

Integrated Scheduling of Application- and Network-Layer Tasks in Delay-Tolerant MANETs

José Brustoloni, Sherif Khattab, Christopher Santamaria, Brian Smyth and Daniel Mossé

Dept. Computer Science, University of Pittsburgh

{jcb,skhattab,cjsst55,bps,mosse}@cs.pitt.edu

Abstract—Natural or man-made disasters can partition networks while threatening human lives. Because conventional Mobile Ad-Hoc Networks (MANETs) cannot route messages across partitions, they may not adequately support relief efforts. To forward messages across partitions, delay-tolerant networks (DTNs) exploit in-network storage and mobility. Many previous DTN routing protocols either opportunistically use, but do not modify, nodes’ mobility, or require dedicated mobile gateways. This paper contributes a cross-layer DTN routing approach based on the observation that application-layer orders from a MANET’s leader also control workers’ mobility and ability to forward messages. Our approach attempts to minimize deadline misses and energy consumption by scheduling worker tasks considering both application- and network-layer needs. Simulations demonstrate performance benefits of our approach in a variety of scenarios.

I. INTRODUCTION

MANETs enable communication when network infrastructure does not exist (e.g., in military tactical communication) or has been damaged or compromised (e.g., because of a hurricane or terrorist attack). Each MANET node processes its own applications while also forwarding packets destined to other nodes.

This paper considers the problem of how to route packets in sparse MANETs with a leader. In such networks, a distinguished node, the *leader*, assigns *tasks* to and receives *reports* from other nodes, the *workers*. Our motivating application is an emergency response team whose workers and leader (e.g., firefighters and chief) use a MANET to communicate.

Conventional MANET routing algorithms, such as DSR [2], AODV [3], and ZRP [4], may fail when nodes are sparsely distributed. These algorithms do not discover routes and drop packets when there is no end-to-end path between sender and receiver, as is common in sparse MANETs.

When end-to-end paths do not exist, some or all nodes can take on the role of delay-tolerant network (DTN) *gateways*, enabling communication that conventional MANET protocols cannot provide. DTN protocols are explicitly designed under the assumption that network partitions are common and that links may have time-varying capacity and loss probability. DTNs are inspired by constraints common in interplanetary communication [5]. However, they can also model sparse sensor and ad-hoc networks [6]. DTN’s main service is the non-interactive transport of a *bundle* (i.e., self-contained message) from a source to a destination. The bundle can be forwarded

between application-layer gateways overlaid on networks with widely different characteristics and variable connectivity. A gateway may not be able to forward a bundle immediately. In this case, the gateway stores the bundle in nonvolatile memory until a *contact* (i.e., forwarding opportunity) occurs. A gateway may take *custody* of a bundle, i.e., guarantee to the sender the bundle’s eventual delivery. The sender may select from a variety of postal-like levels of service and options, including delivery notification.

Several routing protocols have been proposed for DTNs. Some of them exploit preexisting node mobility to forward bundles opportunistically [7], [8], [9], [10], [11]. If mobility patterns happen to be unfavorable, however, these protocols may fail to deliver certain bundles, regardless of their urgency. Such failures may make opportunistic protocols inappropriate for emergency response.

Several other DTN protocols require special mobile gateways that are dedicated to forwarding bundles [12], [13], [14], [15]. These protocols usually predetermine the dedicated gateways’ mobility so as to satisfy bandwidth requirements [12], [14] or buffering constraints [15]. In emergencies, these protocols may have two shortcomings. First, they require dedicated gateways that may be unavailable. Second, the gateways may occupy personnel or other resources that could more profitably be allocated to application-layer tasks, which the protocols do not take into account.

This paper contributes a new, cross-layer DTN routing approach based on the observation that application-layer orders from a MANET’s leader also control workers’ mobility and ability to forward messages. Therefore, a leader may assign to workers not only application-layer tasks, but also *courier* tasks, whose purpose is to provide network-layer mobility needed for communication. Our approach attempts to minimize deadline misses and energy consumption by scheduling worker tasks considering both application- and network-layer needs.

Our simulations demonstrate that cross-layer DTN routing can have significant performance benefits. Compared to routing schemes that are oblivious to application-layer demands, such as selecting as gateway a worker closest to a destination, or that dedicate a set of nodes as gateways [12], [14], [15], cross-layer optimization can better adapt to different mixes of application-layer and network-layer loads.

The rest of this paper is organized as follows. In Section II, we state our assumptions about the network and its application. In Section III, we propose algorithms for courier scheduling. Section IV presents our simulation study and results illustrat-

ing the performance of these algorithms under a wide range of parameter values. We discuss related work in Section V, and conclude in Section VI.

II. NETWORK AND APPLICATION MODEL

This section describes our assumptions about the network, its applications, and performance metrics.

We assume that the network has a leader node and n worker nodes. The leader receives *reports* and responds to them by sending *task assignments* to workers. We consider that each task has an expected processing time and a deadline, and significant losses result from deadline misses. For example, a beach patrol leader may receive report of a drowning, and respond by assigning a cardio-pulmonary resuscitation task to a lifeguard. This task needs to be performed by a deadline, otherwise a life may be lost.

We assume that, after joining a network with a leader, a worker moves or performs a task only as result of an assignment from the leader. Therefore, the leader always knows approximately where its workers are supposed to be. We assume that the leader and workers have a map of the network's area. The map is annotated with landmarks that leader and workers can identify and report to each other, so as to convey approximate position information. Alternatively, the map is annotated with geographic coordinates that nodes can match to actual GPS measurements.

We further assume that assignments received from the leader may cause the network to become partitioned. The leader may send one or more workers to a site that, when reached, will lack an end-to-end route to the leader. In such a case, the network is split between the *main* partition, containing the leader, and a *subordinate* partition, containing the remote workers.

Nodes within each partition communicate with each other using conventional multi-hop routing protocols, such as DSR [2], AODV [3], or ZRP [4]. When the leader needs to communicate with a worker W_s who is in a subordinate partition, the leader sends a *courier assignment* to a worker W_m within the main partition. W_m physically moves between partitions so as to be able to forward task assignments to W_s and receive reports from W_s or other workers and forward them back to the leader.

The leader uses a *courier scheduling* algorithm to select W_m . The team's performance will depend on this algorithm. We consider two performance metrics. First, we measure the percentage of application-layer deadlines that are missed. This metric can be interpreted, e.g., as rate of human casualties. The courier algorithm should minimize this metric. A poor courier algorithm can increase missed deadlines by giving network-layer (i.e., courier) assignments to workers who could instead complete more application-layer tasks. Second, we measure the total distance traveled by couriers. This metric is a proxy for the energy that the algorithm spends on transportation. If nodes have energy constraints, increases in energy consumption decrease network uptime. Other factors being equal, this metric should also be minimized.

Performance of courier scheduling algorithms can be expected to depend on the application-layer load, L_a . The leader

can control the latter by varying the rate r_d at which it dispatches task assignments. If the average task processing time is p_{avg} , the dispatch rate at which all n workers can be expected to be busy is equal to $r_{dcap} = n/p_{avg}$. We define: $L_a = r_d/r_{dcap}$, i.e., we say the application-layer load is equal to the fraction of workers that can be expected to be busy, given the current task assignment dispatch rate.

III. COURIER SCHEDULING ALGORITHMS

This section describes the four courier scheduling algorithms that we compare in our simulations.

The leader uses one of these algorithms when the leader needs to communicate with a node in a subordinate partition:

- **random courier** – This is the baseline algorithm. It picks W_m randomly from among the workers in the main partition. The main advantage of this algorithm is that it requires only $O(1)$ time and does not need information about node locations, speeds, or current assignments.
- **dedicated courier** – This algorithm sets aside n_c nodes from the main partition for use as couriers. Similarly to dedicated mobile gateways used in previous works, such as [12], [13], [14], [15], dedicated couriers perform no application-layer tasks, and only dedicated couriers perform courier assignments. The leader selects the dedicated courier that is closest to any worker in the destination's partition or, in case of a tie, the dedicated courier with the lowest identifier. If there are $O(n_s)$ nodes in the destination partition, selection of the closest dedicated courier by exhaustive search takes $O(n_c n_s)$ time.
- **closest courier** – This algorithm picks as courier the worker W_m in the main partition that happens to be the closest to any worker in the destination's partition. If there are $O(n_m)$ workers in the main partition, such a selection by exhaustive search takes $O(n_m n_s)$ time.
- **highest slack** – This algorithm adds cross-layer optimization to the *closest courier* algorithm. It picks as courier the worker W_m in the main partition that has the *highest slack*. Given a worker W 's current application-layer task T , W 's slack S is the difference between time till T 's deadline and T 's remaining processing time. The leader picks as courier the worker W_m with the highest slack S . Using a priority queue, selection takes $O(\log n_m)$ time.

In each of these algorithms, the leader selects W_m only from among nodes that the leader can communicate with using conventional multi-hop routing. When there is no such W_m , the leader enqueues task assignments for later transmission in a single courier assignment. When W_m receives a courier assignment, it immediately interrupts whatever task T_m it may be performing, and moves toward the node W_d that is closest to W_m in the destination partition. If there are $O(n_s)$ nodes in that partition, simple exhaustive search can select W_d in $O(n_s)$ time. While there is a conventional multi-hop route between the leader and W_m , the courier scheduling algorithm may again pick W_m (e.g., if the latter still is the worker with highest slack when a new task assignment needs to be dispatched to the destination partition). Therefore, a courier may carry multiple task assignments in a single trip.

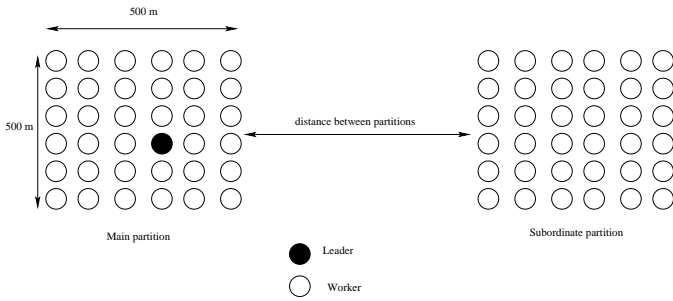


Fig. 1. The simulated scenario has 72 emergency responders equally divided between two partitions. The leader is near the center of the main partition.

TABLE I
SIMULATION PARAMETERS

Routing protocol	DSR
Work order deadline (s)	exponential(440)
Processing time (s)	$0.5 \times$ deadline
Radio range	200 m
Period of courier forwarding attempts	5 s
Experiment runs	10
Experiment duration	10,000 s
Discarded transient	First 1,000 s

While moving, W_m periodically tries, using conventional multi-hop routing, to forward the leader's task assignments to the destination partition. W_m may need to move all the way to within range of W_d for forwarding to succeed. However, forwarding may succeed sooner if there are intermediate couriers that form a path to the destination partition. After W_m successfully forwards the destination partition's task assignments, or the latter's deadlines all expire, W_m returns to its initial position and reports its return to the leader. If there still is time to complete T_m by its deadline, W_m does so; otherwise, W_m drops T_m and picks another task to execute. Workers do not need to report to the leader their positions or task completions as long as the latter reasonably comply with the task assignments previously received from the leader. Reports are forwarded to the leader by conventional multi-hop routing or returning couriers. Reports from workers can in turn cause the leader to assign new tasks to workers.

IV. EVALUATION

This section reports the performance of the previous section's algorithms in the scenario illustrated in Fig. 1, implemented on the ns-2 simulator [16].

In the simulated scenario, emergency responders are uniformly distributed between two partitions, with the leader close to the center of the main partition. Dedicated couriers, if any, reside on the side of the main partition facing the subordinate partition. Workers on courier assignment move at a speed of 5 m/s. By default, this is also the speed of dedicated couriers, the number of dedicated couriers (n_c) is 1, and the distance between partitions is 1200 m, but we vary these parameters for sensitivity analysis. Table I summarizes other parameters.

Fig. 2 shows the percentage of deadlines missed when the application-layer load in the subordinate partition ($L_{a,sub}$) is 0.6. When the application-layer load in the main partition

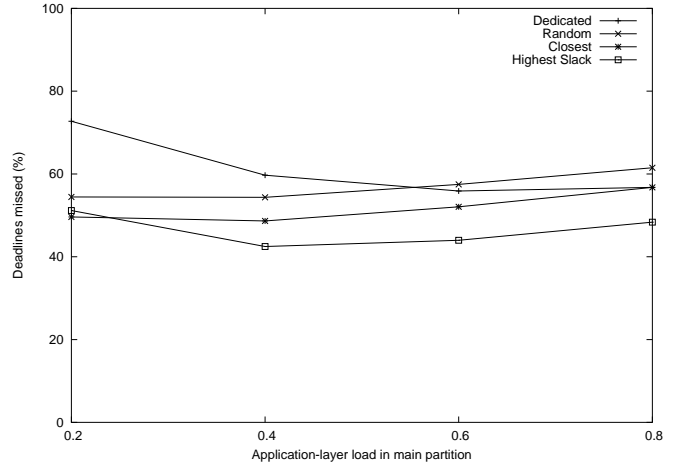


Fig. 2. Percentage of deadlines missed when application-layer load in subordinate partition is 0.6, as a function of application-layer load in main partition. Drawing information from multiple layers, *highest slack* minimizes deadline misses.

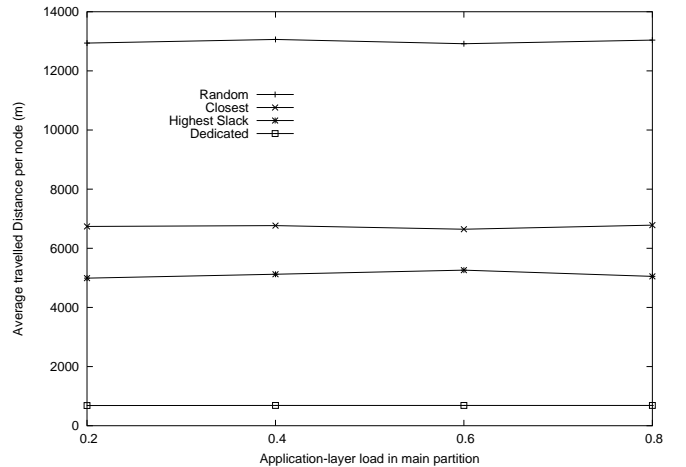


Fig. 3. Average distance traveled per node when application-layer load in subordinate partition is 0.6, as a function of application-layer load in main partition. *Dedicated courier* aggregates more messages per trip and minimizes distance traveled, at the expense of more missed deadlines.

is low ($L_{a,main} = 0.2$), the *dedicated courier* algorithm significantly underperforms the other algorithms (73% vs. 50 to 55% missed deadlines). For almost all values of $L_{a,main}$, *highest slack* results in significantly less missed deadlines than do the other algorithms (e.g., at $L_{a,main} = 0.8$, 48% vs. 57% or more missed deadlines). This advantage is due to cross-layer optimization. The second-best algorithm is *closest*, followed by *random*. The latter algorithm outperforms *dedicated courier* for low values of $L_{a,main}$, but not for high values.

Fig. 3 shows the average distance traveled per node under similar conditions. This average is calculated by dividing the total distance traveled by the total number of workers (71 in the simulated scenario). *Dedicated courier* results in the least distance traveled. The single dedicated courier is often away on a courier assignment when the leader needs to dispatch a new task assignment to the subordinate partition. Consequently, the leader accumulates many task assignments and sends them on a single courier assignment when the dedicated courier returns to the main partition. A disadvantage

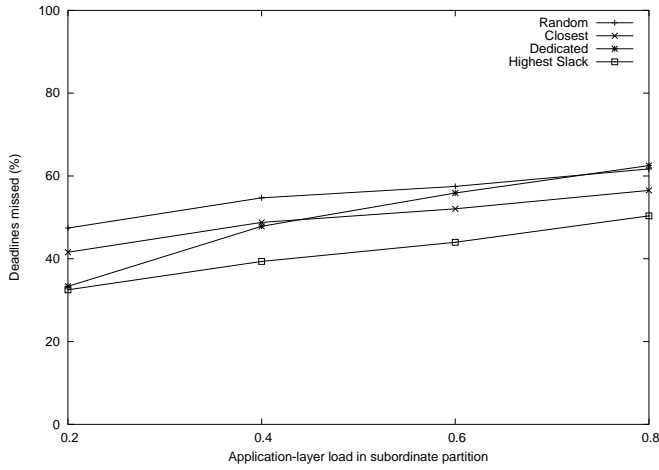


Fig. 4. Percentage of deadlines missed when application-layer load in main partition is 0.6, as a function of application-layer load in subordinate partition. *Highest slack* again provides the best performance.

of this accumulation is that it delays messages and increases missed deadlines, as shown in Fig. 2. *Highest slack* benefits from the fact that the selected courier may take considerable time to lose connectivity with the main partition. During this time, the leader selects the same courier if the leader needs to dispatch a new task assignment to the subordinate partition and the courier still has the highest slack. Thus, the courier may carry several task assignments on the same trip (although not as many as does *dedicated courier*, and without the latter algorithm's delays and missed deadlines). Couriers selected by the *closest* algorithm tend to lose connectivity with the main partition sooner than those selected by *highest slack*, and therefore on average carry fewer task assignments. Thus, this algorithm increases average distance traveled. The worst performance is that of *random*, because a courier carries more than one task assignment only if the courier is randomly chosen more than once. This event that has low probability.

Fig. 4 shows the percentage of deadlines missed as a function of the application-layer load in the subordinate partition, when the application-layer load in the main partition is 0.6. For all values of $L_{a,sub}$, *highest slack* results in the least missed deadlines. This advantage is again due to cross-layer optimization. *Dedicated courier* gives good performance at low values of $L_{a,sub}$. For higher values of $L_{a,sub}$, however, the other algorithms have an advantage because they have a larger pool of nodes to draw upon as couriers.

Fig. 5 shows the average distance traveled per node, under conditions similar to those in the previous figure. For all values of $L_{a,sub}$, *random* resulted in the most distance traveled, followed by *closest*, *highest slack*, and *dedicated courier*. The latter's advantage is again due to greater number of messages carried per trip.

Fig. 6 examines the effect of dedicated courier speeds, when the application-layer load in both partitions is 0.6, and the speed of workers on courier assignment remains constant (5 m/s). The relative performance of *dedicated courier* improves as the speed of dedicated couriers increases, but this speed needs to increase nearly threefold for *dedicated courier* to match *highest slack's* performance.

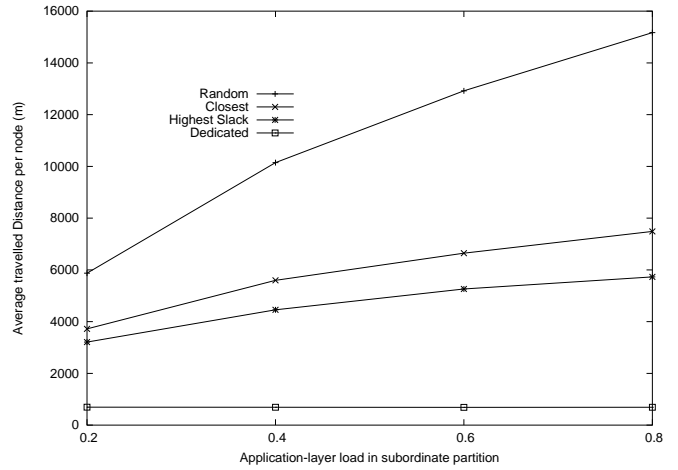


Fig. 5. Average distance traveled per node when application-layer load in main partition is 0.6, as a function of application-layer load in subordinate partition. *Dedicated courier* again minimizes distance traveled, at the expense of missed deadlines.

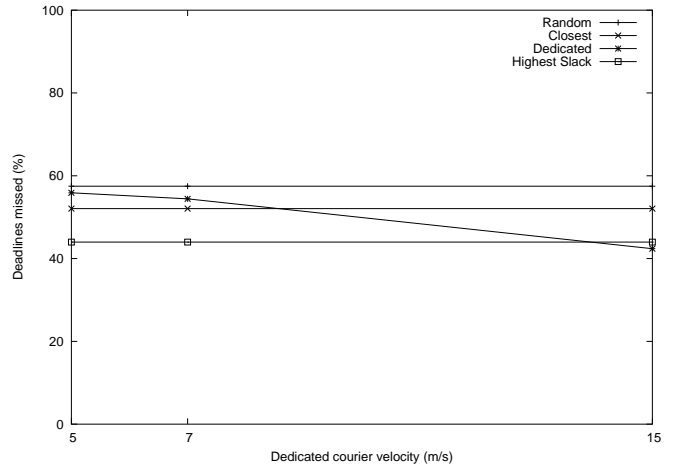


Fig. 6. Percentage of deadlines missed when application-layer load in both partitions is 0.6, as a function of dedicated courier speed. *Dedicated courier* is competitive with *highest slack* only if dedicated couriers move much faster than do workers.

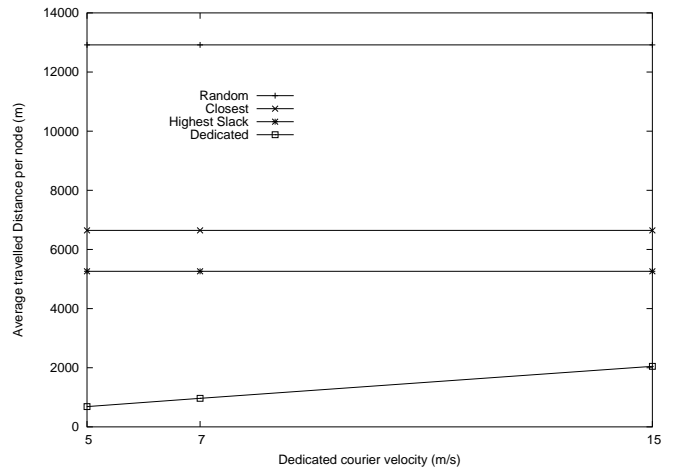


Fig. 7. Average distance traveled per node when application-layer load in both partitions is 0.6, as a function of dedicated courier speed.

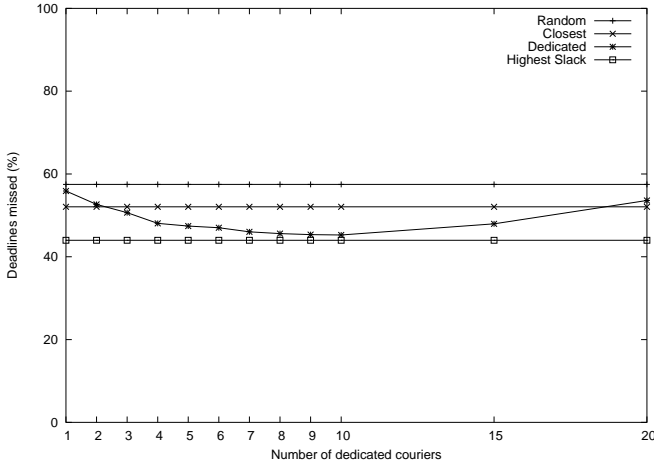


Fig. 8. Percentage of deadlines missed when application-layer load in both partitions is 0.6, as a function of the number of dedicated couriers. As this number increases, the performance of the *dedicated courier* algorithm initially improves, but then worsens, as too few nodes remain available as workers in the main partition.

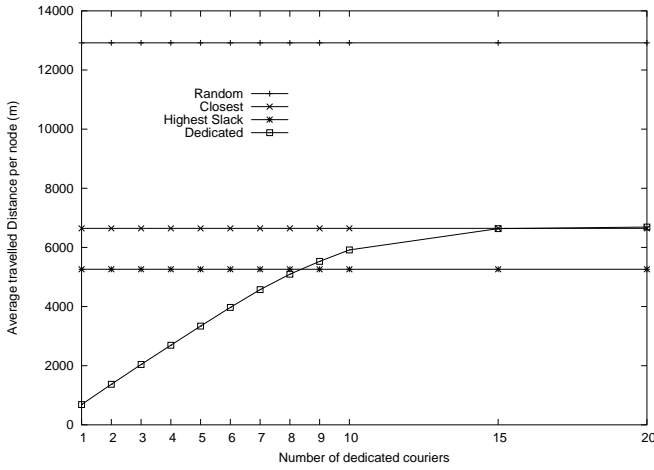


Fig. 9. Average distance traveled per node when application-layer load in both partitions is 0.6, as a function of the number of dedicated couriers. *Dedicated courier's* distance traveled advantage relative to *highest slack* disappears when the number of dedicated couriers increases.

Fig. 7 shows the corresponding distance traveled per node, when dedicated courier speed varies. *Dedicated courier* again results in the least distance traveled per node, but this distance increases with dedicated courier speed. This effect is due to less delay, and consequently less message accumulation, between courier trips, as the courier moves faster.

Fig. 8 shows the effect of varying the number of dedicated couriers (n_c), with application-layer load equal to 0.6 in both partitions. The percentage of deadlines missed with the *dedicated courier* algorithm approaches that of *highest slack* when $n_c = 10$ out of 35 workers in the main partition. However, further increases in n_c result in worse performance because less workers are left for performing tasks in the main partition.

Fig. 9 shows that the distance traveled per node increases greatly as the number of dedicated couriers increases. For more than 8 dedicated couriers, the corresponding algorithm results in more distance traveled than does *highest slack*. For

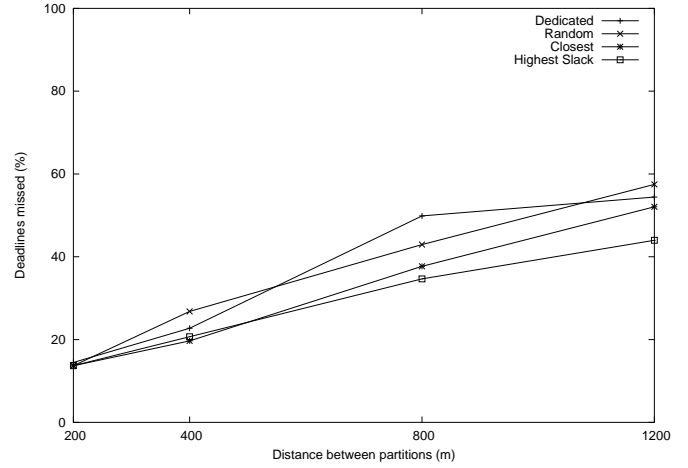


Fig. 10. Percentage of deadlines missed when application-layer load in both partitions is 0.6 and dedicated courier speed is 7 m/s, as a function of the distance between partitions. *Highest slack's* performance deteriorates more slowly with distance because couriers stay in range of the main partition longer and therefore can forward multiple messages in a single trip.

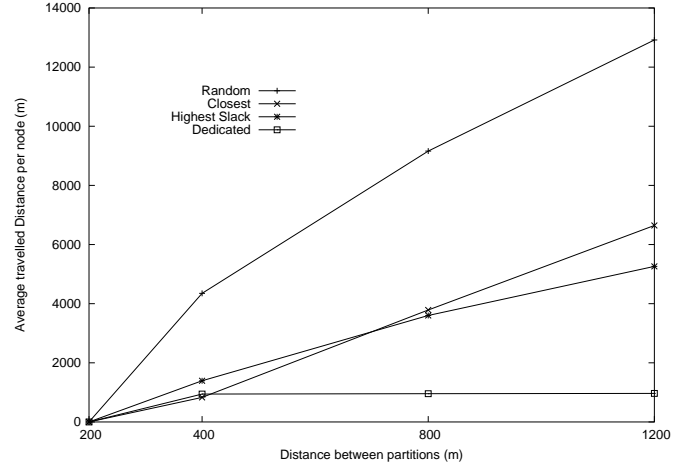


Fig. 11. Average distance traveled per node when application-layer load in both partitions is 0.6 and dedicated courier speed is 7 m/s, as a function of the distance between partitions.

20 dedicated couriers, the average distance traveled is similar to that of *closest*. This effect is due to less messages per trip as the number of dedicated couriers increases.

Fig. 10 shows the effect of changes in the distance between partitions, when the application-layer load in both partitions is 0.6 and dedicated courier speed is 7 m/s. At a distance of only 200m between partitions, they are still within range of each other. In this case couriers are not actually needed, therefore all algorithms perform the same. As distance between partitions increase, more deadlines are missed because couriers spend more time in transit. Performance degradation is highest for *dedicated courier*, which, unlike the other algorithms, draws the courier from a pool of only one node. Performance degradation with distance is lowest with *highest slack*, which selects couriers that stay in range of the leader longer and therefore can accumulate more messages per trip than do couriers selected by *closest*.

Fig. 11 shows the corresponding average distance traveled. *Dedicated courier* results in the least, or close to the least,

average distance traveled for all inter-partition distances. This advantage is due to fewer courier trips (at the expense of more missed deadlines, as shown in Fig. 10). *Closest* outperforms *highest slack* for short distances, but the latter's relative performance improves with greater distances. *Random* results in much worse performance for all inter-partition distances greater than the radio link range, because it often forwards only one message per trip.

V. RELATED WORK

Much of the earlier work on DTNs exploits preexisting node mobility to forward bundles. In Vahdat and Becker's original epidemic approach [7], at each contact, each node forwards to the other node any bundles that the latter does not have. Hopefully, each bundle will eventually arrive at its destination. Epidemic approaches have high overhead, because they flood the network, but are robust and require no prior knowledge. Harras, Almeroth, and Belding-Royer [9] propose and compare techniques for reducing the overhead of epidemic routing, by forwarding probabilistically, imposing a time-to-live or deadline for each bundle, or having the destination of a bundle acknowledge its reception and having the network use such acknowledgement to stop forwarding the bundle. Jain, Fall and Patra [8] propose routing optimizations enabled by greater knowledge that might exist about the network (average or actual schedule of contacts, queuing at each node, or traffic demand). Jones, Li and Ward [10] propose using epidemic routing for flooding information about link states and average waiting times for contacts, and using such information for recomputing routes at each contact. Musolesi, Hailes and Mascolo [11], in contrast, use a distance-vector algorithm to improve on epidemic routing. None of these approaches can guarantee that a bundle will be eventually be delivered to its destination. Preexisting node mobility may be such that delivery is not possible. Therefore, these approaches may not be suitable for MANETs with leader.

There have been several other proposals employing dedicated mobile gateways for forwarding bundles. Zhao and Ammar's original message-ferrying approach [12] optimizes the mobile gateway's schedule so as to satisfy bandwidth requirements of communication among static nodes. Somasundara, Ramamoorthy and Srivastava consider instead how to satisfy buffering constraints [15]. Zhao, Ammar and Zegura generalize [12] to the case of multiple mobile gateways [14]. They also examine tradeoffs when communicating nodes are mobile, such as whether to have such nodes move into contact with a mobile gateway or use instead long-range radio [13]. However, in emergencies, dedicated gateways may be unavailable. Li and Rus [17] propose instead general-purpose mobile gateways that give strictly higher priority to forwarding than to application-layer tasks, after a wait period. Our simulations suggest that performance may be significantly improved by scheduling forwarding and application-layer tasks in a more integrated fashion, as we propose, especially if mobile gateways can also process application-layer tasks [17] or are dedicated to forwarding but require personnel or other resources that are also needed for application-layer tasks.

VI. CONCLUSIONS

Natural or man-made disasters can disrupt communications, making rescue and recovery missions more difficult. MANETs have great potential for providing connectivity in such situations, but ordinarily cannot route packets across network partitions, which are common in emergencies. Previous DTN routing protocols can overcome this limitation under certain conditions, such as favorable node mobility patterns or availability of dedicated mobile gateways. We propose a new, cross-layer DTN routing approach for MANETs with leader, such as typical emergency-response teams. In our approach, the leader schedules workers' tasks considering both application- and network-layer needs. Thus, a worker may be assigned not only application-layer tasks, but also *courier* tasks, whose primary purpose is to provide forwarding needed for the network's operation. Our simulations show that cross-layer DTN routing can result in fewer missed deadlines and less distance traveled (and consequently, greater network uptime) than DTN routing algorithms that allocate or schedule mobile gateways in disregard of application-layer load.

REFERENCES

- [1] The Secure-CITI Project. [Online] <http://www.cs.pitt.edu/S-CITI>
- [2] D. Johnson. "Routing in Ad Hoc Networks of Mobile Hosts," in *Proc. WMCSA'94*, IEEE, 1994, pp. 158-163.
- [3] C. Perkins and E. Royer. "Ad-Hoc On-Demand Distance Vector Routing," in *Proc. WMCSA'99*, IEEE, 1999, pp. 90-100.
- [4] Z.J. Haas and M.R. Pearlman. "The Performance of Query Control Schemes for the Zone Routing Protocol," in *Proc. SIGCOMM'98*, ACM, 1998.
- [5] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott and H. Weiss. "Delay-Tolerant Networking: An Approach to Interplanetary Internet," in *Communications Magazine*, IEEE, June 2003, pp. 128-136.
- [6] K. Fall. "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. SIGCOMM'02*, ACM, Aug. 2003, pp. 27-34.
- [7] A. Vahdat and D. Becker. "Epidemic Routing for Partially-Connected Ad Hoc Networks," Duke Tech Report CS-2000-06, 2000.
- [8] S. Jain, K. Fall and R. Patra. "Routing in a Delay Tolerant Network," in *Proc. SIGCOMM'04*, ACM, Aug. 2004.
- [9] K. Harras, K. Almeroth and E. Belding-Royer. "Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks," in *Proc. Networking'2005*, IFIP, Springer-Verlag, LNCS 3462:1180-1192, May 2005.
- [10] E. Jones, L. Li and P. Ward. "Practical Routing in Delay-Tolerant Networks," in *Proc. SIGCOMM'05 Workshops*, ACM, 2005.
- [11] M. Musolesi, S. Hailes and C. Mascolo. "Adaptive Routing for Intermittently Connected Ad Hoc Networks," in *Proc. 6th Intl. Symp. World of Wireless, Mobile and Multimedia Networks (WOWMOM'05)*, IEEE, June 2005.
- [12] W. Zhao and M. Ammar. "Message Ferrying: Proactive Routing in Highly-Partitioned Wireless Ad Hoc Networks," in *Proc. Workshop Future Trends in Distributed Computer Systems*, IEEE, May 2003.
- [13] W. Zhao, M. Ammar and E. Zegura. "A Message Ferrying Approach for Data Delivery in Sparse Ad Hoc Networks," in *Proc. MobiHoc'04*, ACM, May 2004.
- [14] W. Zhao, M. Ammar and E. Zegura. "Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network," in *Proc. INFOCOM'2005*, IEEE, Mar. 2005.
- [15] A. Somasundara, A. Ramamoorthy and M. Srivastava. "Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines," in *Proc. RTSS'2004*, IEEE, 2004.
- [16] NS-2. "The Network Simulator - ns-2." [Online] <http://www.isi.edu/nsnam/ns/>
- [17] Q. Li and D. Rus. "Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks," in *Proc. MobiCom 2000*, ACM, Aug. 2000.