflow of SCATE modules.

- `GenerateTraffic` module periodically generates data packets with a user specified rate.
- `FTPEncapsulate` module encapsulates data packets received from `GenerateTraffic` module with the header (source and destination IP addresses, packet type, etc.). The encapsulated packets are sent to `ScateRouting` module to determine the next hop.
- `ProcessPacket` module checks the type (HELLO or DATA) of each incoming packet. Each HELLO packet is forwarded to the `ScateLearning` module. The destination IP address of each DATA packet is checked. If the destination is this node, the throughput, delay and overhead statistics are calculated and sent to the host machine through controller using `SendMessageSocket` module. If the destination is a different node, the DATA

packet is forwarded to the `ScateRouting` module to determine the next hop.

- `ScateLearning` module updates the list of neighbor nodes and distance metrics based on the information received from the HELLO messages, and sends them to the `HelloGen` and `ScateRouting` modules.
- `ScateRouting` module determines the next-hop node with the minimum "distance" and forwards the DATA packet (from the `FTPEncapsulate` and `ProcessPacket` modules) to this node. Routing information is updated with the new neighbor lists and distance metrics from the `ScateLearning` module. The relay IP address is encapsulated in the packet and sent to the `AdaptiveSendSocket` module.
- `HelloGen` generates HELLO packets that includes the self IP address, the IP addresses of the neighbor nodes, social distances to neighbors, and the time stamp to enable the receiver node to calculate the communication delay.

OLSR is run in CORE using the OLSR daemon (`olsrd`) with the default setting (HELLO Interval = 6.0s). SCATE and OLSR are compared in CORE simulations using five nodes. Source-destination pairs are (1,3), (2,3), (3,1), (4,5), and (5,2). Traffic is generated at rate 1024 bps. In Test 1, the frequency of the HELLO packets (sent in every 6 sec) is set the same in OLSR and SCATE. In Test 2, the overhead of the two protocols is made similar by increasing the HELLO interval for OLSR. The snapshot of the topology, mobility, and simulation performance are shown in Fig. 2.

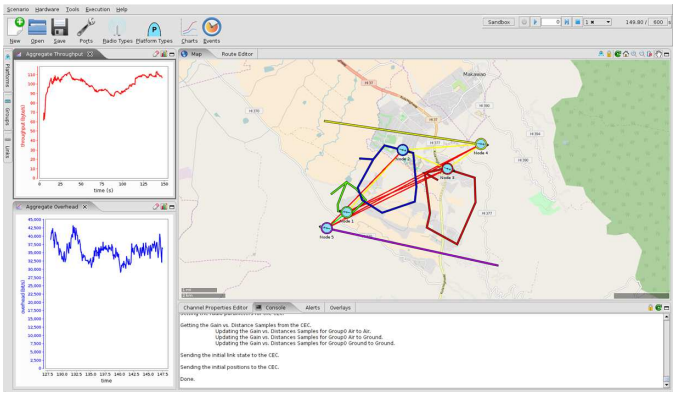

Fig. 2. Snapshot of the CORE simulation.

Table IV and Table V show the results under RF Pipe and 802.11a/b/g EMANE models, respectively. SCATE significantly outperforms OLSR under both EMANE models in terms of throughput and overhead. With reduced overhead in Test 2, OLSR cannot collect reliable information under RF Pipe EMANE model and its throughput drops, whereas OLSR finds slightly more opportunity for data transmission under 802.11 a/b/g EMANE model and its throughput slightly increases. Overall, SCATE significantly improves the overhead and the throughput relative to OLSR.

TABLE IV
CORE SIMULATIONS UNDER RF PIPE EMANE MODEL.

Protocol	Overhead (bps)	Throughput (bps)
SCATE with FE	4000 $\pm$ 200	1248 $\pm$ 56
OLSR (Test 1)	14000 $\pm$ 300	280 $\pm$ 16
OLSR (Test 2)	4000 $\pm$ 300	206 $\pm$ 7

TABLE V
CORE SIMULATIONS UNDER 802.11 EMANE MODEL.

Protocol	Overhead (bps)	Throughput (bps)
SCATE with FE	8400 $\pm$ 500	1920$\pm$ 320
OLSR (Test 1)	37000 $\pm$ 5000	800 $\pm$ 80
OLSR (Test 2)	8700 $\pm$ 400	912 $\pm$ 80

VII. CONCLUSION

We considered a multilayer network with MANET communication and social links, and designed the SCATE routing protocol that learns social-cognitive metrics in a distributed/decentralized network setting. We compared the performance (overhead, end-to-end delay and delivery ratio) of SCATE to OLSR (with and without social links) through stand-alone and CORE simulations with different EMANE models. Simulation results showed that compared to OLSR, the SCATE routing provides major benefits to MANET by significantly reducing the overhead and the end-to-end delay, and increasing the end-to-end delivery ratio.

REFERENCES

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic Cascade of Failures in Interdependent Networks," *Nature*, Apr. 2010.

[2] Y. Zhuang, O. Yagan, "Information propagation in clustered multilayer networks", *IEEE Trans. Network Science and Engineering*, Dec. 2016.

[3] W. Gao, Q. Li, B. Zhao, and G. Cao, "Social-aware Multicast in Disruption Tolerant Networks," *IEEE/ACM Trans.Networking*, Oct. 2012.

[4] E. Bulut and B. K. Szymanski, "Exploiting Friendship Relations for Efficient Routing in Mobile Social Networks," *IEEE Trans. Parallel and Distributed Systems*, Dec. 2012.

[5] Y. E. Sagduyu, Y. Shi, S. J. Mackey, D. H. Cansever, M. P. Patel, and B. F. Panettieri, "Analytical Framework for MANET Learning and Routing," *IEEE MILCOM*, Nov. 2016.

[6] T. A. Babbitt, C. Morrell, B. K. Szymanski, and J. W. Branch, "Self-selecting reliable paths for wireless sensor network routing," *Computer Communications*, Oct. 2008.

[7] K. Neema, Y. E. Sagduyu, and Y. Shi, "Search Delay and Success in Combined Social and Communication Networks," *IEEE GLOBECOM*, Dec. 2013.

[8] Y. E. Sagduyu, Y. Shi, and K. Neema, "Search in Combined Social and Wireless Communication Networks: Delay and Success Analysis," *IEEE Trans. Wireless Communications*, Sept. 2015.

[9] Y. E. Sagduyu and Y. Shi, "Navigating a Mobile Social Network," *IEEE Wireless Communications Magazine*, Oct. 2015.

[10] Z. Lu, Y. E. Sagduyu, and Y. Shi, "Friendships in the Air: Integrating Social Links into Wireless Network Modeling, Routing, and Analysis," *IEEE NetSciCom*, Apr. 2016.

[11] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, Oct. 2003, Network Working Group. RFC 3626.

[12] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time Network Emulator," in *Proc. IEEE MILCOM*, 2008.

[13] J. Ahrenholz, T. Goff, and B. Adamson, "Integration of the CORE and EMANE Network Emulators," in *Proc. IEEE MILCOM*, 2011.

[14] H. Inaltekin, M. Chiang and H. V. Poor, "Average Message Delivery Time for Small-world Networks in the Continuum Limit," *IEEE Transactions on Information Theory*, Sept. 2010.

[15] N. Eagle and A. Pentland, "Reality Mining: Sensing Complex Social Systems," *Journal of Personal and Ubiquitous Computing*, May 2006.