

Incentive-Aware Routing in DTNs

Upendra Shevade Han Hee Song Lili Qiu Yin Zhang

The University of Texas at Austin

{upendra, hhsong, lili, yzhang}@cs.utexas.edu

Abstract—Disruption tolerant networks (DTNs) are a class of networks in which no contemporaneous path may exist between the source and destination at a given time. In such a network, routing takes place with the help of relay nodes and in a store-and-forward fashion. If the nodes in a DTN are controlled by rational entities, such as people or organizations, the nodes can be expected to behave selfishly and attempt to maximize their utilities and conserve their resources. Since routing is an inherently cooperative activity, system operation will be critically impaired unless cooperation is somehow incentivized. The lack of end-to-end paths, high variation in network conditions, and long feedback delay in DTNs imply that existing solutions for mobile ad-hoc networks do not apply to DTNs.

In this paper, we propose the use of pair-wise tit-for-tat (TFT) as a simple, robust and practical incentive mechanism for DTNs. Existing TFT mechanisms often face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism that incorporates generosity and contrition to address these issues. We then develop an incentive-aware routing protocol that allows selfish nodes to maximize their own performance while conforming to TFT constraints. For comparison, we also develop techniques to optimize the system-wide performance when all nodes are cooperative. Using both synthetic and real DTN traces, we show that without an incentive mechanism, the delivery ratio among selfish nodes can be as low as 20% as what is achieved under full cooperation; in contrast, with TFT as a basis of cooperation among selfish nodes, the delivery ratio increases to 60% or higher as under full cooperation. We also address the practical challenges involved in implementing the TFT mechanism. To our knowledge, this is the first practical incentive-aware routing scheme for DTNs.

I. INTRODUCTION

Disruption tolerant networks (DTNs) are a class of networks in which no contemporaneous path may exist between the source and destination at a given time. Different from traditional networks, data in DTNs are opportunistically routed toward the destination by leveraging temporary connection.

When nodes in a DTN are controlled by rational entities, such as people or organizations [1], [3], [5], they can behave selfishly and try to only maximize their own utility without considering the system-wide criteria. A selfish user may drop others' messages and excessively replicate its own messages to increase its own delivery rate while significantly degrading other users' performance or even cause starvation. Since DTNs have limited connectivity, if any, simply removing selfish nodes results in serious performance penalty. Therefore it is necessary to design incentive-aware routing for DTNs in order to fully take advantage of temporary connections.

While there has been considerable work on studying selfish behavior and designing incentive-aware routing schemes (*e.g.*, [10], [13], [14], [15], [17], [21]), to our knowledge, this paper

is the first one that studies these issues in DTNs. The lack of contemporaneous path, high variation in network conditions, difficulty to predict mobility patterns, and long feedback delay make the problem very different from the traditional networks like Internet and mobile ad hoc networks. Therefore the existing solutions do not directly apply.

In this paper, we first study the impact of selfish behaviors in DTNs. Using simulation based on both synthetic and real mobility traces, we show that the presence of selfish users can degrade total delivered traffic to less than 20% as what can be delivered under full cooperation.

Motivated by the significant damage caused by selfish users, we propose the use of pairwise tit-for-tat (TFT) as a simple, robust, and practical incentive mechanism for DTNs. Existing TFT mechanisms often face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism that incorporates generosity and contrition to address these issues. We then develop an incentive-aware routing protocol that allows selfish nodes to maximize their individual utilities while conforming to TFT constraints. We also address the practical challenges involved in implementing the TFT mechanism. To the best of our knowledge, this is the first practical incentive-aware routing scheme for DTNs. We evaluate the effectiveness of our incentive-aware routing scheme using both synthetic and real DTN traces. Our results show that with TFT as a basis of cooperation among selfish nodes, the total delivered traffic increases to 60% or higher as under full cooperation.

Our main contributions can be summarized as follow.

- We study the impact of selfish behavior in DTNs and show that it results in serious performance degradation.
- We develop a practical incentive-aware routing scheme based on the TFT mechanism for selfish users to optimize their own performance without significant degradation of system-wide performance.
- We demonstrate the effectiveness of our incentive-aware routing scheme using trace-driven simulation.

The rest of the paper is organized as follow. In Section II, we survey related work. We motivate the incentive-aware routing problem for DTNs in Section III. We present cooperative DTN routing in Section IV and selfish DTN routing in Section V. We describe our evaluation methodology in Section VI and performance results in Section VII. We conclude in Section VIII.

II. RELATED WORK

DTN routing. Routing decisions in DTNs must be made in the absence of end-to-end contemporaneous paths and with limited

and possibly stale information about the network. There is a large body of work on routing in DTNs. Most existing schemes are incidental in nature [1]: they do not explicitly optimize a specific performance metric, but opportunistically routes data when temporary connection becomes available. For instance, in epidemic routing [22], whenever two nodes meet, they exchange all messages that the other does not have. Since epidemic routing is essentially flooding, its overhead is high. To reduce the overhead, utility-based replication has been proposed, where nodes replicate data over the best contacts according to some utility (*e.g.*, based on previous mobility [11]). However the effect of how utility relates to the desired performance metrics is unclear. In contrast, RAPID [1] explicitly tries to optimize system-wide metrics such as average delay while incorporating resource constraints, and is shown to be highly effective. Motivated by RAPID, our routing protocol also explicitly optimizes user performance based on the network conditions. However, in contrast to RAPID, which considers only mean link delays, our protocol considers both the mean and variance of link delays and as a result, is robust against high variability in link characteristics. In addition, RAPID only considers global performance objective, whereas our work explores the effects of selfish users in the system.

Incentive mechanisms. Cooperation in the presence of selfish agents has been extensively studied in the Internet, mobile ad-hoc networks, wireless mesh networks, and peer-to-peer applications. Most existing work falls into one of the following three categories. The first category attempts to identify misbehaving nodes and isolate them from the network [13]. These protocols usually assume a set of trusted nodes that can detect and verify misbehavior that results in the selfish node being denied participation in the network. The fear of detection and punishment motivates nodes to cooperate. The second category is based on credits, where nodes earn credits by forwarding packets. These credits can then be used to obtain forwarding service from any node in the system. However, existing credit-based protocols require either secure hardware [4] or trusted centralized banks [26]. In the third category of solutions, nodes reciprocate good or bad behavior on part of the peer in a tit-for-tat fashion [15], [10], [21]. A node autonomously lowers service to a neighbor if it detects that the neighbor is misbehaving, and fully cooperates with the neighbor if no misbehavior is detected. This leads to partial and probably temporary isolation of misbehaving nodes.

We choose tit-for-tat (TFT) as the incentive mechanism for DTN routing. In TFT, every node forwards as much traffic for a neighbor as the neighbor forwards for it. In this way, rather than attempting to detect misbehavior, our approach focuses on detecting good behavior. In our solution, packet acknowledgement acts as the proof of work done by a next-hop. This positive feedback allows a node to engage in balanced exchange with its neighbors—rewarding good behavior with equal reciprocal service and ignoring “misbehavior”. In case of the punishment-based approaches, the strong reaction to misbehavior (isolation of the node by peers) is justified

only if we have high confidence that the mechanism has very low false positives, so that innocents are not punished, and very low false negatives, so that miscreants cannot get away by flying under the radar. In case of DTNs, there are no reliable ways to detect misbehavior. The existing watchdog mechanism [14] designed for ad hoc networks cannot work in DTNs since it assumes that the sender can listen for the next hop’s transmissions to detect if the next hop properly forward the traffic and this assumption fails to hold in DTNs since these two nodes are often disconnected. In addition, due to large variability in mobility patterns and network condition, a node may not be able to deliver a packet within its target deadline in spite of its best intentions. Therefore, the potentially high false positive and false negative in detecting selfish nodes render punishment-based scheme unsuitable for DTNs. In addition, the TFT-based incentive mechanism does not require trusted nodes or special hardware, which fits well with decentralized and low-cost DTNs we envision.

Previous papers (*e.g.*, [10], [15], [21]) have laid the game-theoretic foundation for the use of TFT in wireless networks. The existing TFT mechanisms face bootstrapping problems or suffer from exploitation. We propose a TFT mechanism with generosity and contrition to address these issues. Furthermore, our protocol tolerates significantly large feedback delay in DTNs and supports multi-hop paths (as opposed to single-hop paths in [21]).

TFT has been particularly successful in combating free-riding behavior in P2P file sharing systems. TFT is used to ensure that only agents who actively contribute are allowed to download files from others. Bit-torrent [7] employs a variant of TFT, where k top-performing neighbors in an interval are given equal download rates in the next interval. Analyzing this strategy and improving upon it have been the focus of several recent papers([16], [12]). TFT based file sharing is different from TFT-based routing in DTN in the following ways. First, file sharing is a purely bilateral transaction between two nodes, while routing typically involves interactions among multiple relaying nodes, thus complicating analysis. Second, DTN has large feedback delay and high uncertainty, which makes it critical to address bootstrapping and exploitation. Third, the bilateral nature of file sharing also implies that a neighbor’s performance can be evaluated directly, while we must rely on end-to-end acknowledgements to do the same in case of DTN routing. Due to these important differences, we cannot directly apply the TFT mechanism for file sharing to DTN routing.

III. MOTIVATION

In this section, we motivate the need for incentive mechanisms by demonstrating that network performance incurs serious degradation without an incentive mechanism. We then identify several important research questions on incentive-aware routing in the DTN context.

The need for incentive mechanism. We compare the performance of fully cooperative DTNs with DTNs consisting of only selfish nodes. Results for the fully cooperative network are obtained by formulating the DTN routing problem as an

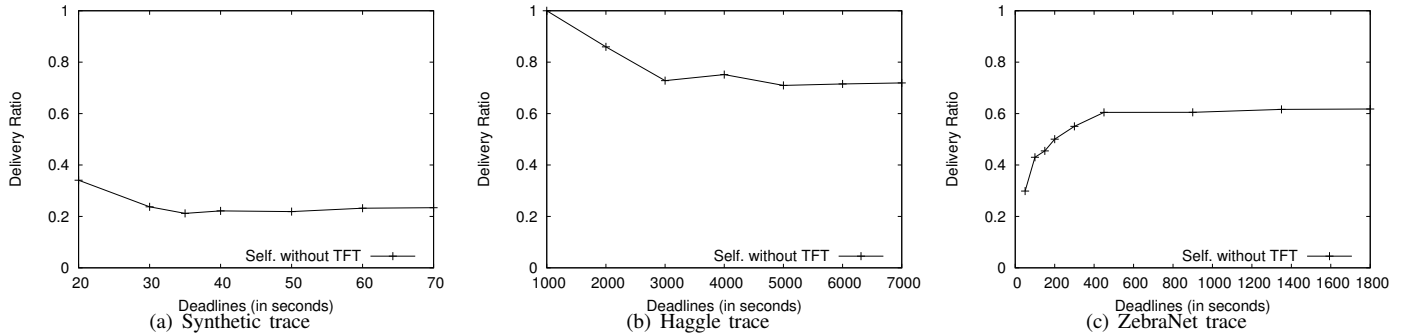


Fig. 1. Comparison of fully cooperative DTN against non-cooperative DTN, where the y-axis shows the ratio between delivery rate under a selfish DTN and that under a cooperative DTN.

Linear Program, which is solved using the CPLEX solver [8]. In the absence of any incentive mechanism to facilitate cooperation, the rational strategy is to free-ride if possible and not relay traffic for anyone else, since there is nothing to be gained by doing so and no notion of penalty in not relaying. If all the nodes in the network follow this strategy, everyone drops relay requests from others implying that packet delivery is only possible when the source directly meets the destination.

We use the fraction of packets delivered within deadline as the performance metric. As shown in Figure III, the delivery rate reduces to only 20% of what is achieved under full cooperation in the synthetic trace, to only 70% in the Haggie trace [18], and to only 60% in the ZebraNet trace [23]. The higher delivery rate in Haggie is due to its higher network connectivity.

These results further confirms our intuition and shows that unless cooperation is somehow incentivized, system operation will be critically impaired. A network of selfish entities needs an incentive mechanism that can act as a basis for such cooperation. As justified in Section II, in this paper we choose TFT as the incentive mechanism because it fits well to the unique characteristics of DTNs, such as lack of contemporaneous path, large feedback delay, high variation in network conditions, and unpredictable mobility.

Research questions. In the rest of this paper, we focus on understanding the following two “prices” in the context of incentive-aware DTN routing.

- **The price of anarchy (PoA)** — Under the given incentive mechanism, what is the performance that can be achieved when all DTN nodes are selfish relative to the optimal performance that can be achieved when all DTN nodes are cooperative?
- **The price of incentive mechanism (PoI)** — Assuming that all users are cooperative, what is the performance penalty that can be achieved under a given incentive mechanism relative to the optimal performance without the incentive mechanism?

PoA and PoI are complementary with each other: PoA quantifies the effectiveness of an incentive mechanism in limiting the damage of selfish nodes, whereas PoI quantifies the performance loss of cooperative nodes due to the presence

of the incentive mechanism. In this paper, we show that with our design, TFT can achieve both low PoA and low PoI for DTN routing.

IV. COOPERATIVE DTN ROUTING

We first study the following two cooperative DTN routing schemes. The first scheme optimizes the global objective when everyone is cooperative. This is an interesting baseline since it provides an upper-bound of the performance under TFT constraints. The second scheme optimizes the global objective under TFT constraints. By comparing the performance of these two schemes, we can estimate the PoI of TFT. In the next section, we consider DTN routing under TFT constraints when nodes are selfish.

Routing objective. Throughout this paper, we consider maximizing total delivered traffic within a given deadline. This is a natural and useful optimization objective. While DTN applications tend to be much more tolerant to delays, it is often useful to be able to impose some application-specific deadline (as opposed to waiting for the delivery for ever). When the deadline is equal to infinity, the objective translates to maximizing the total delivered traffic. Finally, although we only present results for this optimization objective, our algorithms can directly support other optimization objectives, such as minimizing total delay.

A. Global Optimal

In this paper, we consider the delivery ratio within a given deadline as the performance metric. The problem of maximizing total delivery ratio within a given deadline over all flows can be solved in the following four steps.

Candidate path generation. First, we generate a set of candidate paths for each given flow by enumerating all possible paths between its source and destination that have at most 3 hops (similar to RAPID [1]). By limiting the length of candidate paths, the number of candidate paths for each flow is bounded by $O(n^2)$, where n is the total number of nodes.

Path performance computation. Next, for each path of a flow, we compute the delivery ratio within a given deadline if the flow is routed through this path. Since the inter-contact

time may not follow any well-known distribution, for generality we compute a lower bound of the delivery ratio using Chebyshev's Inequality [24], which holds for any distribution.

Specifically, we first compute the mean and variance of the waiting time on each link as follows. Given a link between two nodes, we break time into ON periods (in which the two nodes are in contact) and OFF periods (in which the two nodes are not in contact). We assume that if a packet arrives during an ON period, it can be delivered immediately (*i.e.*, the waiting time is 0); if a packet arrives during an OFF period, it can be delivered at the beginning of the next ON period (*i.e.*, the waiting time is equal to the residual time in the current OFF period). For simplicity, we ignore the propagation delay during ON periods because it is typically much smaller than the duration of OFF periods. Assuming that the packet arrival time is uniformly distributed, we can then compute the mean and variance of the waiting time on this link.

We can then approximate the delivery ratio (within a given deadline) for an end-to-end path as follows. For any given path, let random variable X denote the total waiting time. Let μ and σ^2 denote the mean and variance of X . μ and σ^2 can be computed by summing up the mean and variance of the waiting times on different links on this path (under the assumption that these waiting times are independent). Let D denote the desired deadline. Then according to Chebyshev's Inequality, the delivery ratio within the deadline D can be bounded as follows:

$$\begin{aligned} Pr(X \leq D) &= 1 - Pr(X \geq D) \\ &= 1 - Pr(X - \mu \geq D - \mu) \\ &\geq 1 - \left(\frac{\sigma}{D - \mu} \right)^2 \end{aligned} \quad (1)$$

Route optimization. Then we maximize the total delivery ratio within the deadline for all flows by formulating the problem as the linear program (LP) shown in Figure 2. Here $X_{f,i}$ is the traffic allocation of flow f on path i ; $P_{f,i}$ is the lower bound of the delivery ratio when traffic of flow f is routed through path i given by Inequality (1); Cap_i denotes the smallest capacity of all links on path i . Constraint C1 specifies the capacity constraint, *i.e.*, the total amount of traffic routed through path i should not exceed the capacity of path i . Constraint C2 mandates that the total traffic assignment for flow f does not exceed the demand of flow f by a factor of $RepFactor$, where the replication factor $RepFactor$ is a control parameter that can be tuned to improve the total delivery ratio at the cost of more replication traffic. In our evaluation, we keep $RepFactor$ constant at 3.

Online optimization. Finally, to deal with fluctuations in traffic demands and network connectivity, we break time into segments and perform route optimization at the beginning of each segment based on predicted traffic demands and path performance. Specifically, we use the exponentially weighted moving average (EWMA) of past values to predict the traffic demands and the mean and variance of path waiting time for the current segment. We then use these predicted values in the

Input : $Flows, Demand(f), P_{f,i}, Cap_i$

Output : $X_{f,i}$

$$\mathbf{max} : \sum_{f \in Flows} \sum_i X_{f,i} P_{f,i}$$

Subject to:

$$\begin{aligned} [C1] \quad & \sum_{f \in Flows} X_{f,i} \leq Cap_i \quad \forall i \\ [C2] \quad & \sum_i X_{f,i} \leq RepFactor * Demand(f) \quad \forall f \end{aligned}$$

Fig. 2. LP formulation to maximize delivered traffic within a given deadline.

above LP formulation to optimize the routes for the current segment. The routes remain unchanged until the beginning of the next segment.

B. Global Optimal with TFT Constraints

Next we incorporate the TFT mechanism when maximizing total delivered traffic. This can be achieved by adding the following TFT constraints into the LP shown in Figure 2. The TFT constraints simply state that the total amount of traffic through link $A \rightarrow B$ is equal to the total amount of traffic in the opposite direction (*i.e.*, through link $B \rightarrow A$).

$$\sum_f \sum_{i: AB \in Path_i} X_{f,i} = \sum_f \sum_{j: BA \in Path_j} X_{f,j} \quad \forall nodes A, B \quad (2)$$

TFT constraints with generosity: The basic TFT constraints (as described above) have problems with bootstrapping. When two nodes meet for the first time, since no packets have ever been successfully relayed by the other node, the basic TFT prevent any relaying. To address this issue, generous TFT enables initial cooperation of up to ϵ , which allows a node to send ϵ number of packets more than it has earned the right to send by relaying in the previous interval. Generous TFT is also useful for handling asymmetric traffic demands by absorbing traffic imbalance up to ϵ amount. The ϵ value is important. A larger value loosens TFT constraints and yields better performance and faster bootstrapping when everyone is cooperative. On the other hand, it also means that a selfish node is less cooperative since this generosity can be exploited by the node. We model exploitation as follows. Every selfish node checks if it has performed enough work in the previous interval to be able to satisfy its predicted demand for the upcoming interval without requiring any generosity from the neighbor. If not, it has incentive to provide generosity in order to get increased service in the next interval. Otherwise (*i.e.*, if its predicted service rate is no less than its predicted demand), it has no incentive to provide any generosity to that neighbor. In addition, it assumes that the neighbor will provide it generosity. As a result, it does ϵ less work and can get away with it if the other node indeed does provide ϵ generosity. In Section VII, we empirically study the impact of generosity.

TFT constraints with generosity and contrition: Generosity alone is still insufficient. While it can absorb transient asymmetry in delivery of up to ϵ , any imbalance exceeding ϵ

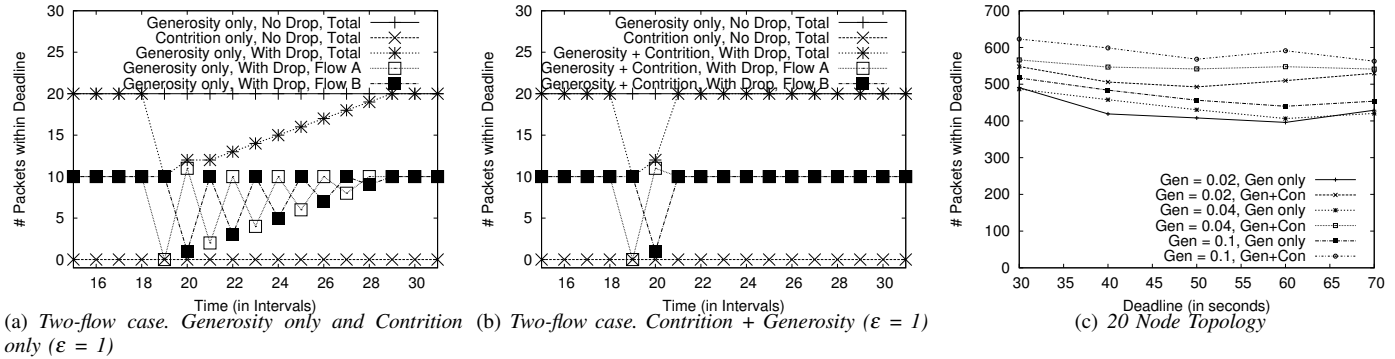


Fig. 3. Effect of Generosity, Contrition and Generosity + Contrition. ($\epsilon = 1$)

could lead to lengthy retaliation between two neighbors. This is illustrated in the following example.

Consider a toy topology of four nodes with two flows $A \rightarrow B \rightarrow C$ and $B \rightarrow A \rightarrow D$, each with a demand of 10 packets/interval. When network connectivity is stable, generosity allows relays A and B to gradually increase their cooperation until their demands are satisfied after which both will attempt to exploit the other. However, generous TFT can cause protracted vendetta between neighbors if one of them delivers less packets than expected for the other, possibly due to variation in mobility. This is demonstrated in Figure 3 (a), where outage on link $B \rightarrow C$ in interval 19 causes zero packets to be delivered for A 's flow in that interval. Since this drop in delivery exceeded ϵ , A retaliates in the next interval by delivering correspondingly less. In interval 21, generous TFT constraints require B to retaliate back, which sets off a long-lasting vendetta.

Contrition solves this problem by refraining from reacting to a valid retaliation to its own mistake. With contrite TFT, B realizes that A 's reaction in interval 20 was due to its own actions in interval 19, and so does not lower traffic in interval 21. This way cooperation is restored from interval 21. In addition, since contrition does not always provide leeway like generous TFT, it cannot be exploited. However, contrition cannot work by itself either in case of DTN routing. Contrition only provides a way to return to stability after perturbation, but provides no way to reach that stability, *i.e.*, as Figure 3 (a) shows, it does not provide any way for cooperation to bootstrap like generosity does.

Hence, we propose a novel variant of TFT by combining generosity and contrition. The generosity component enables bootstrapping and absorbs transient asymmetries, while contrition prevents mistakes from causing endless retaliation. Figure 3 (b) shows that by interval 21, contrition and generosity working together allow total delivery to recover from the outage in interval 19.

In Figure 3 (c), we show that the above argument remains valid even for a larger topology of twenty nodes. We compare the number of packets delivered for different values of generosity and different packet deadlines. Packet delivery increases with increasing generosity as TFT conditions are loosened to accommodate asymmetric demand. More

importantly, TFT with both generosity and contrition prevents unnecessary retaliation and outperforms TFT with generosity alone by up to 30% in some cases.

V. SELFISH DTN ROUTING

The previous section describes methods for optimizing routes when all nodes in a DTN are cooperative. In this section, we present a practical distributed protocol that allows selfish users to optimize their individual performance while conforming to TFT constraints.

Our routing protocol consists of the following three components: (i) every node periodically exchanges link state, (ii) each source computes the forwarding paths based on link state and uses source routing to send its traffic, (iii) upon receiving data, each destination sends ACK via flooding and the source uses it to update its TFT constraints for the next interval.

Link state dissemination: Every node keeps track of the mean and variance of the waiting time on links between the node itself and other nodes. It also computes the link capacity using the duration of the meeting and the bandwidth available during that time. At the end of every interval, every node floods these three link metrics so that all nodes have the information about all links in the network. This is similar to many link state protocols, such as OSPF. We assume that link state is disseminated faithfully—we focus on making the data-plane incentive compatible and leave securing the control plane to future work.

Route computation: We use source routing to send traffic. This gives a source complete routing control so that it can directly optimize its own performance metric. Moreover, if different senders are interested in optimizing different performance metrics (*e.g.*, some want to minimize delay and others want to maximize delivery rate), this can be easily supported using source routing. In order to prevent the source route from being tampered in transit, it is digitally signed by the sender (*e.g.*, using Hierarchical Identity Based Cryptography (HIBC) [19], which is shown to be practical for DTNs [20]).

Based on the link state, a source maximizes its total delivered traffic as follow. For each data packet, a source generates *RepFactor* number of packets and specifies complete source route for each generated packet. The routing strategy is computed at the beginning of every interval. Given average delay,

variance, and link capacity of each link (which is disseminated throughout the network), a source first computes the delivery ratio within a given deadline for each path using Equation 1. The end-to-end ACKs (as described below) indicate how many packets are successfully delivered on each path for each flow during the previous interval. Then it can compute the total background traffic along each path in the previous interval, and estimate the background traffic volume information in the new interval using ACK packets of data delivery. Next it updates its routing strategy in the new interval by moving traffic from the worst path to the best path in terms of delivery ratio. The move continues until either link capacity constraint or TFT constraint is violated. Then it starts to move traffic to the second best path and so on until all the paths with better delivery rate have reached their raw link capacity or TFT constraints. Figure 4 outlines the algorithm.

```

1  src: node ID of flow  $f$ 's source
2   $X_{f,i}^{prev}$ : amount of traffic from flow  $f$  is allocated on path  $i$  in the previous interval
3   $X_{f,i}^{curr}$ : amount of traffic from flow  $f$  is allocated on path  $i$  in the current interval
4   $B_{f,i}^{prev}$ : amount of background traffic on path  $i$  in the previous interval
5  compute the lower-bound of the delivery ratio along each path using Equation 1
6  sort paths in the order of increasing delivery ratio
7   $worst = 1$ ;  $best = totalPaths$ 
8  while  $P(worst) < P(best)$ 
9      // compute maximum amount of flow  $f$ 's traffic that can be sent on path  $i$ 
10      $TFTCap(best) = \infty$ ;
11     for each relay link  $k$  on path  $best$ 
12          $TFTCap(best) = \min(TFTCap(best), forwarded(k))$ ;
13     end
14      $cap = \min(TFTCap(best), Cap(best)) - B_{f,i}^{prev}$ ;
15     // we can move up to  $moveTotal$  traffic from the worst to best path
16      $moveTotal = \min(X_{f,worst}^{prev}, cap)$ ;
17      $X_{f,worst}^{curr} - = moveTotal$ ;
18      $X_{f,best}^{curr} + = moveTotal$ ;
19     if  $X_{f,best}^{curr} == cap$ 
20          $best = best - 1$ ;
21     end
22     if  $X_{f,worst}^{curr} == 0$ 
23          $worst = worst + 1$ ;
24     end
25 end

```

Fig. 4. Route computation at selfish nodes.

Note that $TFTCap$ in Figure 4 is based on the total traffic others have forwarded in the previous interval, denoted as $forwarded(k, src)$. However, as all TFT-based schemes, the actual TFT constraints should be based on the total traffic sent during the current interval. Moreover the background traffic along each path may also change over different time intervals. Therefore the route derived above may not satisfy the actual TFT constraints. To address the issue, we apply the following dropping strategy. Let $T_{i,j}$ and $T_{j,i}$ denote the total traffic node i relays for j and the total traffic node j relays for i in the current interval. If $T_{i,j} > T_{j,i} + \epsilon$, node i ensures TFT by dropping traffic from j that exceeds $T_{j,i} + \epsilon$ packets.

ACK dissemination: Upon receiving a packet, the destination floods an ACK so that we can use it to derive the TFT constraints for the next interval. In our protocol, acknowledgements serve the following three purposes.

- First, packet acknowledgements generated by the destination act as proofs of successful relay by intermediate nodes. Once the packet reaches its destination, the desti-

nation node extracts the source route and the accompanying signature and attaches them to the acknowledgement packet. The ACK packet is then flooded through the network. The size of ACK is much smaller than the size of data traffic, so the ACK overhead should be small. To further improve efficiency, we can combine multiple ACK packets into a single ACK during the flooding.

- Second, acknowledgements can provide useful feedback required by TFT. Specifically, every node receiving the ACK first verifies the integrity of the attached source route and then checks if its identifier is present in the relay list. If it is, then the node increments its local TFT counters to indicate that the next node in the list successfully relayed a packet for it. Credit is only given to relay nodes on the forwarding path.
- Third, flooded acknowledgements disseminate key information about the network operation to every node. Each node uses information provided in the ACK to compute how many packets were sent between every source and destination pair and along which paths. Since the ACK contains the entire source route, a node can also calculate the number of packets traversing every link in both directions.

VI. EVALUATION METHODOLOGY

We implement the routing protocol described in Section V in dtmsim from [9]. For comparison purpose, we also evaluate the routing computed by solving LP as described in Section IV using CPLEX [8]. We compare different routing schemes by varying mobility traces and traffic demands. In all evaluation, we set link capacity to 10 packets/second.

Mobility traces: We use synthetic mobility traces to gain insights to the routing schemes under controlled scenarios, and use real traces to assess their performance under realistic scenarios. We generate synthetic traces as follows. We have 20 nodes and randomly create 114 links among them. We then generate the ON/OFF time for each link, where the ON time is kept constant at 1, and the OFF time follows a Gaussian distribution whose mean and variance are 10 and 0.5, respectively. In addition to the synthetic traces, we also use the Huggle trace [18] which involves 41 iMotes carried by IEEE INFOCOM attendees and the ZebraNet trace consisting of the movement of 20 male zebras in a 6km-by-6km field each carrying a radio with a range of 500m, generated using the same methodology as [23].

Traffic demands: To study the impact of traffic demands on the routing performance, we generate traffic demands as follows. We first randomly generate 5 flows originating from each node. We then set the total traffic demands from all nodes to be either all equal or following a Zipfian distribution. In Zipfian distribution, the top i -th demand from a node is proportional to $1/i$. We use Zipfian distribution, because a number of studies show that realistic user demands often exhibit Zipf-like distributions [2], [6]. Finally, we partition the total traffic demand at a node to all its flows either equally or

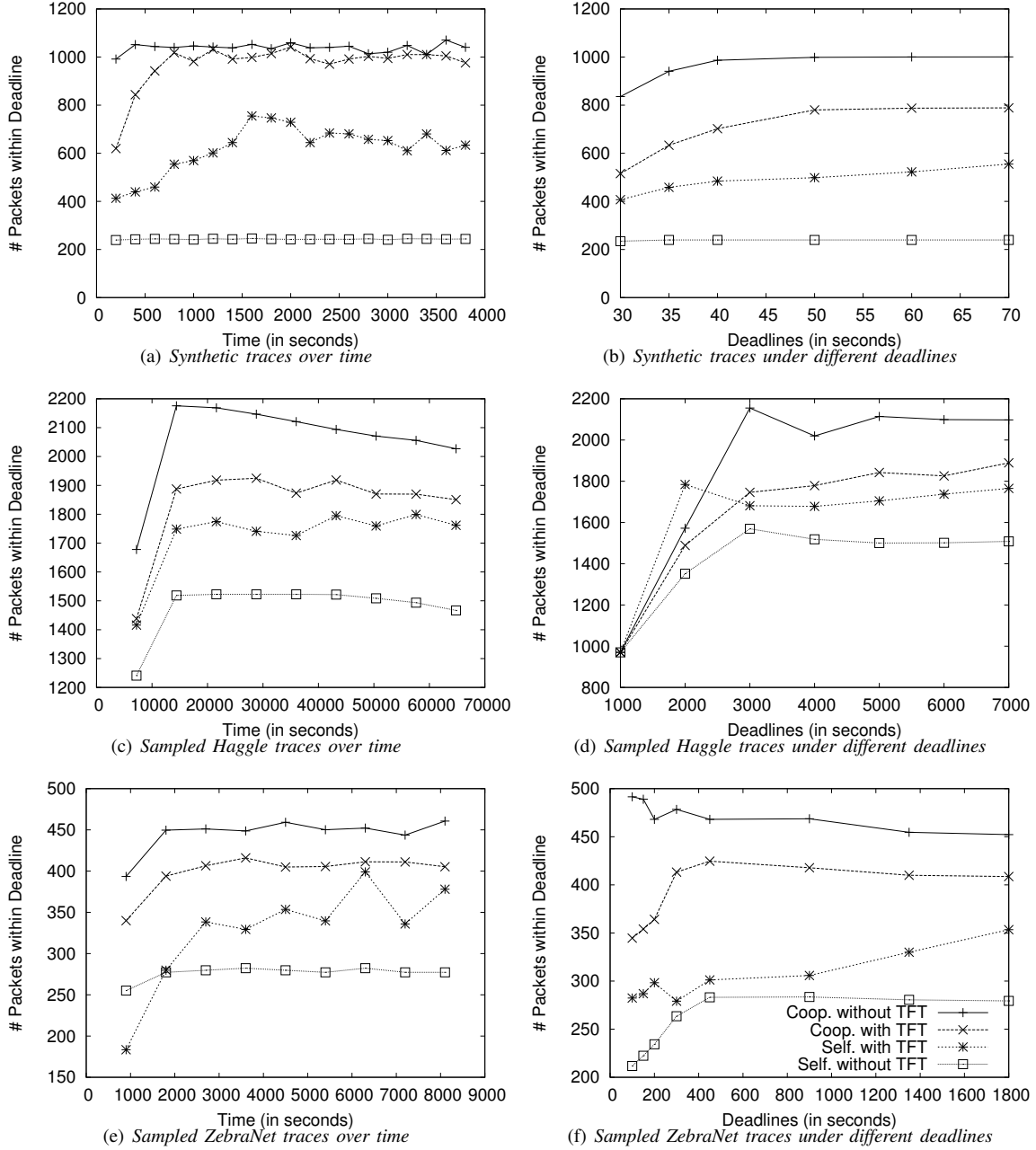


Fig. 5. Impact of different traces.

using Gravity model [25]. In Gravity model, the total traffic from A to B is proportional to the total outgoing traffic from A and the total incoming traffic to B . In this way, we have four spatial distributions: (i) *equal/equal*, (ii) *equal/gravity*, (iii) *Zipf/equal*, and (iv) *Zipf/gravity*.

VII. EVALUATION RESULTS

We evaluate the performance of our routing scheme by varying the mobility, traffic demands, and deadlines.

Impact of different traces: First, we focus on one time segment in each trace and assume that we know the mean and variance of ON/OFF time of links between every pair of nodes during the time segment. In the later evaluation, we will

consider the effects of prediction errors.

Figure 5(a), (c), and (e) plot the total number of delivered packets within the deadline over time for the synthetic, Huggle, and ZebraNet traces, respectively. We set the deadline to be 70 seconds for the synthetic trace, 7000 seconds for Huggle trace, and 1350 seconds for ZebraNet trace. As we would expect, cooperation without TFT achieves the highest delivery rate, since there is neither PoA nor PoI. Cooperation with TFT performs the second best and its difference from that without TFT is within 7-20% for all traces, which suggests that the TFT incentive mechanism imposes low cost in the presence of cooperation and therefore has low PoI: PoI of 20% for synthetic, 10.5% for Huggle, and 7% for ZebraNet trace. The

performance of selfish users with TFT also achieves low PoA compared to cooperative counterpart: PoA of 25%, 6%, and 15% respectively. Finally, we observe selfish users without TFT performs the worst, because without an incentive mechanism delivery is only possible when source and destination directly meet. We note that in Figure 5(e) Selfish with TFT performs worse than selfish without TFT in the first interval. This is because the latter delivers only packets destined for the node in contact, while the former also spends bandwidth forwarding multi-hop packets over a contact. This causes fewer packets to be actually delivered to their destinations in the first interval but helps set stage for TFT cooperation in later intervals leading to higher delivery.

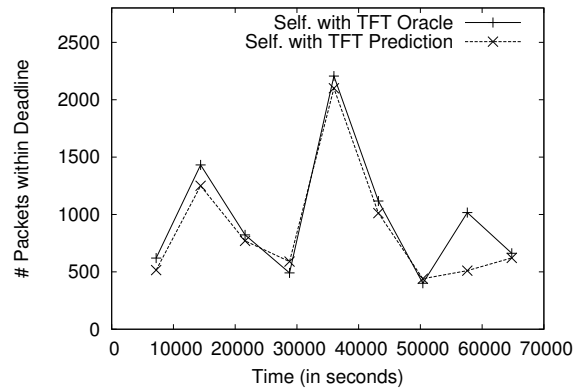
We demonstrate the effect of deadlines in Figure 5(b), (d), and (f), which plot the average number of packets delivered within the deadline. The results are the average over all intervals after the bootstrap phase is over, so Figure 5(b), (d), and (f) are the average of 15, 7, and 8 runs, respectively. As we would expect, packet delivery increases with the deadlines since packets have more time to reach the destination. Moreover, TFT performs well for a wide range of deadline values. The only exception is using 1000 second deadline in Hagggle trace, where all routing schemes perform similarly. This occurs because the deadline is too small to allow multi-hop delivery and the only possible delivery even under full cooperation is through direct contact.

From the above comparison of traces, we found that the performance gap among different cooperation schemes varies according to the nature of the traces. The rank of the schemes, however, remains the same regardless of the difference in the mobility of the traces, indicating that the TFT incentive mechanism is beneficial for various networks.

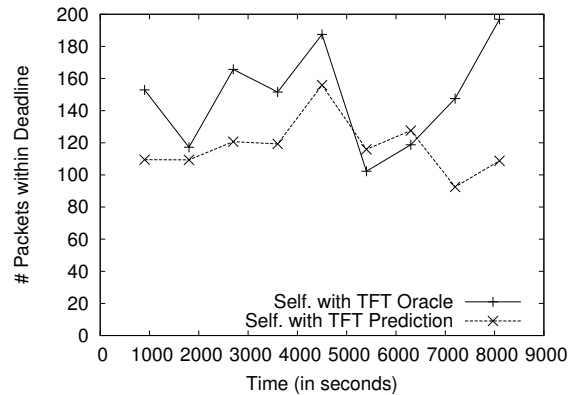
Impact of temporal variation in mobility: We now study the impact of temporal variation in mobility by using the entire raw Hagggle and ZebraNet traces. Since the Hagggle trace captures the mobility of participants at a computer science conference, it consists of periods of high interactivity (presumably indicating lunch or break times), interspersed with low activity periods (presumably indicating conference sessions). In case of the ZebraNet trace, animal movements dictate how and when communication can take place. In the presence of such variation in mobility, we need to estimate the mean, variance of delay and the bandwidth for every link in order to compute the source routes.

We compare two estimation schemes. First, we propose a prediction scheme that uses EWMA values of link characteristics as estimate for the next interval. Second, we use values provided by an oracle that has knowledge of the true values for the next interval. The oracle represents the best possible estimation.

Figure 6 compares the two estimation methods. For the Hagggle trace (Figure 6(a)), we observe that the prediction scheme performs within 10% of the oracle which is reasonably accurate given the high variation in the mobility for the three days of conference. While the ZebraNet trace results (Figure 6(b)) do not show any discernible diurnal pattern, it



(a) Hagggle trace



(b) ZebraNet trace

Fig. 6. Comparison of inter-meeting prediction versus oracle under temporal variation in mobility. EWMA parameters: $\alpha = 0.8$, EWMA interval = trace interval

also has high variability in the node's movements over time. Moreover, the EWMA interval size and the total volume of the packets are order of magnitude less than those of Hagggle traces, making EWMA predictor to follow the oracle's line with 21% error.

Comparing the oracle and prediction results, we can observe that with EWMA-based prediction, the results are reasonably close to perfect prediction. However, we note that predicting link properties is an interesting and open problem for DTNs.

Impact of spatial variation in traffic demands: Next we study the impact of spatial variation in traffic demands using the synthetic trace. Figure 7 shows the total delivery rates for different spatial distributions: equal/equal, equal/gravity, Zipf/equal, and Zipf/gravity, which are described in Section VI. Here the number of packets delivered within deadline represents an average of 15 runs. We observe that the relative performance across the different routing schemes remains the same. Moreover, the PoI under a cooperative DTN is around 5%. In addition, without the incentive mechanism, the delivery rate in a selfish DTN is only around 200 packets/second. In comparison, with the incentive mechanism, the delivery rate increases to 500 packets/second for Zipf-distributed demands, and 600 packets/second for equal demands. These results demonstrate the effectiveness of our incentive-aware routing

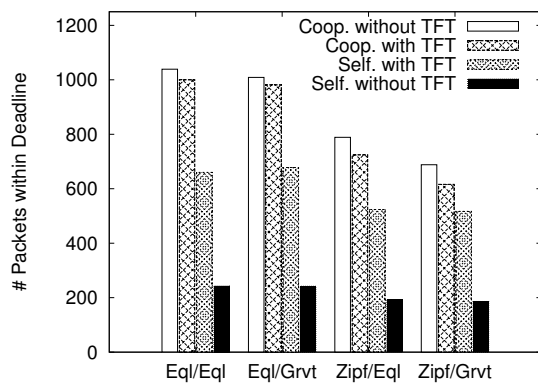


Fig. 7. Comparison of algorithms under spatial demand difference.

scheme.

Impact of temporal variation in traffic demands: Finally we compare routing schemes by varying the traffic demands over time using the synthetic trace. The goal is to understand the responsiveness of different routing schemes to demand changes, *i.e.*, whether the algorithm converges after disruption in demand traffic, and if so, how long it takes for the delivery ratio to stabilize.

We consider the following six traffic variation scenarios: (a) increasing and then stabilized demands, (b) decreasing and then stabilized demands, (c) continuously increasing demands, (d) oscillating demands, (e) spike at an interval and going back to the previous demand level, and (f) dip at an interval and going back to the previous demand level. In all the cases, every flow has the same demand (*i.e.*, their spatial distribution follows equal/equal).

As shown in Figure 8, the relative performance of different routing schemes are consistent across different temporal demand variations. The price of incentive mechanism is small, and the incentive mechanism significantly improves delivery rate in a selfish DTN. In addition, the delivery rate adapts quickly with the change in traffic demands in all cases.

VIII. CONCLUSION

In this paper, we study the impact of selfish behavior in DTNs and show that it results in serious degradation in routing performance. We then propose the use of tit-for-tat (TFT) as a simple, robust and practical incentive mechanism for DTNs. Making TFT practical for DTNs is challenging due to the lack of contemporaneous end-to-end paths, high variation in network conditions, difficult to predict mobility patterns, and long feedback delay in DTNs. Existing TFT mechanisms often face bootstrapping problems or suffer from exploitation in such environment. We therefore propose a TFT mechanism that incorporates generosity and contrition to address these issues. We then develop an incentive-aware routing protocol that allows selfish users to adaptively optimize their individual performance subject to TFT constraints. We also address the practical challenges involved in implementing the TFT mechanism. Using both synthetic and real DTN traces, we show that our incentive-aware routing protocol is

effective in fostering cooperation among selfish nodes and can significantly improve the routing performance.

In this paper, we focus on making the data-plane communication incentive-compatible. For future work, we will focus on making the control-plane exchanges incentive-compatible as well. We also plan to analyze our routing algorithm using more diverse DTN traces.

Acknowledgments

We thank Cristina Nita-Rotaru and anonymous reviewers for their helpful comments. This work is supported in part by NSF Grants CNS-0546755, CNS-0627020, and CNS-0546720.

REFERENCES

- [1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proc. of SIGCOMM*, pages 373–384, New York, NY, USA, 2007. ACM.
- [2] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *Proc. of IEEE INFOCOM*, Mar. 1999.
- [3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: routing for vehicle-based disruption-tolerant networks. In *Proc. of IEEE INFOCOM*, pages 1–11, Apr. 2006.
- [4] L. Buttyán and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proc. of MobiHoc*, pages 87–96, Piscataway, NJ, USA, 2000. IEEE Press.
- [5] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In *Proc. of IEEE INFOCOM*, Apr. 2006.
- [6] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and analysis of a streaming-media workload. In *Proc. of USITS*, Mar. 2001.
- [7] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems*, 2003.
- [8] Cplex. <http://www.ilog.com/products/cplex/>.
- [9] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Proc. of ACM SIGCOMM*, pages 145–158, New York, NY, USA, 2004. ACM.
- [10] J. J. Jaramillo and R. Srikant. DARWIN: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proc. of ACM MobiCom*, pages 87–98, New York, NY, USA, 2007. ACM.
- [11] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. of ACM ASPLOS*, Oct. 2002.
- [12] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent’s incentives. In *SIGCOMM*, 2008.
- [13] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *Proc. of NSDI*, pages 231–244, Berkeley, CA, USA, 2005. USENIX Association.
- [14] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proc. of ACM MOBICOM*, Aug. 2000.
- [15] F. Milan, J. J. Jaramillo, and R. Srikant. Achieving cooperation in multihop wireless networks of selfish nodes. In *Proc. of workshop on Game theory for communications and networks (GameNets)*, page 3, New York, NY, USA, 2006. ACM.
- [16] M. Piatek, T. Isdal, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *NSDI*, 2007.
- [17] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On Selfish Routing in Internet-Like Environments. In *Proc. of ACM SIGCOMM*, Aug. 2003.
- [18] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2006-09-15). Downloaded from <http://crawdad.cs.dartmouth.edu/cambridge/haggle>, Sept. 2006.
- [19] A. Seth and S. Keshav. Practical security for disconnected nodes. In *Proc. of NPSEC*, 2005.

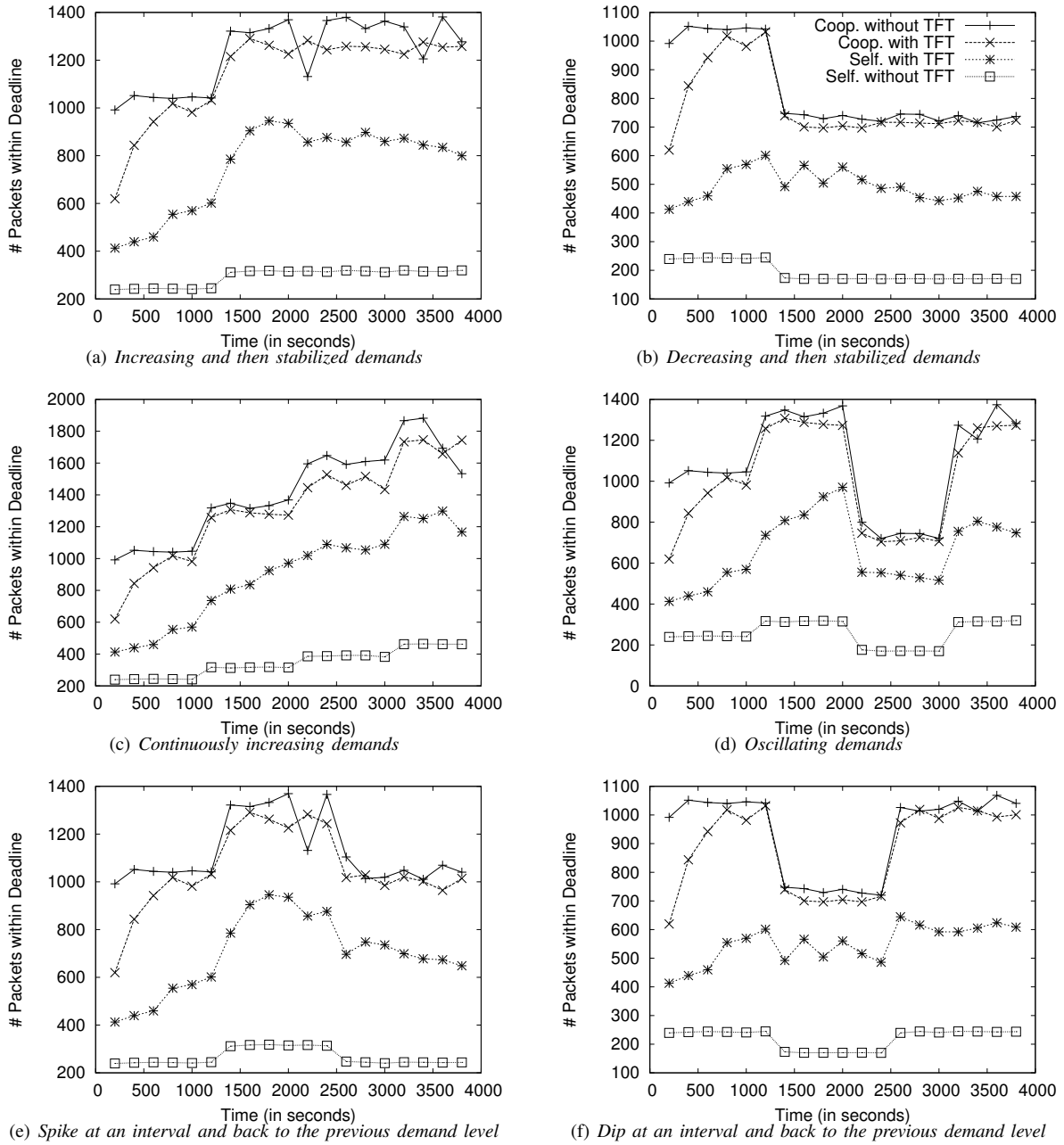


Fig. 8. Comparison of algorithms under different temporal variations.

- [20] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *Proc. of MOBICOM*, Sept. 2006.
- [21] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad hoc networks. In *Proc. of IEEE INFOCOM*, volume 2, pages 808–817 vol.2, 2003.
- [22] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. *Technical Report CS-200006, Duke University*, Apr. 2000.
- [23] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi. CRAW-DAD data set princeton/zebranet (v. 2007-02-14). Downloaded from <http://crawdad.cs.dartmouth.edu/princeton/zebranet>, Feb. 2007.
- [24] Wikipedia. Chebyshev's inequality. http://en.wikipedia.org/wiki/Chebyshev's_inequality.
- [25] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proc. of ACM SIGMETRICS*, 2003.
- [26] S. Zhong, J. Chen, and Y. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proc. of IEEE INFOCOM*, volume 3, pages 1987–1997, Mar.-Apr. 2003.