



Research Center
NRC-TR-2008-006

Controlling Resource Hogs in Delay-Tolerant Networks

John Solis¹, N. Asokan², Kari Kostinen², Philip Ginzboorg², Jörg Ott³

¹University of California, Irvine

²Nokia Research Center, Helsinki
<http://research.nokia.com>

³Helsinki University of technology, Networking Laboratory

July 11th, 2008

Abstract:

Delay tolerant networks (DTNs) are networks whose nodes have low connectivity and/or unreliable links. Because of these characteristics, DTN nodes use a store-carry-and-forward technique to deliver data through the network. Communication over DTN relies on intermediate nodes sharing their resources and can be abused by resource hogs, i.e. individuals who generate messages at a rate that is much higher than the average. In this paper we first show that if not controlled, resource hogs can substantially reduce the proportion of successfully delivered non-hog messages. To combat this problem we propose a basic technique that utilizes coarse-grained priority classes to control resource hogs. User or node authentication can be the basis for constructing priority classes: for example, messages from certain verifiable senders are assigned to a class with higher priority. The basic technique effectively deals with strangers who act as resource hogs but cannot counter verifiable senders who exhibit resource hog behavior. We extend the basic technique into three fine-grained solutions for dealing with such "insider" hogs. We show the effectiveness of each solution in restoring message delivery ratio to the scenario where no resource hogs are present.

Index Terms:

delay-tolerant networks
resource management
security

Controlling Resource Hogs in Delay-Tolerant Networks

John Solis, N. Asokan, Kari Kostianen, Philip Ginzboorg, Jörg Ott

Abstract—Delay tolerant networks (DTNs) are networks whose nodes have low connectivity and/or unreliable links. Because of these characteristics, DTN nodes use a *store-carry-and-forward* technique to deliver data through the network. Communication over DTN relies on intermediate nodes sharing their resources and can be abused by *resource hogs*, i.e. individuals who generate messages at a rate that is much higher than the average.

In this paper we first show that if not controlled, resource hogs can substantially reduce the proportion of successfully delivered non-hog messages. To combat this problem we propose a basic technique that utilizes coarse-grained *priority classes* to control resource hogs. User or node authentication can be the basis for constructing priority classes: for example, messages from certain verifiable senders are assigned to a class with higher priority.

The basic technique effectively deals with strangers who act as resource hogs but cannot counter verifiable senders who exhibit resource hog behavior. We extend the basic technique into three fine-grained solutions for dealing with such “insider” hogs. We show the effectiveness of each solution in restoring message delivery ratio to the scenario where no resource hogs are present.

I. INTRODUCTION

Challenged networking environments differ from traditional networking in that they are prone to communication delays and frequent disruption of communication links. In recent years, there has been a growing interest in developing delay- and disruption-tolerant networks (DTNs) [17], [13] for communication in challenged networking environments. Even though the challenges for deploying DTNs are daunting, there are many examples of DTNs [10], [22], [27] including several that are in practical use [8], [36], [30].

Communication opportunities in DTNs may be highly sporadic, leading to data transfer rates that are lower than in traditional networks. Effective operation of a DTN therefore relies on the co-operation of network nodes. Specifically, when a new message is injected into the network the sender of the message must rely on intermediaries to expend their resources to store, possibly carry, and forward the message for successful delivery. One aspect in which DTNs differ is where these intermediaries come from: Organized deployments, e.g., for space communications or bus-based networks [16], will use dedicated (fixed or mobile) infrastructure nodes for message forwarding. Sensor networks deployed in a coordinated fashion may or may not use infrastructure nodes, but are usually made up from homogeneous nodes sharing a common goal. In contrast, ad-hoc DTNs made up from mobile devices of human users are likely to have heterogeneous capabilities, users’ goals and cooperative behavior. More variations and hybrids exist.

In any DTN, intermediaries must make some of their resources available as a *public good* for use by others. However, any system that incorporates uncontrolled use of finite public goods is susceptible to “the tragedy of the commons” [19]:

the few who use more than their fair share of the public good will bring down the effectiveness of the system as a whole.

It is natural to ask whether this problem will manifest itself in DTNs and what can be done to mitigate its implications. While infrastructure nodes are deployed specifically for this purpose (and so should be equipped with the necessary resources) and homogeneous sensor nodes exhibit a coherent behavior in this respect, the situation is different for devices of mobile users: they offer resources voluntarily to the ad-hoc community, and their contributions may differ vastly. To study this issue, we introduce the notion of a *resource hog*. A resource hog is an individual who, on the average, attempts to send through the DTN more own data, and possibly forward less of the other nodes data, than is typical for well-behaved participants.

In this paper, we motivate the resource hog problem by showing that hogs can adversely affect the data transfer rates of honest users which reduces the utility of the network to them, and may discourage their future participation. An effective solution to this problem should result in restoring data transfer rates to the level where it would be when no hogs were present.

To combat resource hogs we propose a basic technique based on coarse-grained *priority classes*. User or node authentication can be the basis for constructing priority classes: for example, messages from certain trusted, and verifiable, senders can be assigned to a class with higher priority. We use simulations to show that this basic technique effectively deals with strangers who act as resource hogs but cannot counter trusted senders who exhibit resource hog behavior. We consider three different extensions for dealing with such “insider” hogs.

Contributions: first, we draw attention to the problem of resource hogs and argue that resource management techniques, including those based on user/node authentication, can be useful in controlling resource hogs. This is in contrast to recent work [9] which has argued that attacks against DTNs can be countered without resorting to authentication. Second, we propose and experimentally evaluate specific resource management techniques.

II. MOTIVATION

There are two classes of resource hogs: A *naïve resource hog* is someone who does make the resource contributions required from participants, but attempts to use more than the nominal or fair share of resources for its own messages. Therefore, a naïve resource hog will use the underlying routing protocol, and any resource management scheme, as specified, but may send significantly more messages than is typical for

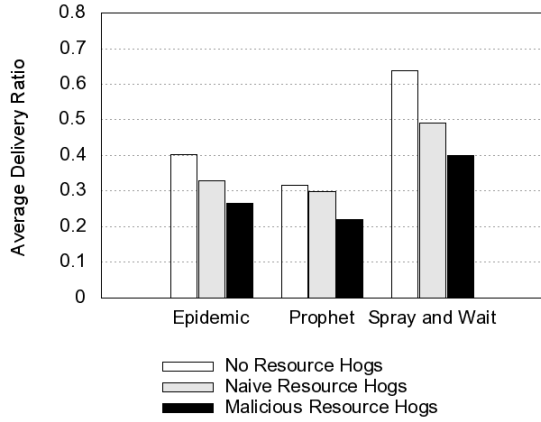


Fig. 1. Impact of 10% of hogs on message delivery ratio of honest nodes

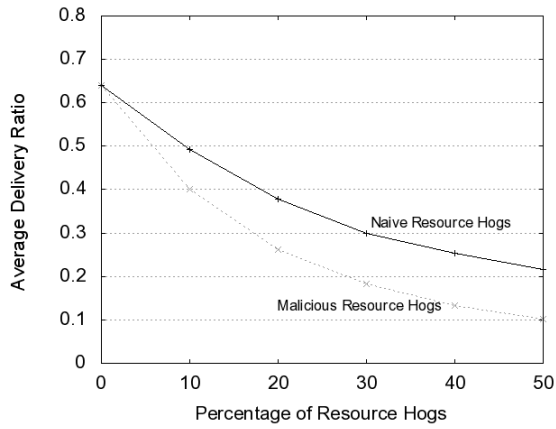


Fig. 2. Proportion of hogs vs message delivery ratio

normal participants. Note that a naïve resource hog is not necessarily an intentional attacker of the system.

A *malicious resource hog* is someone who does *not* make the requisite contributions but still attempts to use the common pool of resources, usually in excess of the nominal fair share. Thus, a malicious resource hog is analogous to a *free rider* in economic theory [18]. In our context, a malicious hog will likely use custom or manipulated software in order to give himself the best possible advantage. For example, it may drop incoming messages from others as soon as it accepts them (black holing)¹ or send out multiple copies of the same own messages to the same next hop.

To illustrate the impact of hogs, we now give an advance peek at the results of our simulations, described in more detail in Section V. We used a DTN simulator [23] on an example scenario consisting of a network with 250 nodes. We modeled a resource hog as a node that sends ten times more messages than an ordinary node. We examined the effect of hogs on three different routing algorithms (Spray and Wait [32], Epidemic [34], and Prophet [24]).

¹This would make it harder to distinguish a malicious resource hog from an honest user.

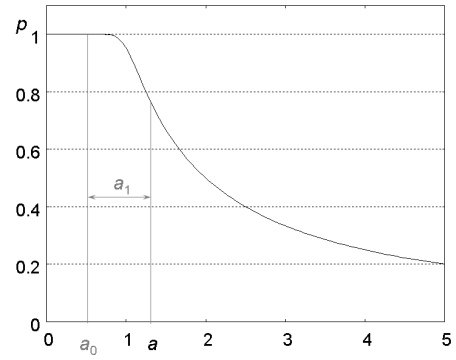


Fig. 3. p as a function of the load a for $K = 20$

We use *message delivery ratio* r as the metric for data transfer rate. For a given set of nodes, r is defined as

$$r = \frac{\text{number of its messages successfully delivered}}{\text{number of messages sent}}$$

Figure 1 shows that 10% of hogs in the network population causes r of honest nodes to decrease considerably.

While this is only an example scenario, it serves to illustrate the potential harm that hogs can cause, and thus to motivate the reason to develop mechanisms to control them. As can be seen from this figure, different routing protocols suffered to different extents, with the more effective routing protocols showing a greater vulnerability to hogs.

Figure 2 shows how the average delivery ratio decreases as the proportion of hogs in the network increases. The decrease is more significant with malicious hogs, as expected. For the remainder of the paper, we set (a) Spray and Wait as the routing protocol since it exhibited the greatest observed impact and (b) the proportion of hogs at 10%.²

The resource hogs problem can be also illustrated qualitatively by modeling a DTN network as a single queue with a finite buffer and varying its load a , i.e., the ratio between arrival and departure frequencies. This model, denoted M/M/1/K in queuing theory, has a single server, Poisson input, exponential service time, and capacity K (see, e.g., [26]). The probability p of an incoming message being accepted into the queue is $(1-a^K)/(1-a^{K+1})$ when $a \neq 1$ and $K/(K+1)$ when $a = 1$. As K increases, p approaches unity for $0 \leq a \leq 1$, and $1/a$ for $a > 1$. The dependency of p on the load for $K = 20$ is illustrated on Figure 3.

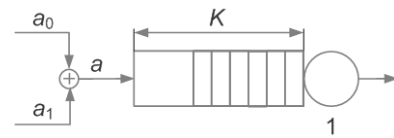


Fig. 4. Finite buffer queue with two sources

²We choose the 10% value in our simulations to show a situation in which a minority of network nodes has a noticeable effect on the delivery ratio of the majority.

Suppose that as shown in Figures 3 and 4, the input contains messages of two Poisson sources that generate loads a_0 and a_1 , i.e. $a = a_0 + a_1$. Let a_0 model traffic originating from normal participants, and a_1 model traffic originating from hogs. Under normal conditions, a_0 is likely to be a value close to 1. As the percentage of resource hogs increases, a_1 increases well beyond $1 - a_0$ and p falls in proportion to $1/a$. This effect of hogs predicted by Figure 3 is in fact what we observe experimentally in Figure 2.

III. RESOURCE HOG PROBLEM

As seen in the previous section, even a small percentage of resource hogs can negatively impact the message delivery ratio of rest of the network population. The problem we wish to solve becomes: *How to restore the delivery ratio of honest participants even in the presence of hogs?*

One standard solution to the problem of hogs is to treat the resources as “club goods” by structuring the system as an “association of clubs” [2]. Members belonging to the same club trust each other and may have some way of verifying this trust. We can apply this solution to DTNs by grouping users into *domains*. A domain is a subset of users whose messages can be identified as originating from that particular subset.

An intermediary node receiving a message can reliably *authenticate* messages sent by domain members. It is not required that the sender and the verifier be members of the same domain. The sender authentication can be based on, e.g. (1) including a certificate signed by a well known authority, or (2) using a remote attestation protocol to prove presence of a particular piece of software [3].

Restricting usage of resources to members of trusted domains alone is not desirable in DTNs, as Burgess et al. show in [9]. By allowing other unknown users to use his resources, a user will maximize the chances that his own messages will be carried by other users who cannot authenticate him. This is necessary as members of one domain may, e.g., travel to areas where others dominate, yet still want to be able to communicate. Thus, we can refine the original problem by requiring that an acceptable solution must not be exclusive.

IV. RESOURCE MANAGEMENT

A. Assumptions and Requirements

We make the following assumptions about the context in which we want to solve the resource hog problem:

Assumption 1: Local Decision Making. No information relevant to resource management can be assumed to be available to all nodes. Therefore, nodes must make all decisions locally, but can store information that can be learned directly (e.g. number of bytes forwarded on behalf of a certain sender). This allows us to apply our solutions to any DTN network as it frees us from assumptions about infrastructure support or regular information exchange between nodes.

Assumption 2: Constant Size Resources. Each node sets aside a constant amount of resources to be managed. This would also apply to infrastructure nodes. The amount of

resources may differ across nodes; nevertheless, we restrict our considerations to equal contributions from every node.

Concerning the resources most relevant for mobile nodes, we focus on the storage for buffering messages as the resource to be managed.³ The message buffer is used for temporary storage of DTN messages and thus directly impacts the capability to carry messages prior to forwarding.

We identify the following requirements for a satisfactory resource management scheme:

Requirement 1: Restore Delivery Ratio. It should restore the delivery ratio seen by honest participants as close as possible to the scenario with no hogs. Suppose $r_{m,s}$ is the delivery ratio when a resource management scheme m is applied in a particular scenario s and $r_{0,0}$ is the delivery ratio when no resource management scheme is used and there are no hogs. The *normalized delivery ratio* for applying m in s is defined as $\bar{r}_{m,s} = r_{m,s}/r_{0,0}$. Requirement 1 states $\bar{r}_{m,s}$ should ideally be 1.

Requirement 2: Do No Harm. It should have minimal impact when there are no hogs present in the network. In other words, $\bar{r}_{m,0}$ should ideally be 1. This can be seen as a special case of Requirement 1.

Requirement 3: Routing protocol independence. It should be independent of the underlying routing algorithm.

Requirement 4: No Exclusivity. When there is no resource congestion on the node implementing the scheme, it should allow a message from any sender to use the resources on that node.

B. Pre-emptive Buffer Management

Intuitively, we will structure our buffer management algorithm around the concept of domains and give domain members priority to buffer space. However, to meet the “no exclusivity” requirement we will use a pre-emptive scheme to guarantee a certain level of service to domain members, while accepting non-domain traffic.

In the following, we define a principal as a source of a message and it can be a single node or a set of nodes (e.g., a domain). We will classify all unauthenticated messages as coming from same principal.

The notion of allocating resources for a particular domain can be formalized as the concept of a *threshold*, representing the nominal share of of buffer space utilization of that domain in a desired or ideal condition.

Figure 5 presents the pre-emptive buffer management algorithm which is called to decide on accepting an incoming message m . If the buffer has enough free space for m then m is accepted. Otherwise the buffer is deemed congested and we must make room by dropping at least one message. Since we desire to allocate space according to the specified set of thresholds T , we will drop a message from a *principal* that exceeds its threshold the most. If the message chosen to be dropped is m , then it is simply rejected.

³Another major aspect is energy consumption which we consider outside the scope of this paper. We simply make the (not unreasonable) assumption that device batteries will last for a full day and the devices can be recharged during the night.

```

HANDLE-INCOMING-MESSAGE( $m$ )
1  if SIZE( $\{m\}$ ) > 1
2    then DROP-FROM( $\{m\}$ )
3    else  $A \leftarrow A \cup \{m\}$ 
4   $T \leftarrow$  UPDATE-THRESHOLDS( $A$ )
5  while SIZE( $A$ ) > 1
6    do
7       $i \leftarrow$  GET-CANDIDATE-FROM( $A, T$ )
8       $S \leftarrow$  MESSAGES-OF( $i$ )
9      DROP-FROM( $S$ )

```

Variables:

S is a set of messages.

$u_i(S)$ denotes the fraction of the buffer space utilized by messages from principal i that are in S .

T_i is the threshold for principal i . The sum of all T_i 's is 1.

n is the number of principals known to the node.

T is the set of thresholds $\{T_1, \dots, T_n\}$.

m denotes an incoming message.

A is the set of messages accepted into the node.

Macros:

SIZE(S) is the fraction of buffer space occupied by the messages in S .

DROP-FROM(S) drops a message belonging to set S from the buffer. The decision on which message to drop is done by the DTN routing protocol.

UPDATE-THRESHOLDS(A) updates the set of thresholds for principals in A .

GET-CANDIDATE-FROM(S, T) given a set of messages S , returns an identifier i of a principal whose buffer usage exceeds its threshold the most: i is such that $u_i(S) - T_i \geq u_j(S) - T_j$, where $i \neq j$.

MESSAGES-OF(i) returns the list of messages from principal i .

Fig. 5. Pre-emptive buffer management algorithm.

V. EXPERIMENTAL EVALUATIONS

We simulated various scenarios to investigate the extent to which the pre-emptive buffer management algorithm above can control resource hogs in the network. For each tested scenario we ran ten simulations using the ONE simulator [23] and averaged the results across all ten simulation runs. In the rest of the paper, when simulation results are presented, they are the average of ten simulation runs. Where appropriate, we also give the value of the standard deviation σ among the ten runs in square brackets [] (e.g., $r=0.64$ [0.02] means that the average value of r among ten runs was 0.64 with $\sigma = 0.02$).

A. Synthetic Mobility Model

First we used the mobility model generator included in the ONE simulator to generate mobility traces. Our intention was

to simulate users with portable mobile devices walking around in an urban area during the course of a day sending messages like emails and pictures. We selected the parameters with this scenario in mind.

Mobility. The map of a downtown area (about 14 km²) was used for the mobility model. We modeled the movement of 250 nodes which move for 12 hours along shortest paths of the map with speeds selected uniformly at random between 0.5 and 1.5 m/s. A node that reached its destination will pause for a period between zero and 120s, selected uniformly at random, before selecting a new destination. Each of the ten simulation runs for a given simulated scenario used a different seed for the random number generator in the mobility model generator.

Connectivity and transmission. At most two nodes can communicate with each other at a time. The communication is bi-directional, at a constant transmission rate of 250 kB/s, and will continue as long as one of the nodes has messages that the other does not, provided that they stay within a 10 m range of each other.

A node will try to communicate with a randomly chosen node among those in range. If there is no data to exchange with the chosen node, or if the chosen node cannot respond, e.g., because it is exchanging data with another node, then a different node will be randomly chosen among those in range.

Message transmission is atomic. If two communicating nodes drift out of range during message transmission, then the partially transmitted message is retained by the sending node and discarded by the receiving node.

We used Spray and Wait protocol [32] that initially starts with six ‘‘copies’’ of each message.⁴ In our implementation, messages whose destination is the current peer node are always transmitted *first*; other messages are transmitted in the FIFO order.

Traffic model. An ‘‘honest’’ network node generates on the average one message per hour. A resource hog node generates on the average ten messages per hour. Node behavior, i.e. whether it acts as a hog or not, does not change with time. The destination of each message is chosen at random. Messages have a time-to-live (TTL) attribute set to 3 hours and their sizes are uniformly distributed between 100 kB and 2 MB.

Buffer management. Each node has a buffer of 20 MB reserved for DTN traffic and that buffer is normally (i.e. when no explicit resource management scheme, like the pre-emptive buffer management, is active) managed as follows. The node accepts an incoming message that is not larger than the size of the buffer if, after removing expired messages, there is enough space for it in the buffer. Otherwise it invokes DROP-FROM() (with the entire set of messages in the buffer as input) to discard a message until enough space is freed up for the incoming message. The implementation of DROP-FROM() depends on the routing algorithm. In the case of the routing protocols we have used so far it discards the oldest message in the current buffer.

⁴A node keeps track of how many ‘‘copies’’ it has. The actual message is not replicated within a node.

Scenario	Delivery Ratio		Latency (sec.)	Hop Count
	Observed r	Norm. \bar{r}		
No hogs	.64 [.02]	1.00	4293 [59]	2.43 [.02]
Naïve	.49 [.01]	.77	3507 [51]	2.42 [.03]
Malicious	.40 [.01]	.63	3274 [90]	2.36 [.04]

TABLE I

EFFECT OF 10% RESOURCE HOGS. ALL FIGURES ARE FROM THE POINT OF VIEW OF HONEST NODES.

Nodes do not retransmit own messages that have been dropped by the DTN layer. Those messages are counted as lost. Honest nodes and naïve hogs do not treat a message that they generate differently from messages coming from outside. Malicious hogs accept and immediately drop all messages from other nodes. Duplicates of already received messages are discarded by the receiver and not counted.

B. Effect of Resource Hogs

Table I shows the results of the first three simulated scenarios (no hogs, naïve hogs and malicious hogs). Average latencies and hop counts are shown in addition to the average message delivery ratio. All the figures are from the point of view of honest nodes.

As we already saw in Section II, the average delivery ratio r drops significantly in the presence of hogs. We see that in the presence of hogs latency observed by honest nodes actually *decreases*. This is because due to the increased number of messages in the system, more intermediate nodes must make room for new messages by dropping the ones they carry; messages that do reach their destination travel less time. Similarly the presence of malicious hogs leads to a noticeable decrease in hop count because they silently drop messages and break potential communication paths; messages that reach their destination do so over shorter paths.

The full impact of resource hogs becomes evident only when one considers the point-of-view of honest nodes, as in Table I, rather than the system-wide delivery ratio. To illustrate, the delivery ratio r for the *system as a whole* was 0.47 [0.01] (2680 [46] messages out of 5700) in the scenario with naïve resource hogs and 0.44 [0.01] (2508 [41] out of 5700) in the scenario with malicious hogs. We ran a fourth simulation set in which 250 honest nodes generated 5750 messages over 12 hours. The resulting r value was 0.48 [0.01] (2753 [55] out of 5750).

Another point of view on the effect of resource hogs is to look at how the buffer occupancy of the whole network evolves with time. This view is shown in Figure 6: We see that after initial transition period of about three hours, the system enters steady state in which 10% naïve hogs occupy roughly half of the total buffer space in the network.

C. Controlling Resource Hogs

Now we consider different approaches for controlling hogs using pre-emptive buffer management. First, we focus on a scenario where a DTN node is a member of a particular domain and can only authenticate other domain members.

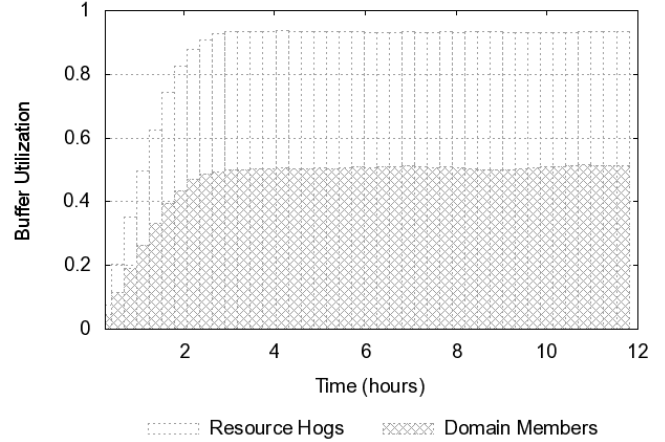


Fig. 6. Buffer occupancy of the network (with 10% naïve resource hogs) as a function of time.

There may be several domains in the same network. However, a given node sees traffic from two principals: “insiders” consisting of senders that belong to its own domain and “outsiders” consisting of all other senders. We will label the insiders domain with “in” and the outsiders domain with “out”. We argue that in most practical scenarios this is the only information that a node will be concerned with.

Our preliminary experiments showed that the delivery ratio from domain members peaks at threshold value $T_{in} = 1$, i.e. the threshold values $T_{out} = 0$ and $T_{in} = 1$ are optimal for domain members in this scenario. In this case the subroutine GET-CANDIDATE-FROM(A, T) in the pre-emptive buffer management algorithm (Figure 5) is very simple. It returns “out” if $u_{out}(A) > 0$, and “in” otherwise. Messages from outsiders are carried “best effort”: they are always pre-empted by messages from insiders.

In our simulations, we consider a single domain in the network. We focused primarily at two different cases: one in which the domain is *small*: where 50 domain member nodes constitute 20% of the total population, and the other in which the domain is *big*: 200 domain member nodes constitute 80% of the total. However, we also looked at other domain sizes, viz. 40%, 60% and 100% of the population. Nodes outside the domain do not use the pre-emptive buffer management algorithm. In each of the above cases, we consider both insider and outsider hogs as well as both malicious and naïve hogs. The resulting delivery ratios of the eight cases are presented in Figure 7. Each data point is the average of r over 10 simulation runs, and the vertical bars above and below a data point represent the associated standard deviation.

Recall that our objective is to restore \bar{r} for honest users as close to 1 as possible (that is, restoring r as close to 0.64 as possible). These results show that when the resource hogs are outsiders, the coarse-grained approach can restore r for honest users to between 0.59 ($\bar{r} = 0.92$) for malicious outsiders in small domain and 0.63 ($\bar{r} = 0.98$) for naïve outsiders in big

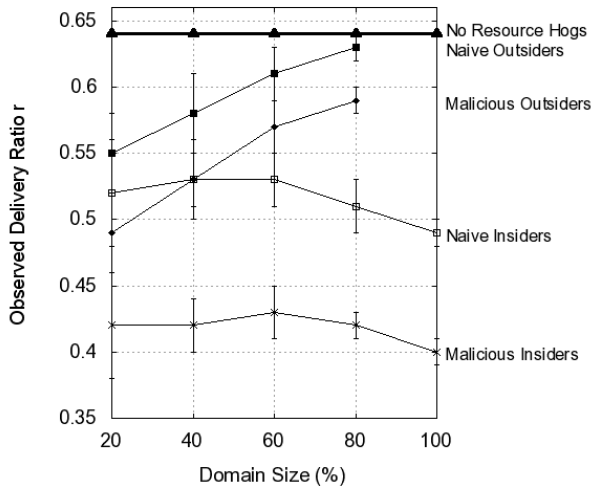


Fig. 7. Effect of coarse-grained buffer management with 10% resource hogs

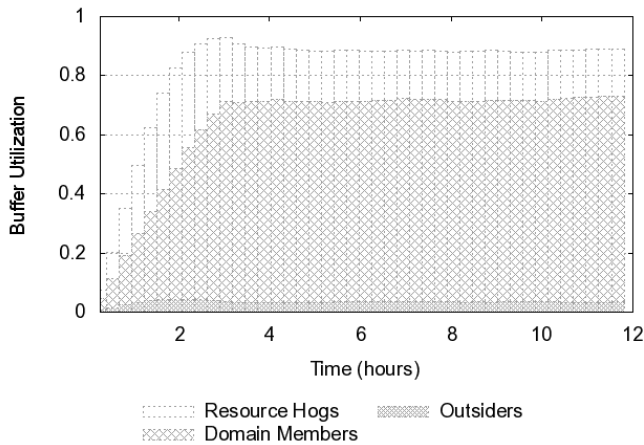


Fig. 8. Evolution of buffer occupancy with coarse-grained domain management during 12 hours.

domain.

Another way to illustrate the effectiveness of coarse-grained buffer management against outsider hogs is to look at how the overall buffer occupancy of the network evolves. Figure 8 shows the buffer occupancy as a function of time when there are 10% naïve hogs and the coarse-grained buffer management is used. One can see that after the system has reached its steady state (in three hours), the hog messages occupy roughly 20% of the total available buffer space. In similar scenario shown in Figure 6, without coarse-grained buffer management the hog messages occupy roughly half of the buffer space.

But the coarse-grained scheme cannot sufficiently protect honest users when hogs are inside the domain. In this situation there is no way to distinguish between an honest node and an “insider” resource hog: each node must accept traffic originating from a fellow domain member. Consequently, undetected insider hogs can send out large amounts of information that

will receive priority in all domain members.

D. Fine-grained Buffer Management

In order to tackle the problem of insider hogs we need a way of distinguishing between individual domain members. i.e., we need a fine-grained approach to buffer management: If the buffer is congested after all outsiders messages have been removed by the above basic algorithm, we apply another method on the remaining insider messages.

We considered three different approaches for extending the basic coarse-grained pre-emptive buffer management algorithm, as described below.

Equal Sub-domains Approach. Our first approach is to further subdivide the insider domain into sub-domains consisting of unique senders. We do this by implementing UPDATE-THRESHOLDS(A) in Figure 5 as follows: it counts the number $n(t)$ of distinct senders whose messages currently occupy the buffer, assigns a separate dynamic sub-domain for each of them, and sets $T_i(t) = 1/n(t)$, where t is the current time and $T_i(t)$ is the threshold for principal i at time t .

Equal sub-domains combines simplicity of implementation⁵ with high delivery ratio r . A disadvantage of this method is a pronounced dive in delivery ratio of large messages (see Figure 10).

Usage-biased Sub-domains Approach. One way to overcome tendency of the equal sub-domains approach to drop large messages, is to keep additional temporal information about past usage of buffer space by individual senders. In accordance with the “local decision making” assumption, we only use historical usage data measured locally on a node. Let $G_i(t)$ be the total amount of data originating from principal i that was received and stored up to time t . Intuitively, we want a node to assign the threshold for a given sender based on how much cumulative buffer space has been used by that sender in the past. Senders with higher usage should be assigned lower thresholds. This can be done by implementing UPDATE-THRESHOLDS(A) in Figure 5 as follows: If $G(t)$ is the sum $1/G_1(t) + 1/G_2(t) + \dots + 1/G_n(t)$, set $T_i(t) = 1/(G_i(t)G(t))$.

Penalty Box Approach. An alternative approach is to identify potential insider hogs and temporarily banish them from the insider domain. This is analogous to a penalty box in ice hockey where players are temporarily removed from play after committing an offense. The Penalty Box approach maintains a list with identifiers of potential hogs. Nodes in that list are treated as outsiders; their messages will be pre-empted by the coarse-grained buffer management in case of congestion.

We have implemented the Penalty Box idea as follows. The buffer is either in “congested”, or in “uncongested” state. The penalty list is populated when the buffer is in the “congested” state: Whenever a congestion event occurs in that state we add to the list a node whose messages are utilizing the most space in the buffer. The penalty list is emptied on each transition from “congested” to “uncongested” state.

⁵It can be shown that setting $T_i(t) = 1/n(t)$ is equivalent to choosing the principal with the highest buffer occupancy at time t .

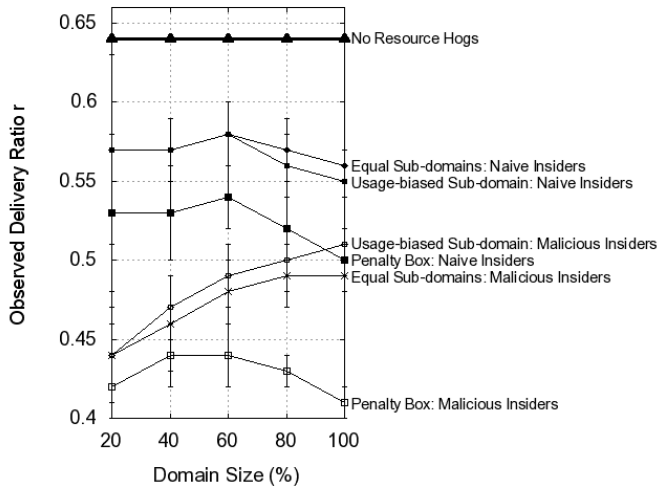


Fig. 9. Fine-grained Buffer Management: Average Effect of 10% Insider Resource Hogs (system load 3000 messages)

The buffer moves between the states based on the value of a counter c that records how many out of last W messages have encountered congestion. Transition from “uncongested” to “congested” state occurs if $c \geq C_1$. Transition from “congested” to “uncongested” state occurs if $c \leq C_2$, where $C_2 < C_1$. We have used $W = 8$, $C_1 = 1$ and $C_2 = 0$ in the simulations of the Penalty Box method.

Figure 9 presents the results of simulating fine-grained buffer management in our example scenario. As can be seen from the figure, both Sub-domain management approaches outperform Penalty Box. In case of naïve insider hogs both Sub-domain management schemes can recover the delivery ratio up to 0.58 ($\bar{r} = 0.91$). With malicious insider hogs the achieved delivery ratios are considerably lower, 0.51 at best ($\bar{r} = 0.80$).

We can conclude that both Sub-domain management approaches deal with naïve insider hogs, but malicious insider hogs cannot be controlled effectively in our target scenario. (We will say more about this in part A of section VI.) Penalty Box approach is not efficient—at least not using the parameters that we have experimented with.

Delivery ratios of different message sizes. Figure 10 illustrates observed message delivery ratio r as a function of message size using Equal Sub-domains, Usage-biased Sub-domains and Penalty Box approaches. The messages sizes have been divided into bins of 100 KB and the y -axis in the figure plots the observed r for messages in each bin. The domain contains 200 nodes (80% of the total), out of which 25 are malicious hogs.

In each resource management scheme small messages have higher delivery ratio, and in Equal Sub-domains large message suffer the most. This is due to the fact that as time goes by the number $n(t)$ of distinct senders that occupy each buffer increases, causing the individual sender’s thresholds to decrease. Naturally, large messages tend to get dropped first.

A similar dependency, with less pronounced dive in r , exists

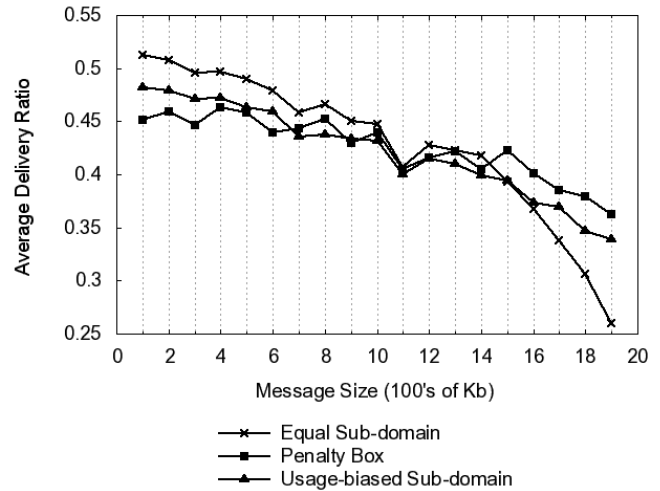


Fig. 10. Delivery ratio r as a function of message size.

when the resource hogs are of naïve type.

E. Real-World Traces

We also studied the resource hog problem using contact patterns extracted from real-world traces. We used Dartmouth campus traces [15] which contain the access patterns of almost 1,400 students throughout various buildings on the campus over five years. We chose 250 students and used their contacts during 10 randomly picked school weekdays within a single building (the library). We ran the same simulations as above.

The results showed a low delivery ratio ($r_{0,0}$ of 0.31 [0.01]). There are fewer, roughly half, contacts compared to the synthetic model because the students move less and the WLAN contacts stem from laptops which are turned off during movement. This limits message spreading and reduces the average buffer occupancy per node (by a factor of three). We observed that both naïve and malicious resource hogs have very little impact (\bar{r} of 0.98 [0.03] and 0.92 [0.03], respectively) in this scenario, even if no resource management is applied. For large domain of 200 nodes, introducing coarse-grained management virtually eliminates the impact of outsider hogs but has little effect with insider hogs. However, introducing coarse-grained management in small domain of 50 nodes makes things worse with $\bar{r} = 0.77$ [0.07] in naïve and $\bar{r} = 0.59$ [0.06] in malicious insider hogs scenarios. When applying fine-grained management, the impact primarily depends on the domain size rather than the hog type: for small domains \bar{r} is 0.7–0.8, for large ones 0.85–1, with the sub-domain management and usage-biased algorithms performing best.

Intuitively, a lower degree of connectivity makes particularly small domains susceptible to insider resource hogs as there are fewer (if any) alternative paths. Buffers of less connected nodes remain empty, while the buffer of a few (likely outsider) nodes get easily filled up with hog messages.

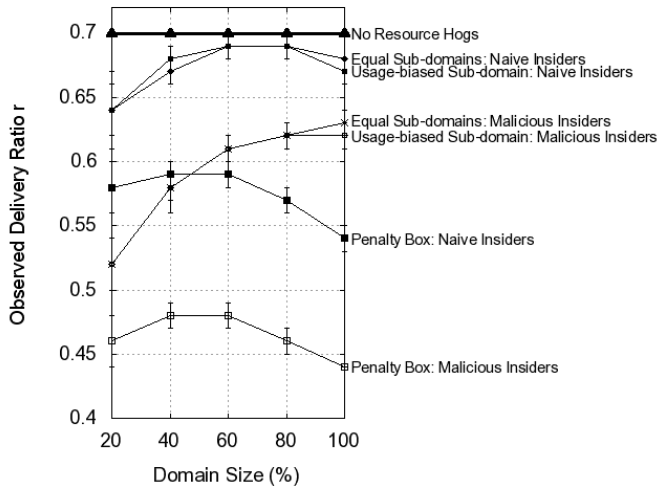


Fig. 11. Effect of fine-grained buffer management with 10% insider hogs in long simulation (48h).

VI. ANALYSIS

Now, we analyze our buffer management schemes based on the requirements identified in Section IV-A and revisit our choice of simulation parameters.

A. Requirements Analysis

Requirement 1: Restore Delivery Ratios. We already saw from Figure 7 that coarse-grained buffer management by large domains can restore the normalized delivery ratio \bar{r} close to 1 (0.98 at best) in the case of outsider hogs.

We introduced fine-grained buffer management schemes to address the case of insider hogs. As can be seen from Figure 9 both Equal Sub-domain and Usage-biased Sub-domain can restore delivery ratio to 0.91 at best in case of naïve insider hogs. Malicious insiders hogs cannot be controlled as effectively with these approaches.

The natural next question is: why the fine-grained buffer management approaches cannot restore \bar{r} closer to one? To understand this question we ran a set of longer simulations (48 simulated hours instead of 12). The results from these simulations are presented in Figure 11. As can be seen from the figure, our fine-grained buffer management schemes restore \bar{r} considerably closer to one (0.99 at best) when the simulation time is three times longer than in our original experiments.

This phenomenon can be explained by comparing Figure 12 to Figure 8. In both simulations, the resource hog messages consume approximately half of the total available buffer space before the system reaches its steady state. Once the system is in steady state, i.e. the buffers of the network nodes are primarily full, the resource management scheme kicks in and controls the share of the buffers occupied by hog messages. The time to steady state is approximately three hours in our system.

In the simulation that lasts 12 hours the effect of the three hour start phase is significant and thus the achieved delivery ratios remain relatively low. In the simulation that lasts 48

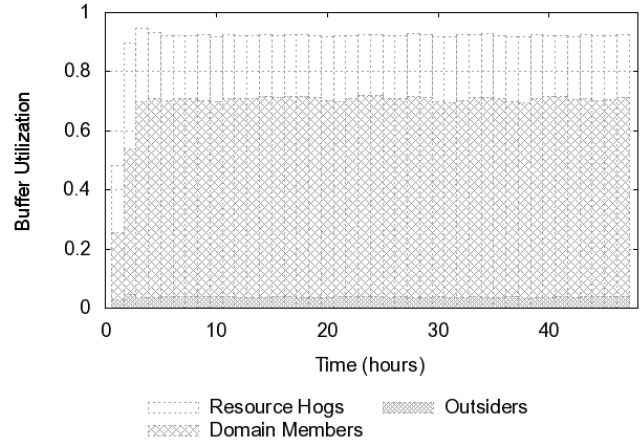


Fig. 12. Evolution of buffer occupancy with coarse-grained domain management during 48 hours.

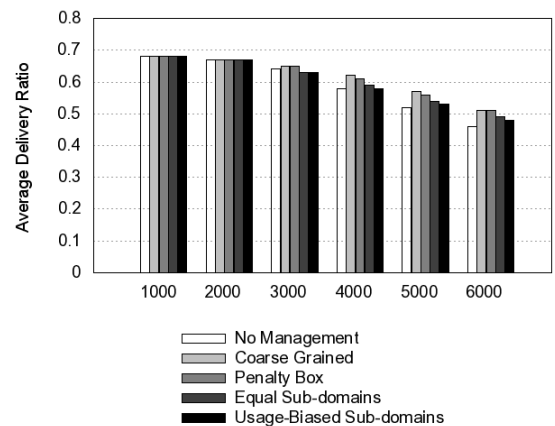


Fig. 13. Delivery ratios of domain members when there are no resource hogs. Domain size is 200 nodes.

hours, the effect of three hour start phase is smaller and so the achieved delivery ratios increase. This observation shows that our technique is applicable to a wide variety of potential DTN scenarios.

Requirement 2: Do No Harm. We ran simulations with each of the buffer management techniques using the “no hogs” scenario. Message sizes are uniformly distributed between 100 KB and 2 MB, and the domain size is 200 nodes (80% of the total population).

Figure 13 illustrates the effect of each scheme under varying network loads. It plots r when no buffer management is used as well as r of the proposed buffer management techniques.

At loads of 1000 and 2000 total messages sent during the 12 simulated hours the buffers of intermediate nodes never become congested. There is too little traffic in the network and buffer management never becomes an issue. All scenarios, as expected, perform identically.

At higher loads we notice the same trends in performances between resource management schemes as described in the

previous section.

Notice that beginning with the load of 3000 total messages sent the coarse-grained domain management appears to increase the delivery ratios of the particular scenario. This is explained by the fact that the delivery ratios provided are from the perspective of the domain members only. The increase in delivery ratio comes at the cost of a lower delivery ratio for non-domain members. As we increase the number of messages in the system, this increases the amount of traffic that could potentially pre-empt the traffic from non-domain members.

After analyzing the proposed management schemes under various loads we conclude that they do not harm overall delivery ratios.

Requirement 3: Routing Protocol Independence. Our approach was generic enough to be used with several different routing algorithms. Our current implementation works by applying the buffer management algorithm to select a set of messages from the buffer, and then invoking the routing algorithm to determine which of these to discard.

However, using our approach with metric-based routing schemes which already define some form of buffer management requires further study. Uncoordinated *accept* or *discard* decisions could potentially impact the overall efficiency of the metric-based routing algorithm.

Requirement 4: No Exclusivity. This requirement is trivially satisfied by virtue of using a pre-emptive scheme which allows messages from unknown nodes to be still carried and forwarded.

B. Buffer vs. transmission management

As described in Section V.A, our implementation of Spray and Wait protocol transmits messages from the message buffer in FIFO order (the only exception are messages that have the current peer node as their destination - these messages are transmitted first.) Our mobility model leads to random mixing of messages from different senders in the DTN buffers of intermediate nodes.

If messages are mixed randomly in the buffers of intermediate nodes, then it is easy to see that FIFO transmission results in allocation of transmission opportunities between buffer users according to their proportional share of buffer space. In particular, if there are messages from $n(t)$ senders in the buffer and the buffer space is equally divided between those senders, then FIFO transmission will result in an equal share of transmission opportunities between those senders.

The Equal Sub-domains buffer management method strives to achieve equal sharing of buffer space between domain members during congestion. It follows that in our case Equal Sub-domains also leads to equal sharing of transmission opportunities between domain members during congestion.

More generally, independent of mobility model, transmission of messages in random order combined with Equal Sub-domains buffer management would lead to equal sharing of transmission opportunities between domain members during congestion.

Another way to solve the resource hogs problem is by managing transmission opportunities of each principal directly. This is done by Fair Queuing (FQ) [28] and many subsequent schemes (e.g. [25]): Each traffic source is allocated an own queue in an intermediate node and the transmission process of that node takes a packet from non-empty queues, e.g., in a round robin fashion.

It may be that delivery ratio r of honest domain members could be increased further by intermediate nodes applying FQ on outgoing messages, in addition to applying sub-domain buffer management on incoming messages. Verification of this conjecture could be a topic for future work.

VII. RELATED WORK

Resource management has a long history for infrastructure as well as for mobile ad-hoc networks, focusing on bandwidth, processor, and buffer management as well as on energy management for mobile nodes. Goals include achieving some notion of fairness with respect to resource utilization between different *flows* (i.e., series of packets) and/or providing service guarantees to individual or classes of flows.

In fixed and infrastructure-based wireless networks, variants of *Fair Queuing (FQ)* [28] and *Weighted Fair Queuing (WFQ)* are established mechanisms for achieving (flow-level) fairness - e.g., max-min or proportional - inside a single router/node per outgoing link. They isolate incoming traffic of different flows into separate queues and serve these, e.g., in a (weighed) round-robin fashion.

(W)FQ could be applied in DTN to manage transmission opportunities during contact between DTN nodes. In this paper, however, we have concentrated on managing buffer space: We felt that ingress management may be easier to integrate with DTN routing schemes than transmission management. Investigating the effect of combined buffer and transmission management in DTN could be a subject of future work.

In mobile ad-hoc networks, where shared communication channels without a central coordinating function exist, this concept has been extended to *Distributed Fair Queuing (DFQ)*: the focus shifts towards arbitrating the access to a common channel for multiple interfering nodes [25], i.e., servicing distributed queues.

Buffer management involves decisions about admitting or dropping individual incoming packets and placing them into one or more queues. Once admitted, servicing the queue and scheduling determines when and which packets are forwarded. With IP networks, end-to-end transport protocols (and applications) are expected to adapt to the network conditions so that the application demands can be met: e.g., use retransmissions to achieve reliability or rate adaptation for congestion control. With DTNs, flows of related packets collapse into messages transmitted one at a time. Drop decisions for a DTN message may thus have a much stronger impact on application performance because, unlike individual lost packets, message loss cannot be instantly repaired by retransmissions and because more information may get lost at once.

It has recently been argued [5] that fairness based upon flow rates is unsuitable and that rather than the *per-flow benefit the cost per accountable entity*, e.g., a user, should be considered for resource control. This notion is more natural to DTNs and is reflected in our approach: we consider all messages originated by an identifiable entity (or a group of such identities) together.

DTN routing algorithms—particularly in mobile ad-hoc settings where upcoming forwarding opportunities are unknown—may not be able to determine a path towards a destination and so replicate messages to increase their delivery probability [21], [37], [29]. Also, a DTN node cannot know in advance for how long it shall store a message; a message may be stored until it expires.

Typically, buffer management is closely tied to routing. To limit the total number of copies of a single message in the system (and thus the associated resource utilization), the number of copies may be defined at the sender (as in Spray and Wait [33]) or a counting protocol may be used to estimate and limit the fraction of nodes that carry a copy of a message [35]. Several DTN routing protocols have dealt with finite buffer space and the need for replacing messages including PRoPHET [24], MaxProp [8], and RAPID [4]. RAPID goes furthest and estimates whether accepting and forwarding a message would increase the yield of a given utility function, thus explicitly comparing the value of messages. Seligman et al. [31] have studied relocating and later retrieving messages to deal with congestion for messages for which message delivery shall be “guaranteed” by the network using hop-by-hop reliability (*custody transfer* [13]).

However, these DTN routing schemes, assume that all messages are equally legitimate and optimize the overall delivery performance of the system. They do not account for the amount of resources taken up by an individual user and hence do not address the resource hogs problem.

Closest to ours is the work by Burgess et al. [9] who discuss the ability of DTN networks with a replicative routing protocol to gracefully survive attacks. The same properties that make routing in DTNs difficult also make attacking difficult. They conclude that malicious adversaries who can flood, corrupt, and drop messages only minimally affect the DTN network.

But unlike Burgess et al., we do not look at attackers who are trying to bring down the network or a particular node. Instead, we identify a different kind of misbehavior: A fraction of nodes increasing their own number of messages sent through the network brings down the delivery ratio seen by other nodes. We have shown that a problem exists with such resource hogs and that resource management is one way to mitigate their detrimental effect.

Outside of DTN specific research, packet forwarding has been studied extensively in ad-hoc networks and many solutions to handle uncooperative nodes have been proposed. The proposals fall into one of three general categories: reputation systems, game theory approaches, and credit based solutions.

Distributed reputation systems such as [6] [7] use state machines to keep track of the reputation of other nodes. Each

node updates their state based on direct observation of peer nodes and through received reports on peer behavior. In addition to having been formally proven to be unworkable [14], reputation systems cannot operate in an DTN environment. The infrequent connections of peer nodes in DTNs do not give nodes the opportunity to build reputations. We therefore restrict our approach to using only information learned first hand.

Game theory approaches, present models for rational nodes and describe strategies that lead to an equilibrium state that allows for forwarding in the network. In general, these solutions are difficult to apply to DTNs because they assume good connectivity and reliable delivery of messages. As with reputation systems, the low connectivity and poor delivery ratios of DTNs does not allow such solutions to converge to a solution. Some solutions, such as, Altman et al. [1] require nodes to know the topology of the network in their immediate area. Knowing the local topology allows each node to compute the equilibrium of the system, but the high mobility of DTN nodes makes it difficult for a node to know the topology of any part of the network for any given length of time.

Credit-based solutions attempt to “charge” for use of network resources. Buttyan and Hubaux [11] [12] present a simple solution that requires the use of tamper-proof hardware. While such a solution may be feasible in a homogeneous ad-hoc network, it is unrealistic to assume that all nodes in a heterogeneous DTN will be equipped with the same hardware. A second problem is the requirement that all nodes have complete or, at the very least the full path from sender to destination, such as in [38]. This is necessary because the cost of sending a messages is a function of the number of intermediary hops. However, as argued before, this assumption does not hold in DTNs.

Overall, it has been suggested that such incentive mechanisms for dealing with selfishly uncooperative nodes in ad-hoc networks (e.g., for the purpose of saving battery) are not really necessary: batteries last sufficiently long and the personal gain or broader economic value is too limited to warrant cheating (e.g., by modifying devices) [20]. This is in line with our general approach which assumes a basic level of cooperation and addresses only unintentional or malicious excessive resource consumption.

VIII. FUTURE WORK

Our work could be used in DTN routing implementations: The distance between buffer utilization by a set of nodes and their nominal buffer share provides additional input to the logic of acceptance and deletion of a messages. However, using this input in metric-based routing protocols needs further study.

In this paper we have concentrated on managing buffer space of DTN nodes: our schemes focus on admission and retention decisions rather than forwarding decisions. Investigating the effect of combined buffer and transmission management in DTN could be a subject of future work.

In the current experiments, we only considered individual messages from a sender to a receiver. But many applications

involve a message exchange, like a request and a response. Successful delivery of the request is useless if the response did not make it way back through. How can buffer management schemes be enhanced to take this into account? For example, a sender may issue a “ticket” for the receiver so that the response from the receiver can be assigned to the same priority class as the request by intermediate nodes. Working out the details of such ticket schemes is an avenue for future work.

Another line of future work is to investigate different variations of the Usage-biased Sub-domains approach. In the current simulations we used historical data observed by each node. One could extend this approach with different methods of collecting and processing that data. For example, we could relax the “local decision-making” assumption so that nodes can exchange recommendations about other nodes. A node can take such recommendations into account in biasing the thresholds.

IX. CONCLUSION

In this paper we have motivated and provided evidence for the problem of resource hogs in delay-tolerant networks. We have shown through simulations that a 10% minority of resource hogs can cause the delivery ratio of honest users in the DTN to suffer. In order to combat this problem we presented a buffer management solution that protects honest users while still allowing messages from unknown users to be carried and forwarded through the network.

The coarse-grained variant of the buffer management scheme uses the concept of domains. From the point of view of a single node, this effectively partitions the network into two classes of users: insiders who belong to the same domain as the node, and outsiders.

Our buffer management schemes are enabled by message authentication, which prevents resource hogs from forging source addresses.

When the insider domain size is large (80% of the total population in our experiments) the coarse-grained variant was effective in protecting against hogs among outsiders. To protect against hogs among insiders, we need fine-grained variants such as Equal or Usage-biased Sub-domains, which allows the behavior of individual entities within the insider domain to be monitored and taken into account in buffer management. Equal Sub-domains is the simplest and most effective among the schemes that we have presented.

Our mobility model causes random mixing of messages from different message sources in the buffers of intermediate nodes. Combined with FIFO transmission this leads to division of transmission opportunities between message sources in proportion to their share of intermediate nodes buffers. For instance, in the case of Equal Sub-domains this leads to equal division of transmission opportunities between domain members during congestion. Similar effect can be achieved independently of mobility model by combining our buffer management schemes with transmission of messages in random order.

In summary, we have investigated the scenario in which a majority of the nodes can authenticate each other’s messages and a minority of the nodes are outsiders. Our method for dealing with resource hog nodes requires only local information available within a DTN node and has three components:

1. Message authentication prevents source address forging and enables to distinguish between insiders and outsiders.
2. Coarse-grained domain management deals with misbehaving outsiders.
3. Fine-grained domain management deals with misbehaving insiders.

Resource management in DTN nodes is of great practical interest. We hope this paper will facilitate further results in that area.

REFERENCES

- [1] E. Altman, A. Kherani, P. Michiardi, and R. Molva. Non-cooperative forwarding in ad-hoc networks. *Lecture Notes in Computer Science*, 3462/2005:486–498, May 2005.
- [2] Ross Anderson and Tyler Moore. The economics of information security. *Science*, 314(5799):610–613, October 2006.
- [3] N. Asokan, Kari Kostiaainen, Philip Ginzboorg, Jörg Ott, and Cheng Luo. Applicability of identity-based cryptography for disruption-tolerant networking. In *Proceeding of ACM MobiOpp’07*, June 2007.
- [4] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proceedings of ACM SIGCOMM’07*, August 2007.
- [5] Bob Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM Computer Communications Review*, 37(2), April 2007.
- [6] Sonja Buchegger and Jean-Yves Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing (EUROMICRO-PDP)*, January 2002.
- [7] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the confidant protocol. In *MobiHoc ’02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 226–236, New York, NY, USA, 2002. ACM.
- [8] John Burgess, George Dean Bissias, Mark Corner, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of IEEE INFOCOM’06*, April 2006.
- [9] John Burgess, George Dean Bissias, Mark Corner, and Brian Neil Levine. Surviving attacks on disruption-tolerant networks without authentication. In *Proceeding of ACM MobiHoc’07*, September 2007.
- [10] Scott Burleigh, Adrian Hooke, Lee Torgerson, Kevin Fall, Vint Cerf, Bob Durst, and Keith Scott. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, 2003.
- [11] Levente Buttyán and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc wans. In *MobiHoc ’00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 87–96, Piscataway, NJ, USA, 2000. IEEE Press.
- [12] Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET)*, special issue on Mobile Ad Hoc Networks, 2002.
- [13] Vint Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-Tolerant Network Architecture. RFC 4838, April 2007.
- [14] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *P2PECON ’05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, New York, NY, USA, 2005. ACM.
- [15] Dartmouth campus traces. Available at: <http://crawdad.cs.dartmouth.edu/meta.php?name=dartmouth/campus>.
- [16] UMass DieselNet. <http://prisms.cs.umass.edu/dome/index.php?page=umassdieselnet>.

- [17] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of ACM SIGCOMM'03*, August 2003.
- [18] The free rider problem. Entry in the Stanford Encyclopedia of Philosophy, May 2003. Available at: <http://plato.stanford.edu/entries/free-rider/>.
- [19] Garrett Hardin. The tragedy of the commons. *Science*, 162:1243–1248, December 1968. Available at: <http://www.sciencemag.org/cgi/content/full/162/3859/1243>.
- [20] Elgan Huang, Jon Crowcroft, and Ian Wassell. Rethinking incentives for mobile ad hoc networks. In *Proceedings of the ACM SIGCOMM Workshops*, 2004.
- [21] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Proceedings of ACM SIGCOMM'04*, August 2004.
- [22] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebraNet. In *Proceedings of ASPLOS'02*, October 2002.
- [23] Ari Keränen and Jörg Ott. Increasing reality for dtn protocol simulations. Technical report, Helsinki University of Technology, Networking Laboratory, July 2007. Available at: <http://www.netlab.tkk.fi/tutkimus/dtn/theone/>.
- [24] Anders Lindgren, Avri Doria, and Olov Schelen. Probabilistic routing in intermittently connected networks. In *Proceedings of The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR)*, 2004.
- [25] Haiyun Luo, Paul Medvedev, Jerry Cheng, and Songwu Lu. A self-coordinating approach to distributed fair queueing in ad hoc wireless networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.
- [26] J. Medhi. *Stochastic models in queueing theory*, chapter 3.3. Academic Press, 2 edition, 2003.
- [27] Disruption tolerant networking. <http://www.darpa.mil/sto/solicitations/DTN/>.
- [28] J. Nagle. RFC 970: On packet switches with infinite storage, December 1985.
- [29] Luciana Pelusi, Andrea Passarella, and Marco Conti. Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44(11):134–141, November 2006.
- [30] Alex Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, 37(1):78–83, January 2004.
- [31] Matthew Seligman. Storage usage of custody transfer in delay tolerant networks with intermittent connectivity. In *Proceedings of ICWN'06*, June 2006.
- [32] Thrasyvoulos Sphyropoulos, Konstantinos Psounis, and Cauligi Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of ACM SIGCOMM'05*, August 2005.
- [33] Thrasyvoulos Sphyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, 2005.
- [34] Amir Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.
- [35] Brenton Walker, Joel Glenn, and Thomas Clancy. Analysis of Simple Counting Protocols for Delay-Tolerant Networks. In *Proceedings of ACM SIGCOMM'07 Workshop on Challenged Networks (CHANTS)*, September 2007.
- [36] Wizzy digital courier. <http://www.wizzy.org.za>.
- [37] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys and Tutorials*, 8(4):24–37, January 2006.
- [38] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, April 2003.