Retiring Replicants: Congestion Control for Intermittently-Connected Networks

Nathanael Thompson, Samuel C. Nelson, Mehedi Bakht, Tarek Abdelzaher and Robin Kravets
Dept. of Computer Science, University of Illinois at Urbana-Champaign
Email: {nathomps,snelso20,mbakht2,zaher,rhk}@cs.illinois.edu

***Abstract*—The widespread availability of mobile wireless devices offers growing opportunities for the formation of temporary networks with only intermittent connectivity. These intermittently-connected networks (ICNs) typically lack stable end-to-end paths. In order to improve the delivery rates of the networks, new store-carry-and-forward protocols have been proposed which often use message replication as a forwarding mechanism. Message replication is effective at improving delivery, but given the limited resources of ICN nodes, such as buffer space, bandwidth and energy, as well as the highly dynamic nature of these networks, replication can easily overwhelm node resources. In this work we propose a novel node-based replication management algorithm which addresses buffer congestion by dynamically limiting the replication a node performs during each encounter. The insight for our algorithm comes from a stochastic model of message delivery in ICNs with constrained buffer space. We show through simulation that our algorithm is effective, nearly tripling delivery rates in some scenarios, and imposes no or little overhead.**

I. Introduction

The proliferation of wireless-enabled mobile devices has vastly increased the opportunities for spontaneous communication between devices. However, in many cases the spontaneous networks formed between these devices are unstable and prone to partitioning and extended periods of disconnectivity [1], [2]. To overcome these network constraints and support both delay- and disruption-tolerant applications, a new class of routing protocols for these intermittently-connected networks (ICNs) has emerged that forward messages using a store-carry-and-forward approach. Given the lack of stable end-to-end paths, message replication is commonly used to increase delivery [3]–[9]. However, the limited resources available to the mobile nodes can quickly be overwhelmed by too much replication which leads to congestion and ultimately to reduced delivery rates. Given the dynamic nature of such networks, dynamic replication management, in terms of when, which and how many messages to replicate at a given encounter, can be used to find a balance between over- and under-replication, both of which reduce the effectiveness of the network.

The goal of replication management is to maximize message delivery by avoiding congestion. However, the point at which congestion occurs is not static in an ICN because the network size, density and message generation rates often change frequently. Any static replication management will either over-replicate, causing excessive message drops and reducing message delivery when congestion is high, or under-replicate, missing potential delivery opportunities and under-delivering when congestion is low. Existing ICN protocols attempt to avoid congestion either by capping replication with a fixed quota [7], [8], intelligently filtering replication opportunities [3]–[6] or, sometimes in conjunction with the other mechanisms, flushing already delivered messages from the network with acks [3], [4], [10]. While these approaches have proven to be effective in different scenarios, all of these replication management approaches are static and so cannot react to changes in network congestion.

The main challenge to effective replication management is the accurate detection of congestion in the ICN. In general, congestion can be defined as the point where network-wide delivery rate decreases due to an overload of network resources such as buffer space or bandwidth. In ICNs, buffer space is often overwhelmed by aggressive replication. While buffer management and flow-based feedback have been successful at preventing Internet congestion [11], [12], the lack of stable or even any end-to-end paths in ICNs eliminates the use of flow-based or end-to-end mechanisms. However, observations about global and local network conditions can be used to effectively reduce congestion and increase delivery rates.

An ideal ICN congestion detection scheme would monitor the entire network to learn the current congestion level and feed that information back to all nodes. However, because of high and variable message delays and unreliable delivery in ICNs, a global algorithm is impractical. Instead, a more attainable approach is to have each node act autonomously, using only *local* metrics to adapt its replication rate. A local congestion detection algorithm is still able to cope with spatial-diversity in congestion and can respond quickly to temporal congestion changes. The hard part of this approach is to determine which local metrics can be used to capture the congestion state of the network.

The goal of our research is to gain an understanding of the global behavior of congestion in ICNs and develop effective local congestion control, ultimately leading to increased message delivery in ICNs. Given these goals, the main contributions of our research are two-fold. First, we present a comprehensive study of the effect of diverse network conditions and limited buffer space on message delivery in ICNs. The model presented in this paper enables the tracking of relevant global network metrics over diverse network scenarios, exposing which metrics indicate congestion in the network. Although not directly implementable as a replication management scheme in ICNs, our model leads directly to an

		Buffer Size (KB)					
		1000	2500	3750	5000	7500	10,000
Message Period (sec.)	20	2	2	2	3	3	4
	40	2	3	4	5	6	9
	60	3	3	4	6	11	11
	80	2	5	7	9	13	18
	100	2	5	8	10	18	20
	120	3	6	11	16	20	20
	200	4	12	14	20	20	20
	300	9	18	18	20	20	20
	400	9	20	20	20	20	20

TABLE I
THE NUMBER OF COPIES FOR SPRAY AND WAIT WHICH GIVES THE BEST DELIVERY RATE FOR THE GIVEN MESSAGE PERIOD AND BUFFER SIZE.

understanding of which local metrics can be used by individual nodes to approximate the relevant global metrics, and so be used effectively in replication management algorithms. Second, we design an algorithm for congestion control in ICNs, including congestion detection and replication management. Based on the insight from our model, our ICN congestion control algorithm adjusts replication rates at individual nodes to maximize delivery rates. By allowing each node to react independently, our algorithm is responsive to both spatial and temporal fluctuations in congestion. Additionally, our congestion control is protocol-independent. It does not interfere with the forwarding decisions of the underlying routing protocol about which messages to send. Instead, it limits the number of messages a node is allowed to replicate during each encounter. We show through simulation over a diverse set of mobility scenarios that our approach is effective at improving network-wide delivery rates and does so with little overhead.

In the next section, we further elaborate on the challenges of congestion control in ICNs. Following that, in Sec. III, we develop a model to provide insight into the behavior of ICNs under congestion conditions. The results from analysis of our model are then used to inspire our design of our local congestion control algorithm described in Sec. IV and evaluated in Sec. V.

II. CONGESTION IN ICNS

Congestion in ICNs is caused by over-replication in the network, which leads to buffer overflow at individual nodes and ultimately to reduced delivery rates in the network. While it is intuitive to approach the problem from a sender's perspective, allowing the sender to regulate replication for its messages, two problems arise. First, intermittent connectivity, path diversity, high loss rates and long response times make it difficult for the sender to react to congestion in a timely and effective manner. Second, replication is performed at every node that has a replica of the message. Therefore, a sender cannot know how every node that will receive a replica of its messages should react. These challenges make it ineffective to apply flow-based and end-to-end congestion control approaches from the Internet [12]. Instead, congestion control in ICNs becomes a problem of managing replication at every node in the network.

Effective replication management requires firmly limiting the number of replications allowed. Existing quota-based ICN protocols perform message-based replication management where each message sender gives a static initial quota for the maximum number of replicas of a message [7], [8], [10]. As messages are replicated, the quota is divided between the new and old replica until each replica has no quota left. While using quotas certainly limits replication, the message sender does not know the congestion levels in other parts of the network or at later times when the message will be replicated. In dynamic networks, the best choice of initial quota can vary significantly depending on congestion conditions. For example, consider a simple network with 100 nodes, each using random waypoint mobility and routing with Spray and Wait [8]. By varying the maximum quota from 2 to 20 and varying the message rate and node buffer size from 1000KB to 10000KB in simulation, the initial quota that achieves the highest delivery rate varies dramatically, ranging from 2 to 20 as shown in Table I. Compared to a fixed initial quota of 8, which offers a nice balance of low delay versus low overhead in a network of this size [8], the performance improvement by selecting the best quota value is on average 7% with a maximum improvement of 20%. Even in this simple network, it is clear that the quota that maximizes delay depends on the state of the network.

Given the limitations of message-based replication, we explore the use of dynamic node-based replication management in ICNs where individual nodes determine how many messages they can replicate at each encounter. The novelty of our node-based replication management is that it is performed locally, independent of the routing protocol used. Essentially, a routing protocol orders the messages that it wants to transfer at each encounter and the congestion control algorithm determines how many of them are actually sent. To determine such local replication limits, nodes must observe the current level of congestion in the network. In the next section, we answer the question as to which metrics are most effective to detect congestion in ICNs and then present and evaluate our measurement-based local congestion control algorithm in the remainder of the paper.

Although buffer management has proven effective for avoiding congestion in the Internet [11], [13], [14], those approaches are not applicable for controlling congestion in ICNs since they assume some interaction with end-to-end transport protocols. However, some existing ICN protocols do use buffer management in ICNs for dropping and forwarding according to a given policy [3]–[6]. While policies can reduce the amount of replication in the network, in all of these protocols if every message in the buffer matches the policy, then every message will be replicated. Without knowing a firm limit of allowable replications, it will be difficult to avoid over-replication in congestion situations. Although not entirely orthogonal to the decision about replication limits, combining replication management with effective buffer management can dramatically improve delivery rates. In our evaluations, we demonstrate the benefits of such a combination using a routing protocol with policy-based buffer management and our replication management. However, we leave the full exploration of different policies for future work.

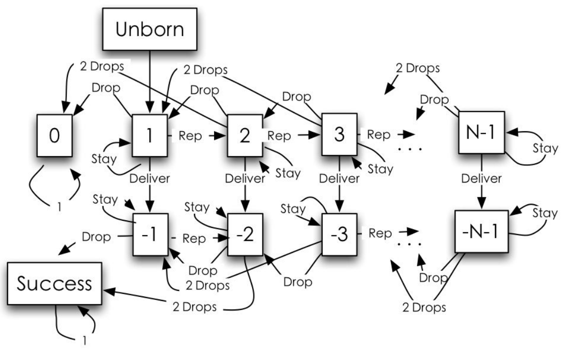

Fig. 1. Markov chain representing the spread of a single message through the network. Each state represents the number of replicas of the message. Special states represent delivery of the message and the pre-creation state. State transitions occur on message drop, replication and delivery.

III. Global Congestion Behavior

Before we can achieve our goal of developing a local congestion control algorithm, we must first understand the behavior of the network under congestion. Given an understanding of the high-level global behavior, local approximations can be developed. The purpose of our study is to find metrics that can track the change in delivery rates due to buffer congestion. Given the large number of dynamic network metrics, the challenge lies in determining which network metrics are the most informative. In any type of network, a decrease in the number of successful or duplicate deliveries or acknowledged messages could indicate an increase in congestion. Similarly, a rise in message drops and buffer usage can also indicate an increase in congestion. In ICNs, the number of message replications or total message copies might also be indications of network congestion. The relation of these metrics to delivery rate depends on complicated interactions of different network parameters. Therefore, as an analytical tool to reason about global congestion behavior, we develop a simple stochastic model that simulates the delivery of messages in networks with constrained buffers. The model is an approximate representation of an ICN and provides interesting insights into the effects of congestion that inspire the design of our congestion control algorithm in Sec. IV.

An effective approach for modeling ICNs is to treat the spread of a message as the spread of a disease [15]. When two nodes have an encounter, they "infect" one another with new messages. While this modeling approach has been used before [16], [17], the models either do not account for the effects of buffer overload and message drops in the network, or are asymptotic and do not capture per-encounter behavior which we use to track metrics. Other analytical models [3], [18] do not consider delivery rates and therefore cannot quantify the effects of congestion on successful delivery. In comparison, our model computes the network delivery rate based on the spread of messages throughout the network in response to changes in the input parameters network size, buffer size, contact rate, message generation rate and replication rate, each of which effect delivery and therefore are key to our model.

Our model tracks message replicas using a Markov chain. Each state in the chain denotes the number of message replicas present in the network. Therefore, the Markov chain starts with potential states $\{0, ..., (N-1)\}$, where N is the number of nodes in the network. A special state SUCCESS represents the successful delivery of a message. When a message is first created it always enters the state 1. The final states of a message are either SUCCESS, for successful delivery, or 0, for no replicas remaining. State transitions occur as message replicas are copied, dropped or delivered. The probability of each state transition depends on the model input parameters and will be given in detail below.

Even after successful delivery, some message replicas will remain in the network. Therefore, our model includes special states $\{-1, ..., -(N-1)\}$, where state $s \in \{-1, ..., -(N-1)\}$ indicates that there are $|s|$ replicas of the message remaining in the system after delivery. The final state -1 transitions into SUCCESS instead of 0 when the last message copy is dropped, since the message has already been delivered.

To accurately model drops the state of all messages must be known. The above model can be modified to include super-states which encapsulate the current states of every message. The probability to transition between a state is the probability that each message transitions into the new super-state. Messages are created sequentially during the simulation depending on the message generation period. The state UNBORN is added to signify that a message has not yet been created. Fig. 1 illustrates the complete Markov chain for a single message.

A. System Model

The network consists of N nodes and M messages created over time. Every node has the same size message buffer, capable of holding BS messages. Messages are all the same size. Because we are only concerned with high-level network behavior, our data and mobility models are simplified to ease modeling. New messages are generated periodically with sources and destinations selected uniformly at random from the entire network. We also assume that encounters occur globally at a fixed period with each participating node selected uniformly at random from the network.

A random drop policy is implemented, so that when a node must drop a message, it selects one at random to be removed. All nodes follow a basic Epidemic forwarding policy [9], where each node replicates to the other node messages it has that the other node does not have. In our model, messages are transferred instantaneously so that a message enqueued to be sent to the other node is not dropped by incoming messages before it can be sent. We do not model the effect of limited contact duration, focusing instead on buffer space. Our later simulations do include limited contact duration between nodes.

B. Transition Probabilities

We model the network as a discrete-time system with minimum time interval Δt. At every time step, the model transitions into a new state with some probability based on the current network state. Two network events can cause message transitions: a node contact or generation of a new message. To model these events, we first define the contact and generation periods. The contact period is expressed as $C\Delta t$ and the message generation period as $G\Delta t$. We assume that only one event occurs during each interval. If two events do occur in the same interval, one is selected with probability $1/2$. The probability of a contact event:

$$P_C = \left(\frac{1}{C} \cdot \left(1 - \frac{1}{G}\right)\right) + \left(\frac{1}{C} \cdot \frac{1}{G}\right)\left(\frac{1}{2}\right), \tag{1}$$

where the first term represents the probability that a contact event occurs with no generation event and the second term the probability that the contact event is selected when two events occur simultaneously. The probability of a generation event, P_G, is defined similarly.

The state of each message m_i is stored in s_i. When referring to a specific message or message state, we use the subscript m_i (s_i). Otherwise, m and s are used to indicate a general message. State transitions are defined by the input parameters, BS, N, C and G. The initial state for every message is the `UNBORN` state, signifying that the message has not yet been created. A new message is created every time step with probability P_G. The probability of message m_i entering the network at a given time depends on the probability of message m_{i-1} having already been created. Explicitly, m_0 transitions from `UNBORN` to state 1 with probability P_G. For each successive message m_j, the creation probability is $(1 - Pr(s_{j-1} = \texttt{UNBORN})) * P_G$.

Once created, a message can be delivered during any contact, which causes the state to transition from s into $-s$. The probability to deliver a message is:

$$P_{deliver} = P_C \cdot \frac{2}{N} \cdot \frac{s}{N-1}. \tag{2}$$

Essentially, a delivery happens during a contact event (P_C) in which one of the two nodes is the destination, and the other node has a replica of m.

The only other allowable transitions from state s are into $(s+1)$ upon replication, into $(s-1)$ or $(s-2)$ upon drop, or to remain in s. The transitions from $-s$ to states $-(s+1)$, $-(s-1)$ and $-(s-2)$ are the same. For contact events, a and b represent the two nodes involved in the contact event. Similarly, for generation events, a represents the node that generates the new message.

All of these transitions depend on the behavior of nodes during arrival of new messages. The following equations describe node behavior:

$$\alpha = \sum_{m_i} \left(\frac{s_i}{N}\right) \cdot \left(1 - \frac{s_i - 1}{N-1}\right) \tag{3}$$

$$P_{full} = \sum_{m_i} \left(\frac{s_i}{N}\right) \cdot \left(\frac{1}{BS}\right) \tag{4}$$

$$P_{drop} = P_{full} * \cdot \frac{R \cdot \alpha}{(BS + R \cdot \alpha)} \tag{5}$$

When two nodes encounter one another, each node has some number of messages that the other node does not have that might be transferred. α represents the expected number of messages to be transferred and is calculated as the sum over all messages of the probability that a has the message and b does not. After an encounter, a node will have to drop messages if its buffer is full. The probability that a node's buffer is full, Eq. 4, is the probability for each message that a node has the message, over the node's capacity. Finally, the probability of message m being dropped, Eq. 5, is the probability of it being one of the randomly chosen messages selected for removal when an overflow happens.

The number of messages actually transferred from one node to the other depends on the replication probability, R, which captures the probability that a particular message is chosen for replication. R ultimately determines the fraction of messages a node is allowed to replicate so that only $R \cdot \alpha$ messages are transferred during a given encounter. If the node's buffer is full, the same number of messages will have to be dropped. Therefore, m is dropped if it is selected among the $R \cdot \alpha$ removed messages. The probability that a node keeps m after a transfer is $P_{keep} = 1 - P_{drop}$. The parameter R in Eq. 5 allows nodes to control the replication probability and so manage congestion. In Epidemic, R is always fixed to 1. We will later analyze different values of R.

To transition from state s to $s+1$, there must be a contact event in which a replication occurs and both nodes keep m:

$$P_{+1} = P_C \cdot \frac{N-2}{N} \cdot \frac{((N-1)-s) \cdot s}{\binom{N-1}{2}} \cdot R \cdot (P_{keep})^2 \tag{6}$$

The contact must be between two nodes neither of which is the destination, which is expressed in the first two terms of Eq. 6. For a replication to happen, only one node can have m, the probability of which is the number of pairs with only one copy over all possible pairs, given in the third term. The last two terms say that the message must be selected for transfer, and then both nodes must keep their copy.

If, on the other hand, both nodes have m and both drop m, then s transitions to $s-2$:

$$P_{-2} = P_C \cdot \frac{N-2}{N} \cdot \frac{\binom{s}{2}}{\binom{N-1}{2}} \cdot (P_{drop})^2 \tag{7}$$

This transition only happens if the event is a contact event in which neither node is the destination. If both nodes have m, expressed as the number of pairs that both have m over all pairs (see Eq. 7), and both nodes drop their replica then s transitions to $(s-2)$.

Finally, the transition from s to $s-1$ is the most complicated transition. A single drop of m can occur during a message generation event, a contact event in which both nodes have m or a contact in which only one node has m:

$$\begin{aligned} P_{-1} = {} & P_G \cdot \frac{s}{N} \cdot P_{drop}(\alpha = 1) + \\ & P_C \cdot \frac{N-2}{N} \cdot \left(\frac{\binom{s}{2}}{\binom{N-1}{2}} \cdot (2 \cdot P_{drop} \cdot P_{keep}) + \right. \\ & \left. \frac{((N-1)-s) \cdot s}{\binom{N-1}{2}} \cdot \left((1-R) \cdot P_{drop} + R \cdot (P_{drop})^2\right) \right) \end{aligned} \tag{8}$$

In a message generation event, a drop occurs if the generating node has m and m is selected for removal because of the new message. In a contact event between two non-destination nodes which both have m, a single drop occurs if either node keeps m and the other drops it. Finally, if only one node of a contact has m, it might replicate m with probability R. s transitions to $(s-1)$ if m is replicated and both nodes drop it, or if m is not replicated and the one node drops it.

C. Solving the System

To solve the model, we repeatedly apply the above transition probabilities between states to converge toward either the 0 or

Algorithm 1 TRANSITION$(M, states, endtime)$

Require: S $= M$ state vectors of length $states$
Require: T $= M$ transition matrices size $(states$ x $states)$

```
1: for all messages $m$ do
2:     S$(m)$(UNBORN) $= 1$
3: for $t = 0$ to $t = endtime$ do
4:     for all messages $m$ do
5:         T$(m) =$ TRANSITIONMATRIX$(m, \mathrm{S})$
6:     for all messages $m$ do
7:         S$(m) =$ S$(m) *$ T$(m)$
```

the SUCCESS state for each message. A transition matrix is created using the above equations. The current state of each message is represented by a vector of probabilities to be in each state. By repeatedly multiplying the transition matrix with the state vector, the probability distribution across states after a certain time can be found. When super-states are used to track all messages, there are $O(M^N)$ states which is unwieldy for meaningful networks. Instead, to solve the system we maintain a separate chain for each message and update the transition matrix at every time step using the current states of the other messages. The current state of a message is computed as $\sum_{\text{state } t} Pr(s_i = t) * |t|$. Alg. 1 shows the process for solving the system.

D. Detecting Congestion

The goal of our model is to discover how different metrics change compared to the network-wide delivery rate and if they can be used to track congestion. At first glance it seems that an increase in network drops would be sufficient to indicate network congestion. However, consider the case when there is no replication at all. Drops will be very low, but so will delivery. As replication increases delivery will increase along with drops. This rise in drops is by itself not an indication of over-replication, i.e. congestion, since delivery is still increasing. Instead, the positive aspect of replication must also be consider in conjunction with the negative (drops). We use the model to evaluate different metrics as either positive or negative indicators of congestion.

To generate congestion we decrease the size of the message buffer. Because the model simulates individual node contacts, it is possible to count drops, replications, and max buffer consumption and track the spread of acks. The expected spread of acks at a given time t is the probability that m has been delivered times $a(t)$. The function $a(t)$ represents the copies of an ack after time t and is computed recursively as the probability that during a contact one of the nodes has the ack and the other does not. The probability of having the ack depends on $a(t-1)$ where $a(0) = 1$.

To discover the behavior of these metrics, the model was solved using Alg. 1, as shown in Fig. 2. From the figure it can be seen that drops and buffer consumption increase with congestion while acks and message replications tend to decrease with congestion. Thus, the ratio of either drops or buffer usage over replications or acks should give a good indication of network congestion by capturing both the negative and positive aspects of message replication. Two observations from outside the model help us determine the best metrics.

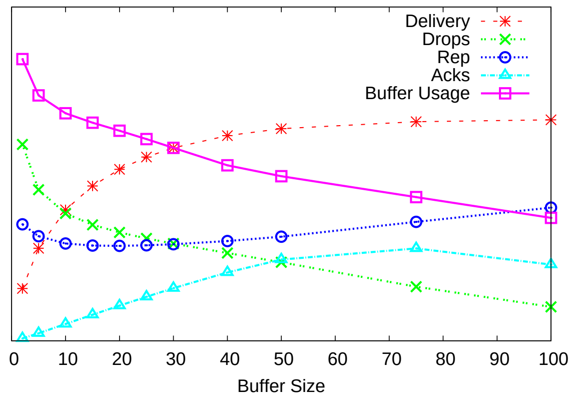

Fig. 2. Change in metrics as congestion is increased by varying buffer size.

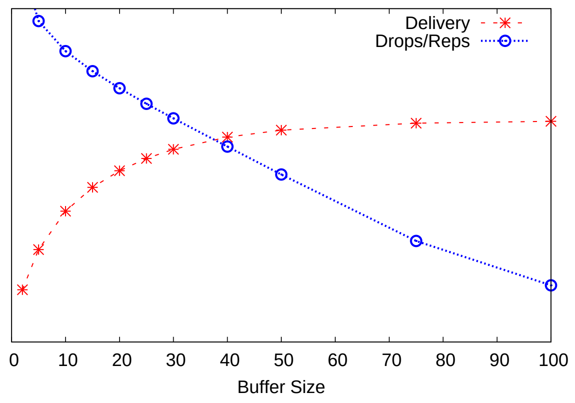

Fig. 3. Change in measured congestion (ratio of drops over replications) and delivery as congestion changes.

In most simulations, the buffer consumption is always near 100%, even in low congestion. Also, the spread of acks is unreliable and delayed. Therefore, we use the ratio of drops over replications to track congestion. This ratio should grow with congestion, inverse of delivery rate, which holds in the model as shown in Fig. 3.

E. Limiting Replication

We have left the discussion of the variable R until now. R represents the probability of forwarding a message and is used to limit replication. R is the only parameter that nodes can change and therefore the means by which replication management is done. The function for R depends on the replication management approach. Our node-based limiting approach specifies the maximum number of messages, L, transferred during each contact. The probability of m being forwarded is the probability that m is one of the L messages selected for transfer. Assuming a random forwarding policy, $R_{limit} = \min\left(\frac{L}{\alpha}, 1.0\right)$, where α is from Eq. 3.

By varying L and computing the delivery probability in different scenarios, it was found that the value of L resulting in the best delivery ranged from 1 to the maximum value. However, because of the simplifications of our model, the full performance impact is not captured. To fully understand the possible impact of node-based replication management via L, we turn to simulation. By fixing the replication limit to

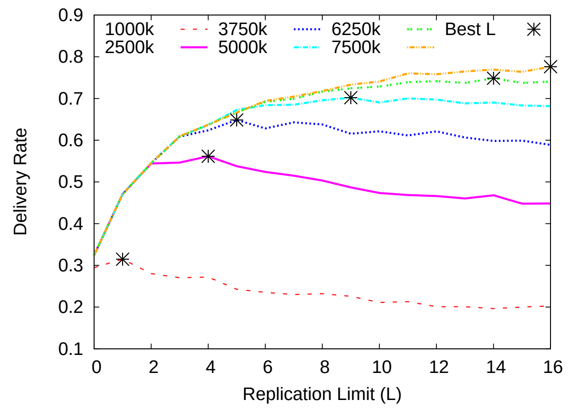

Fig. 4. The change in delivery rate as L is varied with different buffer sizes. The best L value is marked for each scenario.

different values we can compare the delivery rates at different levels of congestion. Choosing the wrong value of L can result in a 10-40% change in delivery rates (see Fig. 4).

IV. ICN Congestion Control

Given the disconnected and dynamic nature of ICNs, congestion control must act locally while thinking globally. Our model in Sec. III highlights which global metrics can track congestion. In this section, we present our local congestion control algorithm based on those observations. Each node independently calculates a local approximation of the current congestion level, CV. As the node samples the network, it continually updates CV. Whenever CV is updated, the node also adjusts the replication limit L following an additive increase, multiplicative decrease (AIMD) algorithm.

To maintain CV, local metrics are needed to approximate the global metrics for drops and replications. Global drops can easily be approximated by measuring the local drops at an individual node. On the other hand, approximating global replications is not as straight forward. When a node replicates a message the node does not know for certain that the other node kept the replica. Therefore, replications are counted by successful *incoming* replications. The fidelity of these two metrics can be improved by including measurements from other nodes. When two nodes meet they exchange their current drop and replication counters. In addition, because a message is replicated along each hop it travels, the hop counts of stored messages give further replication information.

To calculate the congestion value, a node monitors the network for a certain time, and then computes CV' which is the ratio of drops and replications collected in the last sample. The drop and replication counts are reset for each sample period. To dampen the effects of temporary spikes in congestion, the new congestion value is calculated using an estimated weighted moving average (EWMA) with $\alpha = .9$ so that $CV' = \alpha \cdot CV_{sample} + (1-\alpha) \cdot CV$. The most recent sample is given higher weight to keep CV fresh.

After calculating CV', the node adjusts the replication limit. As CV moves up and down, the replication limit should grow to take advantage of all available resources, but back-off when congestion increases, similar to how TCP updates its

Algorithm 2 ProcessEvent($event$)

```
Require: $drops = 0$
Require: $reps = 0$
Require: $limit = 1$
Require: $CV = \infty$
Require: $ai > 0, 0 < md < 1.0$
 1: if $event$ = Drop then
 2:     $drops = drops + 1$
 3: else if $event$ = Receive Message then
 4:     $reps = reps + 1$
 5: else if $event$ = Contact with node $b$ then
 6:     $d = drops + b.drops$
 7:     $r = reps + b.reps + \sum_{m \in \text{stored messages} m} (hops(m) - 1)$
 8:     $reset(drops, reps)$
 9:     $CV' = \alpha \cdot (d/r) + (1-\alpha) \cdot CV$
10:     if $CV' \leq CV$ then
11:         $limit = limit + ai$
12:     else
13:         $limit = limit * md$
14:     $CV = CV'$
```

congestion window [12]. Because changes in the replication limit propagate slowly through the network, the limit should grow gradually so as not to overwhelm the network. If the new congestion value CV' is higher than the previous value CV, there is growing congestion and the limit is reduced by multiplying the current CV by multiplicative factor $(md < 1.0)$. If, instead, CV' is less than CV, the limit is increased by a fixed amount $ai > 0$. Experimentally, it was found that a lower md value, i.e. stronger back-off, coupled with a low ai, i.e. slower growth, resulted in the highest delivery rates in congested scenarios. We use $md = 0.2$ and $ai = 1$.

The choice of time length to sample the network is important to adjust the replication limit at an appropriate pace. CV should be updated as frequently as possible, but not too quickly to lose meaning. The minimum time to detect change in congestion is the minimum time for the congestion metrics to change. As illustrated in Sec. III, replications and drops can only occur during node contact and message generation. Since no replication occurs in message generation, the minimum time to detect change in both metrics is one contact. Whenever a node contact occurs the nodes update CV. Pseudo-code for the algorithm is shown in Alg. 2.

V. Evaluation

The goal of the evaluation is to show the delivery improvement attainable using congestion control across various degrees of congestion and in diverse network configurations. We evaluated the effectiveness of node-based congestion control by integrating it with several different routing algorithms. We ran each protocol with and without congestion control to demonstrate the protocol independent property of our congestion control algorithm. Epidemic [9] was the baseline protocol with no congestion control. For policy-based congestion control, we used the Prophet protocol, which selects messages for forwarding based on the likelihood of the contact delivering the message [5]. For message-based congestion control, we selected Spray and Wait which assigns a static initial quota to each message and distributes the quota in half at each encounter [8].

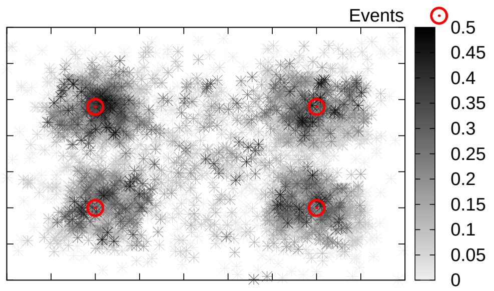

Fig. 5. Map of calculated CV at specific points in the "Events" scenario. Higher congestion values, indicated by darker points, are mostly clustered around the four event locations.

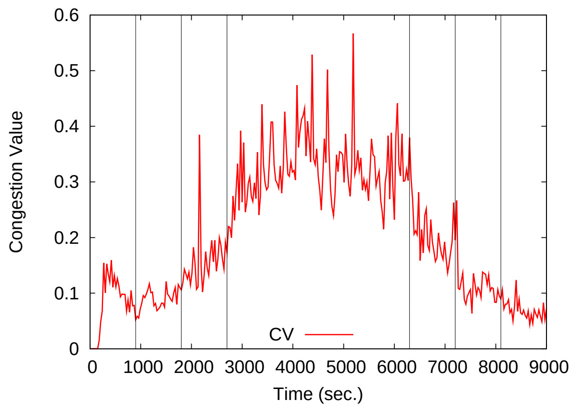

Fig. 6. Change in CV over time as the message generation rate fluctuates in the "Diurnal" scenario.

A. Experimental Set-up

To perform our evaluations, we added a transport layer in the ONE simulator [19]. The transport layer is responsible for performing all of the congestion related functions, assigning sequence numbers to messages and managing message acks. Acks are used to flush already delivered message replicas from the network. The transport layer is queried whenever the routing layer wants to send a message. If the replication limit has not yet been reached for the current contact, the transport layer allows the transmission. When a message is delivered to the final destination, a new ack is generated and inserted into a fixed size cache. Each node stores all received acks in its cache along with the drop and replication counts. At the start of a contact, the transport layer sends the cache to the other node, which merges the information with its own. Any messages for which an ack has been received are removed. Then the replication limit is updated for the ensuing contact.

In all simulations, new messages were generated following a Poisson process at each node with mean message period varying based on the simulation. Each message was 50KB and the destination was selected at random from the network. Each node had a buffer of 1MB. To vary the level of congestion in the network we changed the mean per-node message period. Each simulation had 100 nodes in a 4500 x 3400m area with transmission range of 150m and speed of 250kbps.

To capture various network conditions, several scenarios were used to determine nodes' movement. In the baseline case, random waypoint mobility was used with speeds between 2 and 15m/s The other scenarios create spatial or temporal variations in congestion. The first scenario contains several events in different locations. Nodes congregate around the event locations, increasing congestion due to higher neighborhood message generation rate. When nodes are near events they also increase their generation rates, so that a node's message generation rate is a function of the distance to the nearest event. This scenario, which we call "Events", loosely models the behavior of mobile agents in a disaster [20]. In the second scenario, the message generation rate is changed at regular intervals to temporally vary congestion. For the first steps, the rate is doubled. Once the peak rate is reached, the network remains in high traffic load for several steps before returning to the original low rate by halving at each step. This behavior simulates a day-time/night-time traffic pattern. We call this scenario the "Diurnal" scenario. Nodes move following random waypoint movement.

B. Accuracy of Congestion Detection

Before evaluating the effectiveness of the full congestion control algorithm, we first validated the local congestion detection component in isolation. We ran congestion control with CV calculation, but without the replication limit. Whenever a node updated CV, the timestamp, location of the node and CV were recorded.

In the "Events" scenario, congestion should be higher near areas of high node density, for example, at the event locations. In surrounding areas the congestion should decrease. Our congestion detection algorithm was able to capture this behavior, as can be seen by the map in Fig. 5. Each point is CV at that location in the space. Darker points indicate a higher CV. Interestingly, although the majority of high congestion was around the event locations, there were other areas of congestion in the spaces between events. This underscores the importance of dynamic, node-based congestion control because of unpredictable congestion conditions.

In the "Diurnal" scenario, node densities are fairly uniform due to the random waypoint movement. Changes in congestion instead come from increasing and decreasing message generation rates. Our congestion detection algorithm successfully captures both the increase and the decrease in congestion, as seen in Fig. 6. The mean CV values across the network were computed every 30 seconds. At the beginning there is a learning period as nodes begin to sample the network. After the initial period CV responds fairly quickly, moving both up and down in only a few update periods.

C. Benefits of Congestion Control

We next evaluated the full congestion control algorithm including dynamic replication limiting. We first considered the baseline scenario with uniform congestion. Some benefit of our algorithm comes from freeing buffer space by using acks to flush unnecessary messages. To quantify the benefit from the ack and congestion control components, we ran each protocol

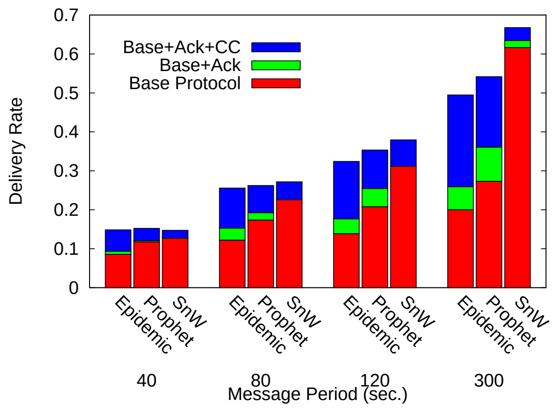

Fig. 7. Breakdown of delivery rate in varying congestion from each component: the base protocol (Base), acks (Ack) and congestion control (CC).

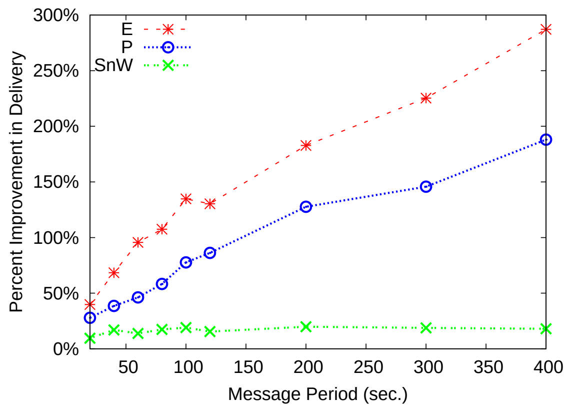

Fig. 9. The percentage improvement using congestion control over the base protocol as base message period increases in the "Diurnal" scenario.

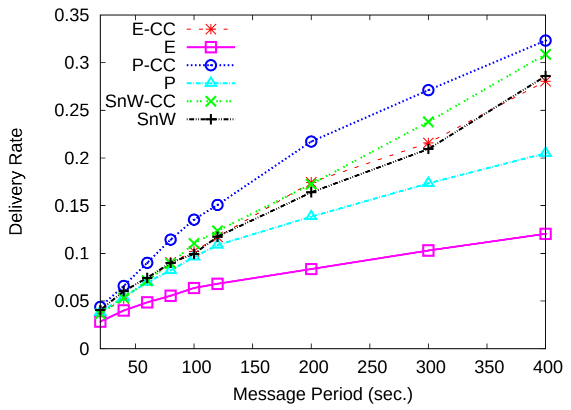

Fig. 8. The change in message delivery rate in the "Events" scenario as the message period increases with congestion control (CC) and without.

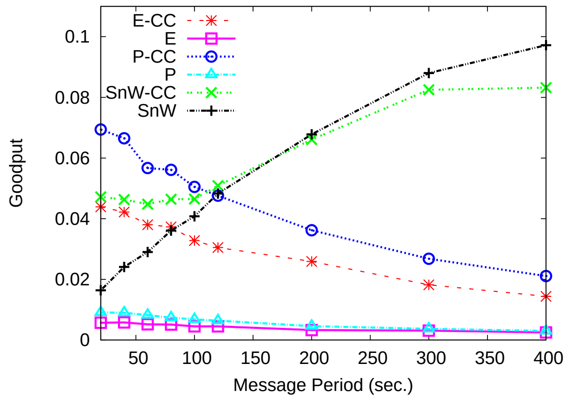

Fig. 10. Goodput of each protocol with and without congestion control (CC).

with full congestion control, with only acks and by itself. By reducing the message period, congestion was increased.

The experiment revealed several things (Fig. 7). First, at very high congestion, acks do not improve delivery because they do not propagate before the replicas have been dropped (see the Base+Ack bars). As congestion lessens, there is more benefit from acks, but the majority of delivery improvement comes from congestion control. Second, dynamic node-based congestion control improves delivery for each protocol in every scenario. Also, the effectiveness of node-based congestion control compared to policy- or message-based can be seen in the improvement of Epidemic. Epidemic by itself is very inefficient, but with node-based congestion control (Epidemic+CC) it generally outperforms the base policy-based (Base-Prophet) and message-based (Base-SnW) protocols in nearly every scenario. Despite the good performance of the basic Spray and Wait protocol, dynamic node-based congestion control improves delivery in all cases by at least 8%. Third, at the highest level of message generation the network is overloaded as seen by the low delivery rate. Newly generated messages quickly push stored messages out before the old messages have time to spread through the network. Even though the network can hardly operate, congestion control improves delivery by 15-73%.

Examining the performance of the different protocols under spatial congestion shows the benefits of node-based congestion control even more. In the "Events" scenario, node-based congestion control is well-suited for the changing congestion. All protocols with congestion control perform better or as well as all base protocols (see Fig. 8). Node-based congestion control does especially well when coupled with policy-based forwarding. This hints at the benefits of choosing an effective forwarding policy to pair with congestion control.

In the "Diurnal" scenario, the relative performance of the protocols is very similar to the "Events" scenario. Because of our dynamic replication management, our congestion control is always able to adapt to the current message load. This is illustrated by the percentage improvement when using congestion control compared to the base protocols, as shown in Fig. 9. In low congestion levels, dynamic congestion control is able to take advantage of the extra available resources, increasing Epidemic by up to 280% and Prophet by 190%.

D. Overhead

Finally, to evaluate the impact that congestion control has on the underlying protocols, we compare the change in goodput and delivery delay when using congestion control. Our congestion control adds overhead from acks. In addition, because replicas remain in the system longer, each message might be copied more often. The goodput, bytes delivered over total bytes sent, quantifies the trade-off between overhead

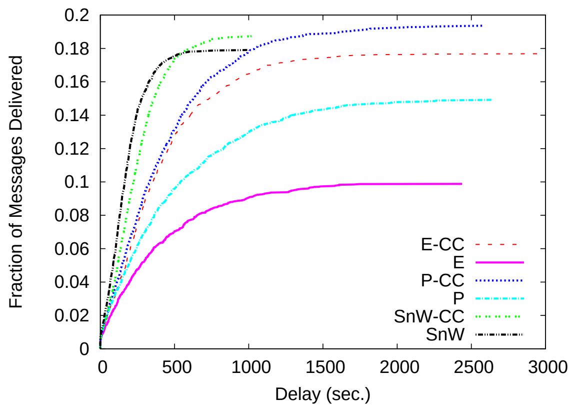

Fig. 11. CDF of message delays for delivered messages with different protocols with congestion control (-CC) and without.

and improved delivery. When operating in high congestion scenarios, congestion control limits the replication, thereby lowering overhead. This leads to a higher delivery rate and a higher goodput (see Fig. 10). As the congestion lowers, congestion control increases the replication limit, pushing down goodput. Used with the flooding protocols congestion control always has better goodput. Spray and Wait, because of its message quota, has a relatively fixed overhead and so delivery rate grows faster than overhead. However, in high congestion our congestion control has a significant impact on the overhead of Spray and Wait.

By limiting replications, congestion control increases the message delivery delay, but messages that otherwise would not be delivered are. This behavior can clearly be seen by examining the CDF of message delays between protocols from one run of the baseline scenario (see Fig. 11). The increase in delay is especially noticeable compared to Spray and Wait which is designed to reduce delay. However, the median increase in delay using congestion control compared to base Spray and Wait ranged at worst from only 6% to less that .1%. Because of some far outliers, the mean ranged from 237% in the worst case, to actually reducing delay by 10%. For Epidemic, the median increase in delay ranged from 3% to less than .1% and for Prophet the median increase was never more than .1%.

VI. Conclusion and Discussion

There is a clear opportunity to improve the delivery rate of ICN networks by responding directly to congestion in the network. In this paper we have shown that congestion in an ICN can be detected by tracking the ratio of drops over replications. Our dynamic node-based congestion control can increase delivery rates by up to 280% in some scenarios, outperforming all existing congestion avoidance schemes.

One challenge in our current implementation is determining the correct multiplicative back-off value, md. When congestion is high, it is best to back-off quickly. However, at a certain point the network quickly switches to an uncongested state and the strong back-off in replication is too aggressive and actually hurts delivery. As a future extension to the existing congestion control algorithm we will investigate how to dynamically scale md to prevent over- or under-reacting to congestion.

While our node-based congestion control is effective for controlling congestion of buffer space, there are two other limited resources in ICNs, energy and contact duration, which were not studied in this paper. One approach to saving energy is to reduce the total number of transmissions. Our congestion control already limits the replications in the network, and could easily be combined with an energy-focused limiting algorithm. Effective buffer management policies are also needed to prioritize messages for replication at each encounter, which might be shortened from replication limiting. Many such policies have been proposed in the literature and, as demonstrated in the evaluation, combining such a policy with our congestion control can lead to even higher delivery rates.

References

[1] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang, "Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing," in *ACM MobiCom*, 2007.

[2] M. Piórkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "On clustering phenomenon in mobile partitioned networks," in *Proceedings of ACM SIGMOBILE workshop on Mobility models*, 2008.

[3] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," in *ACM SIGCOMM*, 2007.

[4] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *IEEE Infocomm*, 2006.

[5] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[6] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *ACM MobiHoc*, 2008.

[7] S. C. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in dtns," in *IEEE Infocomm*, 2009.

[8] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *SIGCOMM Workshop on Delay Tolerant Networks*, 2005.

[9] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep. CS-200006, April 2000.

[10] T. Small and Z. J. Haas, "Resource and performance tradeoffs in delay-tolerant wireless networks," in *SIGCOMM Workshop on Delay Tolerant Networks*, 2005.

[11] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[12] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," Internet Engineering Task Force, RFC 2581, Apr. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2581.txt

[13] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *IEEE FOCS*, 2000.

[14] P. Kouznetsov, R. Guerraoui, S. Handurukande, and A.-M. Kermarrec, "Reducing noise in gossip-based reliable broadcast," *Proceedings IEEE Symposium on Reliable Distributed Systems*, 2001.

[15] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *ACM PODC*, 1987.

[16] T. Small and Z. J. Haas, "The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)," in *ACM MobiHoc*, 2003.

[17] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Computer Networks*, vol. 51, no. 10, pp. 2867–2891, 2007.

[18] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 77–90, 2008.

[19] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Simutools*, 2009.

[20] S. C. Nelson, A. F. H. III, and R. Kravets, "Event-driven, role-based mobility in disaster recovery networks," in *ACM CHANTS*, 2007.