Interactive Analysis and Simulation of VANETs using MOWINE

Ian Downes
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
Email: downes@stanford.edu

Branislav Kusy
CSIRO ICT Centre
Technology Court
Pullenvale, QLD 4069, Australia
Email: Brano.Kusy@csiro.au

Omprakash Gnawali and Leonidas Guibas
Computer Science Department
Stanford University
Stanford, CA 94305, USA
Email: {gnawali,guibas}@cs.stanford.edu

***Abstract*—We present a mobile wireless network evaluation tool (MOWINE) that can analyze mobility and wireless connectivity traces to quantify performance of different network protocols in a given application scenario. We model the highly dynamic connectivity of mobile networks using time-expanded graphs which allow us to study the best-case performance of network protocols. MOWINE also integrates with a network simulator that can evaluate performance of recent VANET routing protocols with a user specified data load. This combination of network modeling and simulation enables users to gain deeper insights into the performance of routing protocols, for example, by distinguishing the limitations of a particular routing protocol from the fundamental limitations of the underlying network. Thus, network engineers can use MOWINE to study and fine-tune performance of real-world network protocols in different scenarios before deploying the network, and hence *engineer* the mobile ad hoc network. We demonstrate the effectiveness of MOWINE by analyzing and understanding the performance of six different VANET routing protocols using real-world taxi cab traces collected in San Francisco. After identifying the bottlenecks, we demonstrate several basic network engineering tasks and their implications on the network performance.**

I. Introduction

Engineering and deployment of Vehicular Ad hoc Networks (VANET) lags far behind research. There are few examples of deployed VANETs despite the need for communication between the vehicles and central application and information servers [1]. Existing networks are limited to one-hop communication with other vehicles or rely on one-hop communication to connect to the networking infrastructure. This is feasible because cities already have communication infrastructure for mobile phones and devices. However, the operational cost for this network architecture can be high—each node needs to subscribe to the mobile data service and such a network may not be present or functional [2]. Architecting these networks as Mobile Ad hoc Networks (MANET) can dramatically reduce the operational cost because the nodes would relay messages for each other without requiring an infrastructure service.

Despite this clear advantage of MANETs, they are not used in practice, in part because there are not the tools and methodologies necessary for network planners and engineers to reason about the performance of real-world MANET deployments. Analysis of such networks is challenging due to the time variability of the network as induced by the mobility and link volatility.

Previous work has simulated the performance and optimal bounds for wireless network protocols under different scenarios [3], [4]. Although useful for comparison the computed bounds do not provide any specific reason why the performance is bounded or insight into how the protocol can be tuned/improved to achieve performance closer to the bound. In this paper, we take the first step towards closing this loop. We develop a new network analysis tool called **MO**bile **WI**reless **N**etwork **E**valuator (MOWINE) that combines analysis, simulation, and engineering. It includes several physically-informed optimizations to improve the running time over state-of-the-art modeling techniques used to analyze MANETs to enable interactive exploration of network and protocol parameters in larger and more realistic networks.

Simulation of concrete protocols for these delay tolerant networks (e.g., First Contact, Direct Delivery, ProPHET) allows us to understand the performance of these protocols in a given mobility scenario. Although simulation of these protocols is not novel, applying the results obtained from these simulations in the context of achievable performance of these networks is new and provides an understanding of the performance seen in the simulation. For example, the low performance achieved by a particular protocol in a given network scenario could be due to the properties of the protocol or due to the fundamental properties of the network itself (e.g., the network is rarely connected). Studying the results of simulation and modeling together helps disambiguate these cases and understand the performance of these protocols with respect to what can be achieved in those networks.

Finally, network engineering allows us to perform what-if experiments and design deployments that will meet the communication requirement of the applications. Figure 1 shows the user's interaction with the MOWINE toolkit. If modeling of a network indicates that a network cannot fundamentally meet the data rate requirements, would it be better to deploy some static nodes or increase the transmit power of the radios? Network engineering has been used to answer such questions in the context of the Internet and enterprise networks for decades. We believe that MOWINE is a first attempt to bring network engineering tools to the analysis and planning of these delay tolerant VANETs.

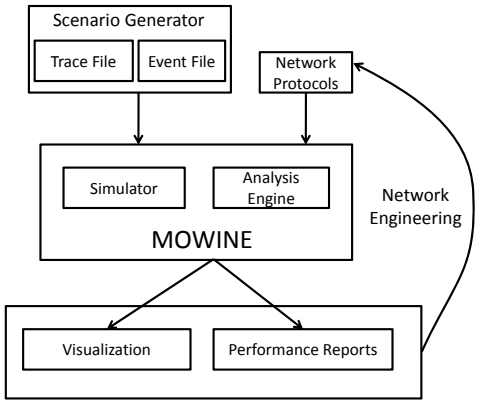

Fig. 1. The main components of the MOWINE toolkit. The simulator and analysis engine enable users to explore and optimize network parameters for the desired mobility and application scenarios.

We demonstrate the benefits of MOWINE by using it to analyze and understand the performance of a network formed by 500 taxi cabs in San Francisco using real-world mobility traces. We then use the engineering tools available in MOWINE to explore the parameters of this network to eliminate performance bottlenecks.

In summary, we make the following contributions in this paper:

- Design of a fast modeling tool that can estimate the upper bound of network performance at interactive speeds even with hundreds of nodes and complex connectivity and mobility.
- Design and implementation of MOWINE, a tool that combines modeling, simulation, and network engineering tasks for mobile and vehicular ad hoc networks.
- Demonstrate the effectiveness of MOWINE in examining real-world VANET, diagnosing the bottlenecks, and performing network engineering tasks to plan network servicing.

In the remaining parts of the paper, we first review related work in Section II. We then describe the network modeling and analysis part of MOWINE in Section III and the network simulation engine that MOWINE runs internally in Section IV. We review various network engineering tasks in Section V and provide an evaluation of MOWINE in Section VI. Finally, we conclude the paper and discuss future work in Section VII.

II. Related Work

A. Routing protocols

In dynamic networks the route between source and destination may change rapidly, quickly leading to poor routing performance of traditional proactive protocols, or a route may not exist, leading to the failure of even reactive routing protocols [5], [6], [7]. Delay tolerant protocols succeed by using routes that exist over time and space, storing packets as necessary and then forwarding when a suitable next-hop is encountered [8]. In our case studies we evaluate DTN protocols as the network is frequently disconnected.

Routing under such conditions is challenging as the routes to be used are over time and are not known in advance. DTN protocols must explore the network to discover routes and this can be costly, depending on the network dynamics. In the simplest approach, epidemic routing, the protocol floods the network, finding the optimal path but with high overhead [9]. Recently developed protocols attempt to find good routes with low overhead and network cost. In [10] the number of packet replications is limited, resulting in only some of the possible routes being explored but much lower cost. In MaxProp nodes use contact likelihood vectors to estimate paths and their cost, enabling ordering of packet replication [11]. Additions to MaxProp include acknowledgements to purge the network of successfully delivered packets and preference given to packets with low hop count, encouraging early replication. Other protocols try to determine the most suitable next hop based on historical information [12], [13]. In each protocol there are parameters that can be tuned to optimize the performance of the protocol under a particular scenario [12] but it often through empirical comparison. MOWINE combines simulation of routing protocols while also analysing the underlying network to understand how each protocol behaves under different conditions.

B. Performance bounds

Determing the performance bounds of wireless networks, even without mobility, is difficult due to the broadcast channel and resulting interference and the degrees of freedom in choosing transmit power levels and modulation schemes. Toumpis explored the capacity regions of wireless networks and considered multi-hop networks under a variety of transmission strategies [14]. In was noted that it was not feasible to analyse networks larger than approximately 15 nodes. MOWINE has a simpler communication model, sufficient for DTN networks, allowing it to scale to the analysis of hundreds of nodes. For very large networks with random and independent and identical mobility patterns Grossglauser and Tse showed in [15] that the average asymptotic throughput is kept constant for communicating nodes, independent of the number of nodes in the network.

Several descriptions of DTN protocols have included comparisons to a computed optimal bound where full knowledge about the network is assumed. These bounds have been formulated as linear or mixed-integer programs [4], [3] and solved using an appropriate solver. In [4] the delay of their RAPID protocol was compared to the delay from optimizing a integer program where time was dividing into discrete intervals and it was assumed that a node meets at most one other node in an interval. They note that the integer program grows in complexity with the number of packets and that it was only feasible to solve the program with at most 350 packets. In [3] their linear program formulation permits the fragmentation of packets and seeks to minimize the average delay over all packets in the network. In both cases interference was not

considered due to the additional complexity but others have incorporated this for optimizing smaller networks [16].

C. Simulation

Several wireless network simulators have recently been developed for DTN and/or MANET simulation [17], [18]. The JiST/SWANS simulator provides an efficient simulator that can scale to tens of thousands of nodes and includes support for several MANET protocols while providing realistic modeling of the radio and channel. The ONE simulator, used in MOWINE, has been developed specifically for DTN simulation and uses a simplified radio model, making large-scale simulations feasible [19]. It provides implementations of many DTN protocols.

III. Network Analysis

Static networks can be naturally described using a graph. The nodes represent the communicating agents and edges represent connectivity between the agents and can include attributes such as capacity, delay and expected successful delivery rates. In static scenarios these attributes are assumed to be time-invariant (or at least stationary) and the network can be analyzed using graph-based techniques. However, if the communicating agents are mobile, the network is dynamic and the existence and attributes of connecting edges is strongly time dependent and traditional analysis cannot be applied.

Our network analysis technique expands the time dependence and constructs a much larger graph which is a complete representation of the network over a time interval. The representation is can be considered an oracle which contains information about all of the mobile agents and their contacts with each other. Routing on this expanded network will provide the best possible paths and gives an upper bound to the performance of any routing algorithm.

A. Modeling Assumptions

Several assumptions are made in modeling the dynamic network which influence the representation and construction of the time-expanded network. These assumptions are more simplifying than those used for static wireless networks but can be justified for a dynamic network where throughput and delivery is generally limited by the agents' motion rather than the characteristics of the radio or channel.

1) A fixed communication range is assumed
2) Agents that can communicate do so without interfering with any other communicating agents.
3) Communication success is deterministic, depending only on the communication range.

These assumptions are the same as those of the DTN simulator used for comparison.

B. Network Representation and Construction

The dynamic network is represented as a time-expanded network, a representation first introduced by Ford and Fulkerson in [20]. The time-expanded network is a static network where nodes are replicated and connected according to the dynamic changes in the network. In effect a copy of the network in each of its configurations is made.

The edges between replicated nodes for each configuration are directed and enforce the directionality of time. Edges between replicated nodes of the same agent encode the option of a node buffering a packet from one time to another. Communication links are encoded as edges between replicated nodes of distinct agents and occur between nodes at the same time. Edges are weighted by their delay, for communication edges this is the packet size/data rate while for buffering edges it is the delay between the times the nodes represent. Edges are also labeled by the radio data rate and the maximum buffer size. A path through the time-expanded network is a sequence of communication and buffering directed edges.

A time-expanded network can be much larger than the original network, depending on the degree of change over time and how it is modelled. The naive approach is to sample time at discrete points and replicate the network at each point. This approach is not scalable due to two related reasons. First, it leads to an extremely large graph which may not capture all necessary information, i.e. changes within an interval. Second, it includes unnecessary duplication if the graph, or parts of it, are static across multiple intervals. A more efficient approach is to determine the discrete points in time where the structure of the network changes and to only include those times in the time-expanded network.

In the dynamic communication network, it is the times when the connectivity changes which dictate the structure of the time-expanded network. For example, for communication purposes, under the fixed communication range model, it does not matter if the nodes move relative to each other provided they either stay out of range or stay within range. It is only the times when the communication link is established or broken that need to be included.

The construction of the time-expanded graph is achieved with the following steps:

1) Determine the node distance function over time for all node pairs.
2) Threshold the distances according to the communication range.
3) Determine pairwise contact intervals and convert to an ordered list of connection and disconnection events.
4) Process events and keep track of the creation and destruction of connected components in the graph. As components are created and destroyed add these as replicated nodes to the time-expanded graph, creating buffer and communication edges as appropriate.

IV. Network Simulation

Simulation of delay tolerant protocols allows us to understand the performance of these protocols under a given mobility and data traffic scenario at the packet level. Numerous simulators and simulation frameworks have been used in the literature, ranging from wireless network simulators adopted to the VANET domain [17], [21], to complex simulation frameworks designed to support realistic mobile traces and

event generators. MOWINE is independent from the underlying simulation tool. It extends the results obtained from the simulations in the context of achievable performance of the underlying networks. Studying the results of simulation and modelling helps disambiguate the cases when poor performance of a protocol is due to poorly connected underlying network from the cases where a protocol is a poor match to the given mobility and data traffic patterns.

We study performance of delay-tolerant network routing protocols using the Opportunistic Network Environment (ONE) Simulator [19]. The simulator framework implements six popular delay-tolerant routing protocols and includes options for generation of motion patterns and radio traffic. In the reminder of this section we provide a brief overview of the routing protocols used, describe how we achieve realistic simulation using traces of taxi cars in San Francisco, and discuss a several application scenarios that motivated our choice of data traffic patterns.

A. Routing Protocols

One framework provides implementation of the following routing protocols:

- **First Contact (FC)**: a single copy protocol where packets are propagated through random walks;
- **Direct Delivery (DD)**: a single copy protocol where nodes cary packets until they reach destination;
- **Spray-and-wait (SW)**: an n-copy protocol that distributes at most n copies of packets through chance encounters [10];
- **Epidemic (E)**: a flooding protocol that copies packets to all encountered nodes;
- **ProPHET (P)**: a protocol that for each node estimates the probability of delivery of a packet to a destination based on the previous history of the nodes [12];
- **MaxProp (MP)**: a flooding protocol that prioritizes message transmissions based on message hopcounts and message delivery probabilities [11].

For all protocols, we run two hour simulations and use either an IEEE 802.15.4 based radio model with 250kbps bandwidth and 200m radio range or an 802.11 based model with 10Mbs bandwidth and 500m range. Nodes transmit 10-50kB packets, with the size selected uniformly at random. Data buffers at each node hold approximately 100 packets and the time-to-live of packets was set to 60 minutes. We focus on the metrics of the reliability of data delivery, measured as the ratio of successfully received unique packets out of the total transmitted and of the latency of data packet delivery.

B. Motion Patterns

The ONE framework provides a spectrum of motion models, from a simple random waypoint model, to more complex models that select a destination from pre-configured points of interest and trace shortest paths between two points using freely available road maps. The simulator also allows for more realistic simulations using external traces of moving objects. We used the San Franciso taxicab dataset which recorded the motion of 500 taxicabs in San Francisco over a period of one month.

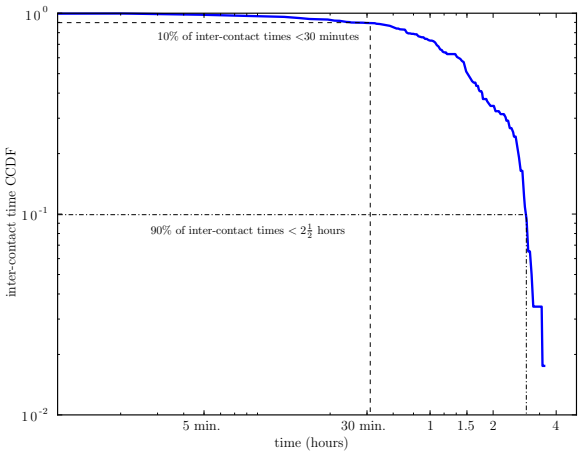

Fig. 2. Intercontact time complementary cumulative distribution plotted on a log-log scale. The distribution indicates that very few nodes reconnect with intervals less than 30 minutes but do after approximately 2 hours. Routes are expected to require several hops and have high average latency.

C. Data Traffic Patterns

We explore two communication primitives in our mobile networks. Peer-to-peer messages are transferred through possibly multiple unicast transmissions from source to destination. Data collection, on the other hand, delivers packets from all nodes in the network to a common data sink. Both modes of transmission are important in the realm of wireless vehicular networks: mobile vehicles can exchange information between themselves using peer-to-peer communication, while a taxi dispatcher might want to receive periodic status updates from all cars in the fleet. We set inter-packet-intervals (IPI) to 30 seconds for peer-to-peer transmissions and triggered data collection every 5-10 minutes.

V. Network Engineering

Network engineering allows us to consider various design alternatives for the network while achieving the required application requirements. Studying these design alternates and what-if experiments can guide network planning and deployment.

Although network engineering is routinely used in the Internet and enterprise network planning and deployment, there is no prior work in using those tools in VANETs. To apply network engineering to VANETs, we need to first identify relevant engineering primitives in the context of VANETs. After a survey of wireless network and MANET/VANET literature, we have identified the following common operations the engineers perform during deployment and performance tuning:

- *Add nodes*. Add nodes in certain parts of the network to improve connectivity or to provide additional relay nodes in multihop routing.

- *Remove nodes*. Remove nodes to reduce interference and improve spatial diversity.
- *Adjust radio transmit power*. Set the radio transmit power to increase or decrease the communication range. An increase in communication range can increase network connectivity. Sometimes, receiver gain can be adjusted separately.
- *Adjust node placement and orientation*. Wireless network engineers sometimes move the nodes away from obstructions, put the nodes in line of sight, move nodes closer to each other, or rotate to align the antenna.

There are additional engineering tasks that are not relevant in wireless networks in general, but are applicable in VANETs. We synthesized these VANET engineering tasks based on the common experimental setup reported in the literature:

- *Adjust velocity*. Adjust the velocity of the mobile agent depending on the type of the mobile agent (person, vehicle, etc.).
- *Adjust mobility structure*. Adjust the structure of the mobility of the mobile agents. For example, random waypoint and reference point group mobility [22].

The network engineering toolset in MOWINE also allows the users to specify these application requirements:

- *Communication pattern*. Currently MOWINE supports collection (all the nodes sending data to the base station) and dissemination (the base station sending data to all the nodes) communication patterns.
- *Data timeout*. The data is not useful if it is not received within this timeout interval after the source injects the packet into the network.

MOWINE allows the network engineers to explore the design space of their networks in the context of these application requirements by performing the engineering tasks and seeing how they impact the performance of the protocols. By analyzing real-world networks the best protocol and parameters can be chosen for the network before deployment.

VI. Evaluation

In this section, we describe the evaluation setup and demonstrate the power of an analysis, simulation, and engineering tool by interactively exploring the design space of a VANET.

A. Implementation

We implemented the three tools that constitute MOWINE as three separate components. The analysis tool is implemented in Python using the LANL NetworkX library [23]. The simulation tool is a script that drives the ONE simulator. The network engineering tool is a set of scripts that interact with the analysis and simulation components. These three toolsets are glued together using shell scripts. They will be integrated using a GUI in the future.

B. Scenarios and Datasets

We have used MOWINE with simple to complex synthetic and real-world VANET scenarios. In practice, we found that analysis of large real-world scenarios is most challenging to the network engineers. It is in these complex scenarios that tools like MOWINE that combine modeling, simulation, and network engineering tasks become critical to not only understanding the performance but also the reasons for the bottlenecks and engineering ways to overcome those bottlenecks. For this reason, we will focus our discussion in this section on the network scenario with 500 taxi cabs in San Francisco.

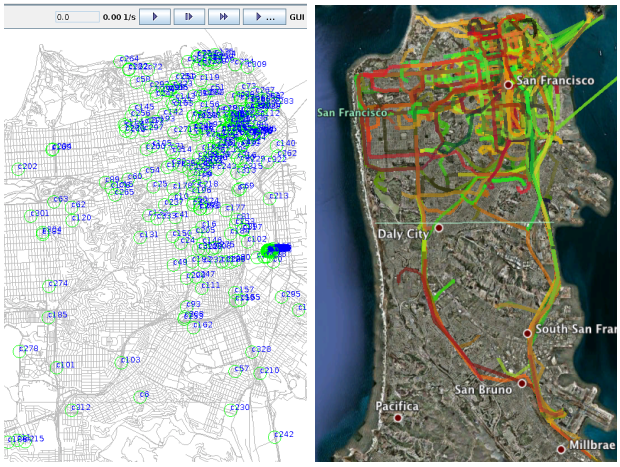

Fig. 3. Screen capture of our simulator in action, showing street map of San Francisco on the left. The Google Earth map of San Francisco overlayed with trajectories from the cab traces dataset on the right.

The dataset containing 500 vehicles recorded over 1 month period is freely available online [24]. Limited filtering was applied to the traces, namely we removed points that violated the maximum speed constraint (160 kph) and fell outside our region of interest.

C. Running time

The biggest hurdle in building an interactive tool such as MOWINE is reducing the running time for the analysis tool. Traditionally, estimating the upper bound of the performance of a mobile ad hoc network has taken hours [3]. This computational overhead and the long latency render these tools unsuitable for inclusion in an interactive tool. In our experiments, we found that MOWINE can analyze the typical VANET scenarios at close to interactive speeds. Table I show the running time for different scenarios based on the taxicab dataset. The time required for construction and analysis of a network depends on the number of mobile agents and the complexity of their mobility and communication graphs. A four hour snapshot of the 500 node taxi cab dataset with a 200 meter radio range is constructed in under three minutes using a single processor core of Sun Blade X6240 (AMD Opteron 2384) and requires approximately 2 GB of memory. Increasing the snapshot time window leads to an approximately linear increase in the processing time and memory requirements. In contrast, increasing the communication range leads to a super-linear (but generally sub-quadratic) increase in processing time and memory due to the more complex and far-reaching interactions between the mobile agents.

TABLE I
CONSTRUCTION AND AGGREGATION ANALYSIS TIME FOR DIFFERENT SCENARIOS BASED ON THE TAXICAB DATASET

Time interval	Comm. range	Running time	Memory
8–10 am	200m	30 sec.	450 Mb
8–10 am	500m	2 min.	1.05 Gb
8–12 noon	500m	3 min.	2.0 Gb

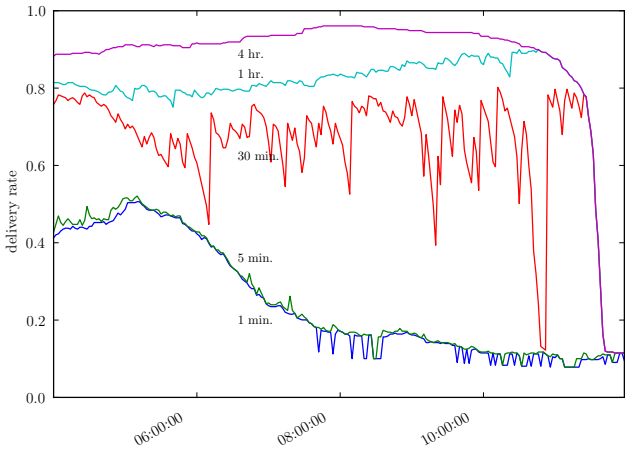

Fig. 4. Delivery rates for different timeout values. The relative high delivery rate at short times (1 and 5 minutes) in the early part of the interval is due to taxicabs stationary at the depot.

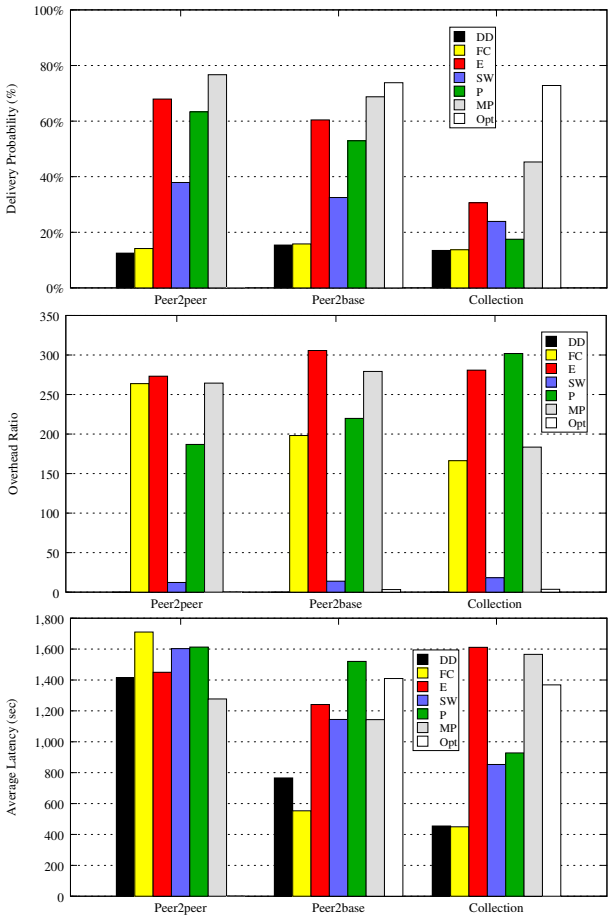

Fig. 6. Delivery rates, overhead ratios, and average latency for the six delay tolerant routing protocols and three different data traffic patterns. The optimal performance is denoted by *Opt*. Spray and wait (SW) protocol might be a good choice if a good network performance and modest resource usage is desired. Otherwise, the MaxProp (MP) protocol clearly outperforms other protocols in all application scenarios.

Through these experiments, we conclude that MOWINE satisfies the requirement of interactive running speed. As a comparison, The One simulator required almost 10 hours to simulate the MaxProp protocol running on a two hour trace of the taxicab network.

D. Temporal Variation in Performance

The main advantage of a tool like MOWINE is being able to study the performance of a network using simulation and analysis and synthesizing the insights using a combination of those results. Figure 4 illustrates this technique by plotting the maximum achievable delivery rate for a selection of timeouts over an eight hour interval. The delivery rate is the fraction of packets that were sent by the sender and that were received at the intended destination within the timeout.

Several observations can be made from these results. Firstly, the performance of the network does change over both small and large time intervals. Second, we notice that the achievable delivery ratio is also less than 100%. With a timeout of 30 minutes, the delivery rate fluctuates between 50% and 70% while the four hour rate remains greater than 90%, indicating that long timeouts are necessary. We also find that the occasional dips in performance are due to the fundamental property of the network at the given time (e.g., disconnection), which also reduced the achievable rate.

MOWINE thus allows us to put the simulation results in context. Simulation results are informative on their own and give insight about the performance of a protocol in a network. Combining these results with analysis results in a single tool helps us explain the cause of good or bad performance based on the fundamental property of the mobile network.

E. Protocol Selection

If we know that a protocol performs far worse than the achievable performance, MOWINE helps us identify alternative protocols that might perform better in the given network. To enable this decision making, we select a number of relevant protocols and configure MOWINE to simulate the selected protocols in the same network. We also run the analysis tool. MOWINE then combines the results of these simulations and analysis into a single graph as shown in Fig. 6.

The figure shows that different protocols are better suited for different optimization goals. For example, in energy constrained systems (such as wireless sensor networks), SW protocol provides the best performance while requiring modest energy resources. On the other hand, MP protocol is clearly the preferred under all application scenarios, if high delivery

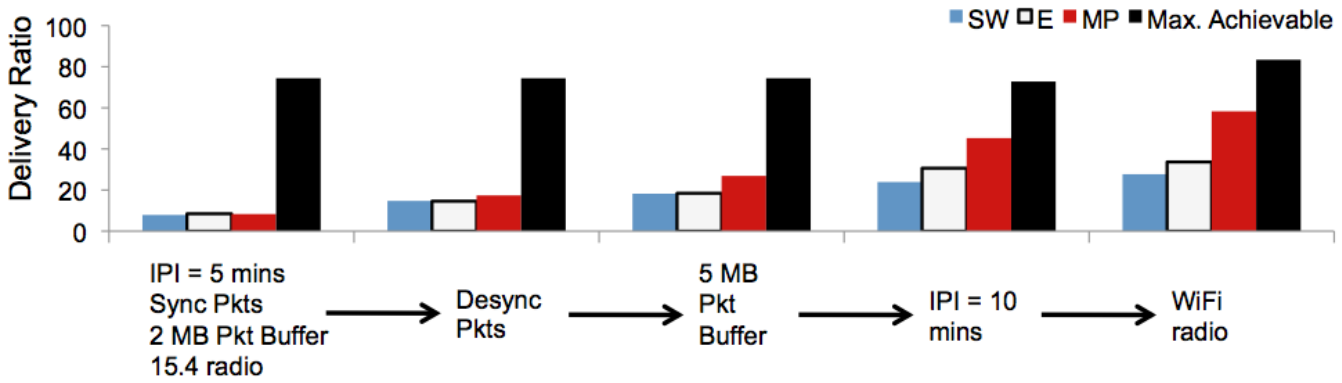

Fig. 5. Exploring the change in delivery ratio and maximum achievable delivery using the network engineering tool in MOWINE

rates are required irrespective of the cost. MOWINE allows us to explore the space of possible solutions, by providing the explicit upper (lower) bounds on the network performance (overhead rates).

F. Network Engineering

One of the observations from figure 6 is that even the best of the protocols still achieve lower performance than the maximum achievable performance. Is there a way we can get closer to the maximum achievable performance? We answer this question while we explore the network engineering features of MOWINE.

We select E and MP protocols and run MOWINE with these settings:

- Inter Packet Interval (IPI) of 5 minutes.
- Synchronized packet injection across the network.
- 2 MB packet buffer.
- 15.4 radio.

The network engineering tools in MOWINE allows us to change each of these parameters one at a time and visualize the change in not only the performance of specific protocols but also the achievable performance. Figure 5 shows the results from this experiment, which involved these tasks performed in MOWINE:

- First, we de-synchronize the packet injection. We observe an increase in the performance of both the protocols. This is due to fewer collisons and channel contention as the nodes inject packets at different times. The tool also reported that packet buffer overflowed on some nodes.
- Second, we increase the packet buffer to 5 MB. We note an improvement in performance.
- Third, we increase the IPI to 10 minutes. This task increased the delivery ratio as well as the maximum achievable delivery ratio because of the added opportunity to deliver the packets before the links are disconnected.
- Finally, we select WiFi interface and rerun the scenario and we discover that the delivery ratio increases further.

Thus, MOWINE allows us to explore a number of networking scenarios and their impact on the performance so that we can make informed decisions about network parameters when we deploy the network.

VII. Conclusions and Future Work

The MOWINE tool combines analysis, simulation, and network engineering to understand network performance and optimization. We found that MOWINE can help analyze and explain the observed performance in a network. It also enables rapid exploration of the network design space to find the optimum protocols and parameters required to overcome the performance bottlenecks and meet application requirements.

We plan to improve MOWINE to study further graph properties of the underlying communication graph. This will provide deeped insight to the performance of protocols in poorly connected or otherwise constrained networks and will help identify parameters of protocols that be optimized for the mobility and data traffic scenarios.

References

[1] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, "Performance evaluation of safety applications over DSRC vehicular ad hoc networks," in *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. New York, NY, USA: ACM, 2004, pp. 1–9.

[2] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1999, pp. 195–206.

[3] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 145–158.

[4] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 373–384, 2007.

[5] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad hoc on demand distance vector (AODV) routing," IETF Internet draft, draft-ietf-manet-aodv-09.txt, November 2001 (Work in Progress).

[6] I. Chakeres and C. Perkins, "Dynamic MANET On-demand (DYMO) Routing," Internet-Draft, draft-ietf-manet-dymo-12.txt, 2008.

[7] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2004, pp. 187–198.

[8] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 145–158.

[9] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., April 2000.

[10] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM, 2005, pp. 252–259.

[11] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," apr. 2006, pp. 1–11.

[12] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," in *Service Assurance with Partial and Intermittent Resources*, ser. Lecture Notes in Computer Science, P. Dini, P. Lorenz, and J. N. d. Souza, Eds. Springer Berlin / Heidelberg, 2004, vol. 3126, pp. 239–254.

[13] T. Spyropoulos, "Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *In Proceedings of IEEE PerCom Workshop on Intermittently Connected Mobile Ad Hoc Networks*, 2007.

[14] S. Toumpis and A. Goldsmith, "Capacity regions for wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 2, pp. 736–748, 2001.

[15] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. Networking*, vol. 10, no. 4, pp. 477–486, 2002.

[16] V. Kolar and N. B. Abu-ghazaleh, "A multi-commodity flow approach to globally aware routing," in *in Multi-Hop Wireless Networks," in IEEE PerCom*, 2006.

[17] S. McCanne and S. Floyd, "ns Network Simulator," http://www.isi.edu/nsnam/ns/.

[18] R. Barr, Z. J. Haas, and R. van Renesse, "JiST: an efficient appraoch to simulation using virtual machines," *Software: Practice and Experience*, vol. 35, pp. 539–576, 2005.

[19] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[20] L. R. F. Jr. and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations Research*, vol. 6, no. 3, pp. pp. 419–433, 1958.

[21] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," *SIGSIM Simul. Dig.*, vol. 28, no. 1, pp. 154–161, 1998.

[22] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *MSWiM '99: Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 1999, pp. 53–60.

[23] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[24] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.