An End-to-end Rate Control Protocol for Intermittently Connected Networks (ICNs)

Euiyul Ko*, Dohyung Kim*, Hanjin Park*, Ikjun Yeom†, and Euiseong Seo†

*Dept. of CS, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea

Email: {euiyul.ko, mr.dikim, yubi001}@gmail.com

†College of ICE, Sungkyunkwan University, Suwon, Republic of Korea

Email: {ikjun,euiseong}@skku.edu

Abstract

In this paper, we propose a rate control algorithm for intermittently connected networks (ICNs), which represent a type of delay/disruption-tolerant network (DTN). ICNs have quite different characteristics from traditional TCP/IP networks that lead to problems that do not occur in legacy networks, such as network partitioning, long and varying delays, high loss probability, and asymmetric data transmission rates. To overcome these issues, many researches have been carried out in recent years. The results have helped in achieving smooth communication between nodes, but there are still some weak points. If many messages are input by the nodes in a network, these schemes do not work well. For resolving this problem, we propose the novel sending rate control algorithm in an additive increase/multiplicative decrease (AIMD) manner. To detect network congestion, we measure the receiving rate, the one-way delay, and the number of copies. When measured receiving rate is increased, the congestion window (*cwnd*) is increased; otherwise, *cwnd* is decreased. Simulations show that the proposed algorithm can adjust the sending rate of nodes to avoid network congestion and provides fair share of the network for the nodes.

Index Terms

ICN, DTN, e2e congestion control, AIMD

I. Introduction

In traditional TCP/IP-based networks, such as the Internet, the main assumption is that there is always a fully connected path between two end nodes for communication between them. In other words, at least one contemporary, fully connected path between two end nodes should exist. Other key assumptions are that the maximum round-trip time (RTT) is not unrestrained and the end-to-end packet loss probability is small. While these assumptions are reasonable for the conventional Internet and some types of mobile network, such as mobile ad-hoc networks (MANETs), other types of network may violate one or more of these assumptions. For example, in wildlife tracking and habitat monitoring sensor networks, military mobile networks, terrestrial mobile networks, inter-planetary networks, and vehicular ad-hoc networks (VANETs), a fully connected path between two end nodes cannnot be

exist or RTT is too long. These networks are called *intermittently connected networks* (ICNs) [1] and represent a type of delay/disruption-tolerant network (DTN) [2].

ICN has attracted substantial research attention due to their variety of applications in military, disaster discovery, and emergency-response systems where the communication infrastructure may not exist [3]. Especially, ICN is expected to be popularly used for Car2Car communication, mobile Internet and intelligent traffic system.

An ICN has quite different properties to a traditional TCP/IP-based network. Fig. 1 shows the characteristics of ICNs. It may become unexpectedly partitioned or have no path between two nodes at times because of node mobility or other unexpected reasons. In addition, these networks do not guarantee a fixed RTT, which is long and highly variable. Since each packet is sent and received through different paths, it has asymmetric data transmission rates and the end-to-end packet loss probability is high. Because of these different characteristics, the term *Challenged network* is used and these networks may not be suitable for some services being used on the conventional TCP/IP networks.

Many researches have focused on overcoming these ICN problems in recent years. One of the well-known research groups is the Delay Tolerant Networking Research Group (DTNRG) [1]. They proposed a DTN architecture and many studies have investigated issues around the proposed network architecture. To achieve smooth communication between nodes in ICN, the group proposed new routing protocols. Resource allocation is also a major research topic because the computing resources such as memory, power and communication time, are critically limited in ICN.

Control of the sending rate is one of the most important issues for improving network performance. Although there is a very effective routing or resource allocation algorithm for ICNs, each nodes suffers in congestion when it pours packet in a network too much. Floyd and Fall investigated the negative impacts from increasing traffic in network without congestion control [4]. An increment of non-congestion-controlled traffic can cause unexpected congestion collapses and extreme unfairness among flows. If the sending rate is controlled as in TCP, each flow will be adapted to the bandwidth available for its path and, in turn, the network can avoid congestion collapse.

Lots of conventional protocols such as TCP and TFRC, which are currently being used for the Internet, incorporate sending rate control schemes. Among these, TCP is the most popular transport layer protocol and variations such as TCP/Reno, TCP/Vegas and TCP/Westwood have been introduced. Algorithms for TCP congestion control are based on self-clocking by an ACK corresponding to a data packet. Since there is no end-to-end path in ICNs, definition of the current available bandwidth is meaningless. In addition, a self-clocking mechanism may not function because the packet delivery ratio is too low and the feedback delay is too high. Thus, algorithms that control the sending rate are not directly applicable to ICNs.

Even though the end-to-end rate control is difficult to achieve in ICNs, some researchers have investigated congestion control using other approaches. Almost all the approaches proposed yield congestion control in layer 3 [5], [6], [7], [8]. Theses network layer approaches are based on an epidemic routing protocol proposed by Vahdat

[1] http://www.dtnrg.org

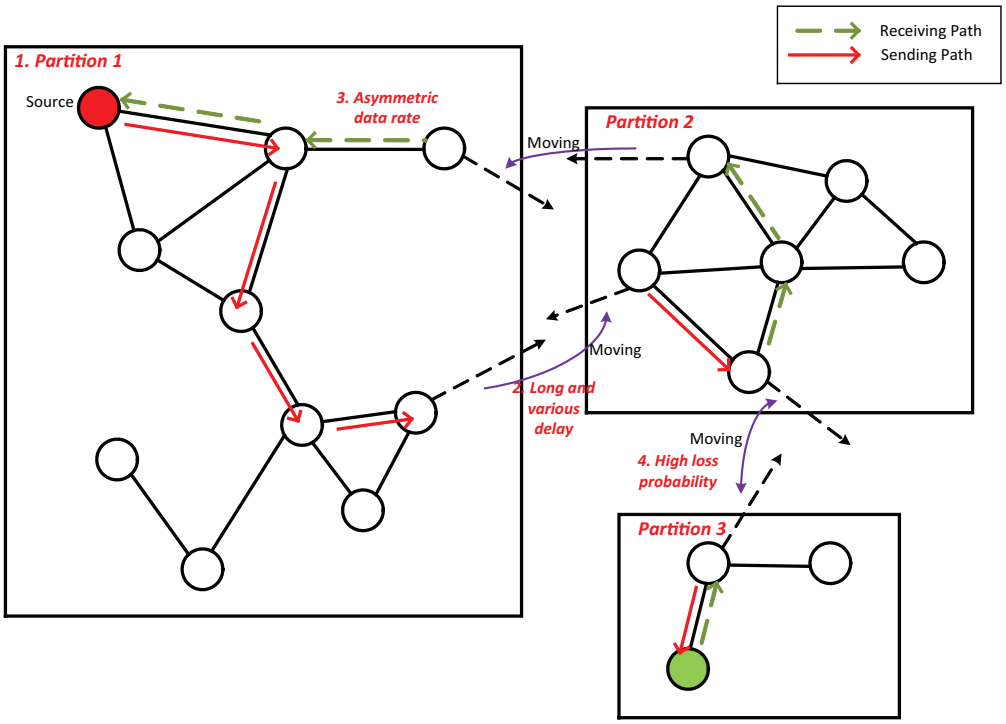

Fig. 1: Characteristics of ICNs

and Becker [9]. While these schemes control congestion in layer 3, they do not offer end-to-end congestion control. To the best of our knowledge, there is currently no scheme that provides end-to-end congestion control.

In this paper, we propose a novel end-to-end rate control algorithm for ICNs. In our scheme, control of the sending rate at each end node decreases network congestion collapses and, thus, the network utilization is maximized. To control the sending rate, the message drop time is measured for each node. When a node forwards a message to its destination via an one-hop path, receives a message, and drops a message in its buffer, it changes its congestion window using the measured message drop time. To change the congestion window *cwnd*, we propose an additive increase/multiplicative decrease (AIMD) algorithm. The one-way delay for self-clocking is also measured. The one-way delay is calculated when a node forwards a message to its destination and then receives a message. When a node sends a message, it will send the next message after the one-way delay measured.

The remainder of the paper is organized as follows. In Section II, we describe the background and related research in brief. In Section III, we define the problem to be solved and our proposed end-to-end sending rate control algorithm. We describe the performance of our algorithm and compare it with other related algorithms in Section IV. We conclude our research in Section V.

II. Background and Related Work

As mentioned above, there has been much research into ICNs in recent years. Many new protocols and network architectures have been developed to support more efficient communication in ICNs.

A major focus has been on methods to avoid congestion. Researchers in this field have mainly focused on routing protocols to resolve this issue [5], [6], [7], [8]. Although routing protocol primarily affects the performance of DTNs, however, modification of existing routing protocols usually induces costly changes in networking layers of all participating nodes while applying a new congestion control requires modification of only two end nodes.

Seligman et al. proposed storage routing (SR) for congestion control and storage management in layer 3 based on a store-and-forward routing approach [5], [7]. In their approach, the available storage is used to determine the suitability of message sending. In addition, collections of messages are migrated from local storage to one or more neighbors when congestion occurs and are then brought back when congestion weakens. Li et al. proposed a congestion control strategy called *N-Drop* [8]. If a buffer is full and a new message needs to be stored, all messages in the buffer are checked and messages with a forwarding number greater than N are erased. If there is no message with a forwarding number greater than N, the last message in the buffer is erased. Burleigh et al. developed a congestion control algorithm for DTNs [6]. Each router autonomously determines whether to accept a message or not on the basis of local information such as average storage and the value and risk of accepting a message derived from historical statistics. Other congestion control protocols have been proposed, but they are not significantly different to those described above. Thompson et al. proposed an AIMD style congesion control for ICNs [10]. They proposed a new replication management algorithm, which dynamically controls the number of replication limits based on the network conditions. The proposed dynamic control algorithm is based on the AIMD appproach similar to our approach. These protocols are very useful for avoiding congestion and can control the sending rate in layer 3. In other words, whereas they can avoid congestion collapses in a network, they cannot control the end-to-end sending rate. This means that applications do not decide their sending rate to avoid congestion.

To observe the impact of the sending rate in ICNs, we performed simple simulations. We used 100 nodes that communicate with each other over a map size of 3,000 m $\times$ 3,000 m according to a random waypoint mobility model at a speed of 30 m/s. Simulations were executed over a time period of 10,000 s. The transmission range and bandwidth were set to 50 m and 10 messages/s, respectively. A node buffer stores 50 messages and the drop policy is drop-oldest. We used a simple epidemic protocol for routing. A simulation result is presented in Fig. 2. It is evident that the message delivery ratio decreases as the sending rate increases. It is obvious that other routing schemes for ICNs without congestion control schemes show similar behavior to that in Fig. 2.

There are many end-to-end congestion control protocols for traditional TCP/IP networks. One type of protocol controls the sending rate for a node using heuristic AIMD methods, such as TCP/Tahoe, TCP/Reno and TCP/NewReno, or equation-based methods, such as TFRC. This involves implicit feedback (ACK) or explicit feedback (ECN) for a packet for reliable delivery and control of the sending rate. The approach guarantees reliable in-order delivery and avoids congestion collapses in a network. The datagram congestion control protocol (DCCP) has some different

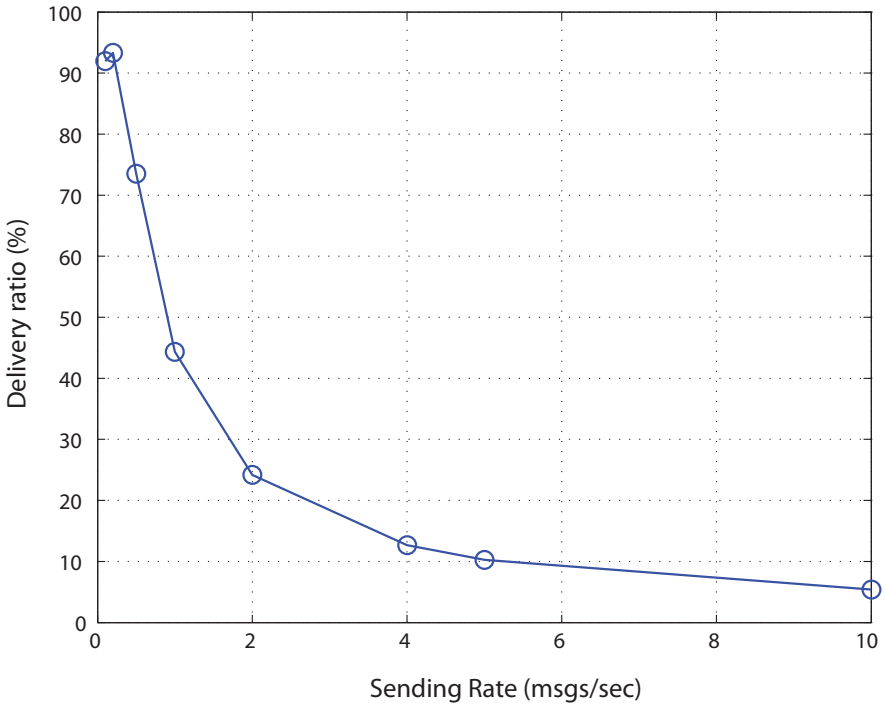

Fig. 2: Delivery ratio as a function of the sending rate.

features to above protocols, but does not guarantee reliable in-order delivery and also needs feedback to control the sending rate for a node.

For feedback to be useful, senders must receive feedback correctly and within a short time. Since there is fully connected path between two end nodes and the propagation delay is low in traditional TCP/IP networks, congestion control protocols based on feedback tend to perform well. ICNs, however, have quite different characteristics to traditional TCP/IP networks. There may not be a fully connected path between two end nodes at times and the propagation delay is significantly longer than the conventional networks, so feedback may not be received by senders or feedback received may not be helpful for controlling the sending rate. Therefore, these protocols are not suitable for ICNs.

In this paper, we propose a new end-to-end control protocol for the sending rate in an ICN in which each end node decides its sending rate according to local information. This involves active measurement of the receiving rate, one-way delay, and average number of copies. The congestion window size is also determined.

In addition to the end-to-end congestion control, there has been a lot of research effort on the application of ICN or DTN to vehicular network [11], [3], [12].

In an energy-restricted DTN such as sensor networks, the operation time of the network is determined by the energy consumption rate of the participant nodes. Therefore, reducing energy consumption for packet forwarding in

such network is an important issue. Li et al. [11] formulated the optimization problem of opportunistic forwarding with the constraint of energy consumed by the message delivery for both two-hop and epidemic forwarding. Based on the solution of the optimization problem, they proposed a few forwarding policies and evaluated them in terms of both transmission probability and energy consumption.

Niyato et al. [3] presented the optimisation formulation based on the constrained Markov decision process (CMDP) to obtain an optimal decision for the mobile router on whether to accept packets from a traffic source. Also, they proposed noncooperative game and optimisation formulations are presented for the cases when traffic sources have a self-interest to maximise their own benefit and a global interest to maximise the total benefit of the network, respectively.

Like Niyato et al., Zhu et al. [12] assumed that the participant nodes in DTN are selfish and some of them can be malicious. To address this problem, they proposed a secure multilayer credit-based incentive scheme to stimulate bundle forwarding cooperation among DTN nodes.

Since the proposed congestion control scheme in this paper can be generally applied to the existing DTN routing protocol, we believe that our scheme can be used together with the existing routing or forwarding schemes depending on the purpose and constraint conditions of the target network.

III. Problem Statement and Algorithm

The goal of the proposed algorithm is to maximize network utilization while keeping a high delivery ratio. We first define what is the problem to be solved and then propose a novel sending rate control algorithm for ICNs. To control the sending rate, each node measures the receiving rate, one-way delay and average number of copies, which are key items for detection of congestion. When the receiving rate increases, a node increases its *cwnd*; conversely, its *cwnd* is decreased when the receiving rate decreases. At an interval corresponding to the one-way delay, *cwnd* messages are sent. The rest of this section describes the approach in detail.

A. Problem Statement

In traditional network architectures, network congestion is defined as a state in which the amount of data flowed into the network is larger than the designated capacity of the network. This causes a serious decrease in performance, such as longer delays, packet losses and blocking of new connections. The same occurs in ICNs.

To define network congestion, we have to define the maximum network capacity in ICNs. Let in_n and out_n be the sending and receiving rate of each node, respectively. Note that out_n includes multiples copies of a message. Fig. 3 depicts a example of the network model dealt in this paper. Each node is represented as a small black circle, and has in_n and out_n. $In(n)$ is the sum of in_n and $Out(n)$ is the sum of out_n. When $In(n)$ increases, $Out(n)$ also increases, but $Out(n)$ decreases after network congestion occurs. Fig. 4 shows simulation results for $Out(n)$. Let In_{max} be the maximum sending rate in an ICN. When $In(n) < In_{max}$, $Out(n)$ is smaller because there are no packets to be received. When $In(n) > In_{max}$, $Out(n)$ is also smaller because packets are dropped in all nodes before arriving at their destination.

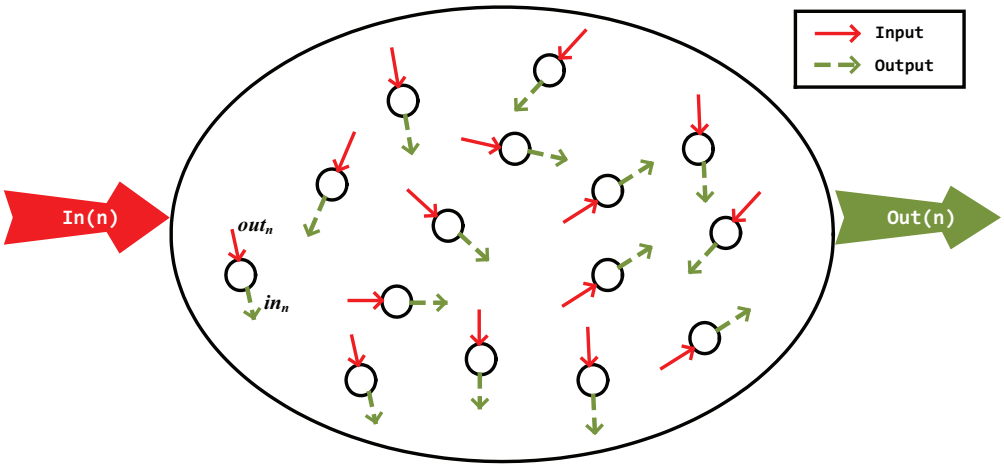

Fig. 3: Example of a network model.

To avoid network congestion and guarantee fairness among nodes, each node has to adjust its sending rate according to $\frac{In_{max}}{n}$, where n is the number of sending nodes. Since it is hard for each node to know In_{max} and n, we suggest a heuristic algorithm for adjusting the sending rate. In the next section, we explain the proposed algorithm in detail.

B. Measurement

To adjust the sending rate, each node measures three parameters: a) the receiving rate; b) the one-way delay; and c) the number of multiple copies. To avoid starvation, communicating nodes exchange measured values and recalculate their own parameters.

1) Receiving rate: We use the measured receiving rate to determine current network utilization. To measure the receiving rate, we use the time sliding window (TSW) algorithm proposed by Clark and Fang [13]. When a message is received, including multiple copies, or is sent to a destination, the receiving rate is updated. Algorithm 1 shows the TSW algorithm used in the proposed scheme.

Algorithm 1 TSW algorithm

1: On message receipt or sending to a destination:
2: $avg\_rate = \frac{(avg\_rate \times win\_len)+1}{win\_len+now-last\_arrival}$
3: $last\_arrival = now$
4:
5: win_len : a constant
6: avg_rate: estimated receiving rate for a node
7: now: current time

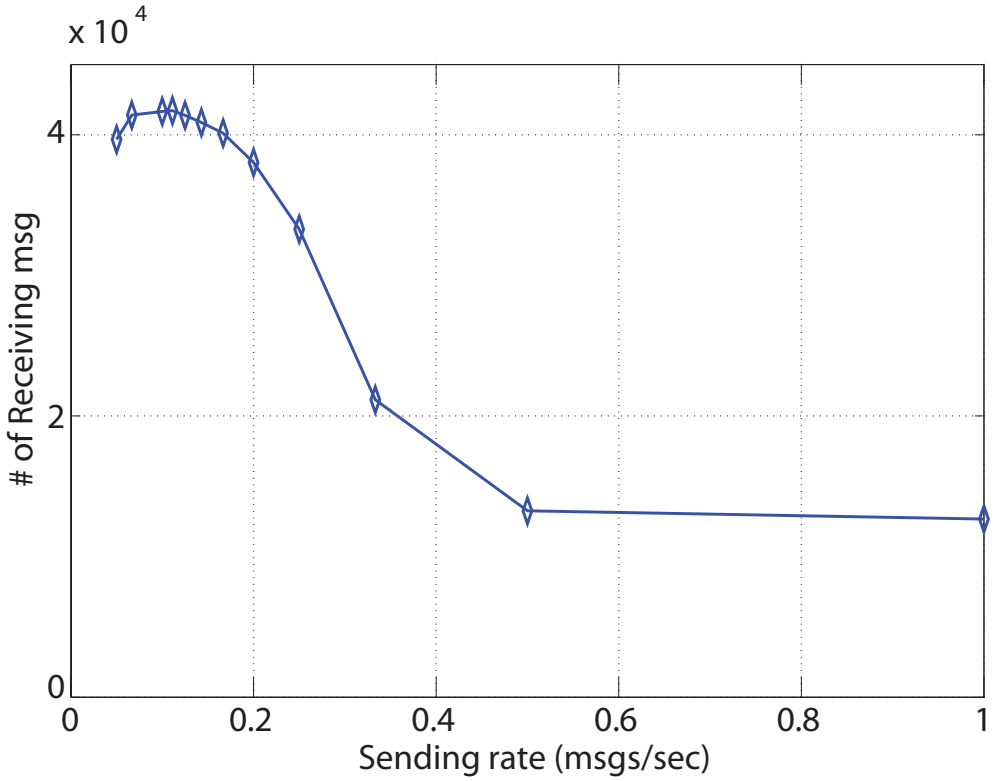

Fig. 4: Simulation results for $O(n)$.

2) One-way delay: If all nodes are synchronized with the global clock, time calculation is easy. However, time synchronization in a distributed environment is difficult. Although time synchronization in ad hoc networks has been widely studied, synchronization is more difficult in ICNs because the probability of sending packets for synchronization is too low. In another time synchronization method, every node sets its time to a global clock using GPS. This approach is also hard to deploy in ICNs because unnecessary overheads are required.

Time synchronization is not required for measurement of message delays. A message maintains two time variables, the message insertion time and the message lifetime. When a node receives a copy of a message, it writes the current time in the message insertion time field for that copy. When a node forwards a message to another node, it updates the message lifetime field for that copy. Let the message lifetime and the message insertion time for an original message be T_l and T_i, respectively, and let the current time be T_c. The message lifetime field for a copy is updated as $T_l + (T_c - T_i)$. T_c and T_i are local times in the same node, so this calculation is reasonable.

Fig. 5 presents an example showing the measurement of message times. A message is generated at node A at local time T_{a1}. At T_{a2}, node A copies this message to node B, when the local time for node B is T_{b1}. The time fields for a copy of this message are filled as T_{b1} and $T_{a2} - T_{a1}$. Time fields for the original message are not updated. At T_{b2}, node B copies a message to node D, when the local time for node D is T_{d1}. Time fields for the

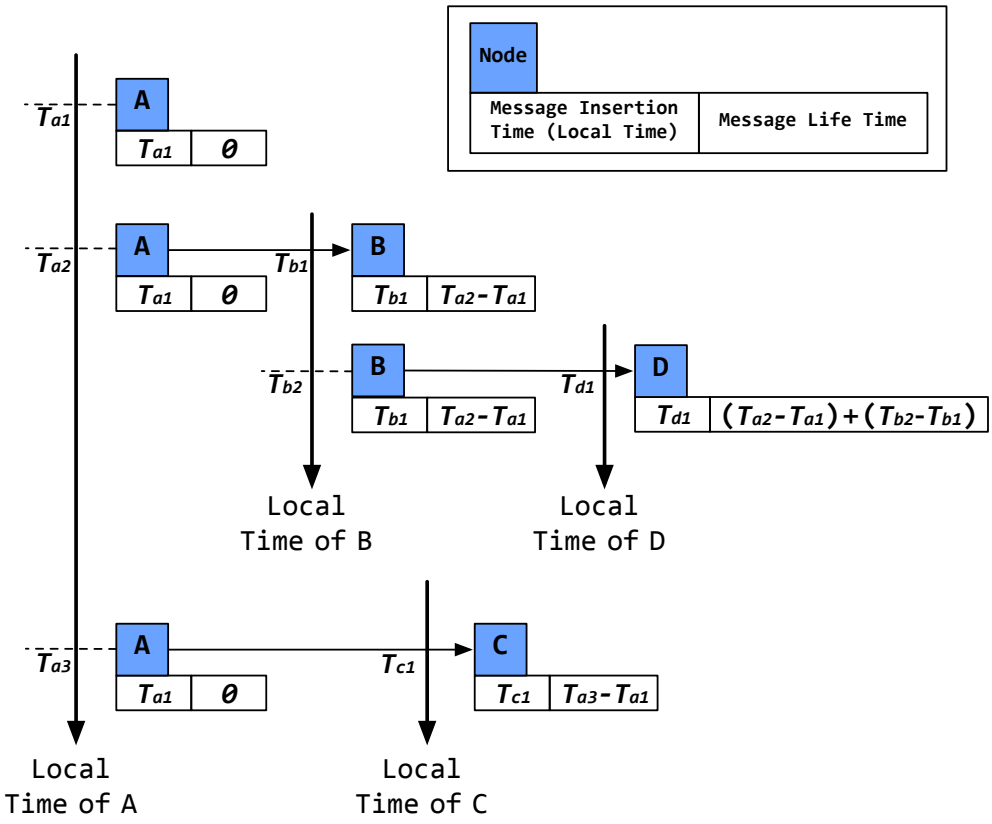

Fig. 5: Example showing the measurement of time variables.

copy in D are updated as T_{d1} and $(T_{a2} - Ta1) + (T_{b2} - T_{b1})$. Finally, node A copies a message to node C at local time T_{a3}. Time fields for the copy in C are written as T_{c1}, the local time for node C, and $T_{a3} - T_{a1}$.

By maintaining these two time fields for a message, we can easily compute the one-way delay. When a message is forwarded to its destination, the message lifetime for a copy at a destination is its one-way delay. For example, if node D is the message destination in Fig. 5, the one-way delay is $(T_{a2} - T_{a1}) + (T_{b2} - T_{b1})$.

3) Average number of multiple copies: There are many copies of a message in ICNs. When a message appears, copies of it are spread throughout the network. Thus, we measure the average number of copies.

A node maintains the number of receipts of a message (N_{copies}) and the number of messages received messages (N_{msg}). On receiving a message, a node calculates the average number of multiple copies as

$$num_{copies} = N_{copies}/N_{msg}. \tag{1}$$

C. Rate Control Scheme

AIMD is a very useful rate control algorithm when an available bandwidth is not exactly known. In terms of AIMD-based TCP congestion control, Internet can be fully and fairly utilized. In ICN environments as well,

Algorithm 2 Calculation of the new congestion window

Require: $cwnd \geq 0$

On message receipt or sending to destinations
if $avg\_rate > old\_avg\_rate$ **then**
    $cwnd+ = (1/num\_copies)$
end if
if $avg\_rate < old\_avg\_rate$ **then**
    $cwnd/ = 2$
end if

$old\_avg\_rate = avg\_rate$

an available end-to-end bandwidth is hardly estimated, and AIMD could be an effective solution for rate control problem.

In the application of AIMD, feedback from the network is essential for congestion detection. If additional control packets are used to notify congestion, an available bandwidth is wasted and the message delivery ratio can decrease. To avoid this problem, in the proposed scheme, nodes detect network congestion implicitly by monitoring receiving rate.

An increase in receiving rate indicates that the network still has an available capacity. Hence, a node increases its *cwnd*. On the contrary, when the receiving rate decreases, a node decreases its *cwnd* because the network has been already fully utilized.

1) Congestion window adaptation: As mentioned above, a node determines how much it sends using measured values. The receiving rate is used as a signal for congestion and the average number of copies is increased accordingly. Algorithm 2 is the AIMD algorithm used in the proposed scheme.

On message receipt or sending to destination, if the current receiving rate is greater than the previous one, the network may be under-utilized, so *cwnd* is increased by $1/num\_copies$ to increase the sending rate. The increment for *cwnd* is $1/num\_copies$ because the number of messages in a network increases with the average number of copies. If the increment is greater than $1/num\_copies$, the sending rate will rapidly increase and cause network congestion.

If avg_rate is less than the previous, *cwnd* is reduced by half because the network is congested. Other flows can then increase their sending rate and fairness among nodes is guaranteed. Then the previous receiving rate is set to the current receiving rate.

Theorem. *The proposed rate control scheme makes sending rate of a node converge to* $\frac{In_{max}}{N}$

Proof: First, let us think of the case that two sending nodes(A, B) exist in the network. If we let In_{max} as

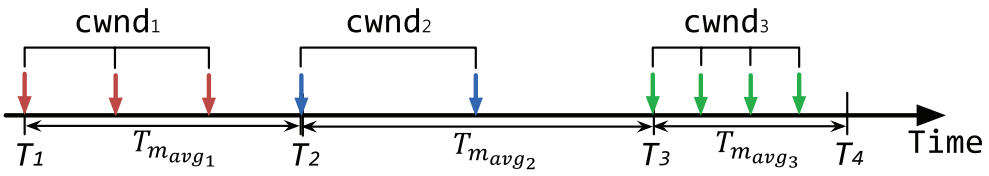

Fig. 6: An example of alternative self-clocking.

C, their sending rates increase up to (α, β) where $\alpha + \beta = C$ before packets are discarded by congestion. Here we assume that both nodes continuously increase their sending rates. Hence, they can detect network congestion simultaneously.

After detecting network congestion, both nodes decrease their sending rates by half ($\frac{m}{2}$, $\frac{n}{2}$), and again start to increase them additively. If we denote κ_1 as the amount of increased sending rate before the second congestion occurs, sending rates increases up to ($\frac{\alpha}{2} + \kappa_1$, $\frac{\beta}{2} + \kappa_1$). Since the sum of sending rates is equal to C, $\kappa_1 = \frac{\alpha+\beta}{4}$.

At the detection of the second congestion, sending rates of both nodes fall by half again ($\frac{\alpha}{4} + \frac{\kappa_1}{2}$, $\frac{\beta}{4} + \frac{\kappa_1}{2}$), and then increase additively up to ($\frac{\alpha}{4} + \frac{\kappa_1}{2} + \kappa_2$, $\frac{\beta}{4} + \frac{\kappa_1}{2} + \kappa_2$). Here, κ_2 is the amount of increased sending rate before the third congestion is observed. Since the sum of both sending rates is equal to C, $\kappa_2 = \frac{\alpha+\beta}{4}$. At the detection of the n-th congestion, sending rates of both nodes can be represented by ($\frac{\alpha}{2^{(n-1)}} + \frac{\kappa_1}{2^{(n-2)}} + \frac{\kappa_2}{2^{(n-3)}} + ... + \kappa_n$, $\frac{\beta}{2^{(n-1)}} + \frac{\kappa_1}{2^{(n-2)}} + \frac{\kappa_2}{2^{(n-3)}} + ... + \kappa_n$). Since $\kappa_i = \frac{\alpha+\beta}{4}$ for all i, sending rates are arranged by ($\frac{\alpha}{2^{(n-1)}} + \frac{\alpha+\beta}{4}(1 + \frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{(n-2)}}$, $\frac{\beta}{2^{(n-1)}} + \frac{\alpha+\beta}{4}(1 + \frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{(n-2)}})$.

As a result, n becomes large enough, sending rates of node A and B converge to $\frac{\alpha+\beta}{2} = \frac{C}{2}$. Also in the case where more than two nodes exist in the network, proof can be done in the similar way. ■

2) Alternative self-clocking: Feedback is used not only as an indication of congestion, but also to determine when a node should send messages. Since the proposed scheme does not use feedback, we propose another method to determine when a node should send messages.

Fig. 6 shows the self-clocking mechanism. As mentioned above, each node measures the one-way delay, $T_{m_{avg}}$. Since a message can be dropped in a burst if a node sends messages simultaneously at the start of a round, messages are individually sent at a certain interval. We calculate this interval as $T_{m_{avg}}/cwnd$. For example, a new round is started at T_1, cwnd is $cwnd_1$, and the measured delay is $T_{m_{avg_1}}$. Each message is sent at an interval of $\frac{T_{m_{avg_1}}}{cwnd_1}$. The next round is started at $T_2 = T_1 + T_{m_{avg_1}}$. Since the measured delay is increasing ($T_{m_{avg_2}} > T_{m_{avg_1}}$), $cwnd_2$ is smaller than $cwnd_1$. A node sends messages at an interval of $\frac{T_{m_{avg_2}}}{cwnd_2}$. The third round is started at $T_3 = T_2 + T_{m_{avg_2}}$. It sends $cwnd_3$ messages at an interval of $\frac{T_{m_{avg_3}}}{cwnd_3}$.

IV. Performance Evaluations

We validated our proposed algorithm using several simulations. For this, we developed a new event-driven network simulator for DTNs [14]. This section presents an overview of the simulator and the simulation results.

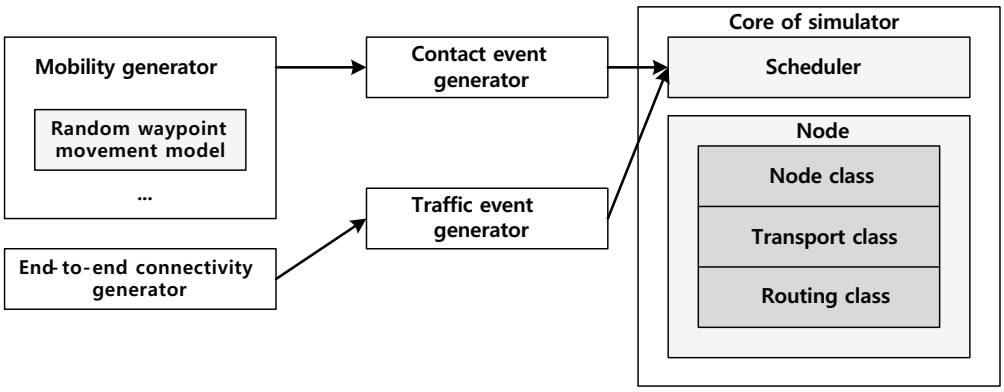

Fig. 7: Overview of the proposed simulator.

A. Simulator Overview

To verify the proposed algorithm, we developed a new network simulator for DTNs. Some researchers have developed DTN simulators, but these have some limitations and problems because they are based on a time-driven approach.

The event-driven simulator progresses time to the next event time. Therefore, all events should be generated before simulation is started. There are three event types, contact events, message creation events and transfer events. A contact event is generated when a node encounters another node. A message creation event is produced by traffic generation. A transfer event is generated when actual transmission occurs.

We developed event generation functions and a framework for the simulator and the new protocols. We also identified basic features of DTNs and the protocols. The simulator first generates all contact, message creation and transfer events to be processed during a simulation instance. The simulation scheduler processes events in order of the event execution time. The simulation states are updated at execution of every event.

The simulator consists of five main parts: a) a mobility generator; b) an end-to-end connectivity generator; c) a contact event generator; d) a traffic event generator; and e) the simulator core, which includes the event scheduler and the node agent. The mobility generator and the end-to-end connectivity generator are independent of the other parts. The simulator is shown in Figure 7.

Node movement is generated by the mobility generator via a random waypoint movement model. An end-to-end connectivity generator makes connections between the two nodes used in the traffic generator. The contact event generator creates contact events using results from the mobility generator. Message events are created by the traffic event generator. Currently, we implement CBR traffic with fixed and random end-to-end connectivity. Contact and message creation events are scheduled in the simulator core. Simulations are performed using configuration files in which traffic patterns, scenarios and other settings can be chosen.

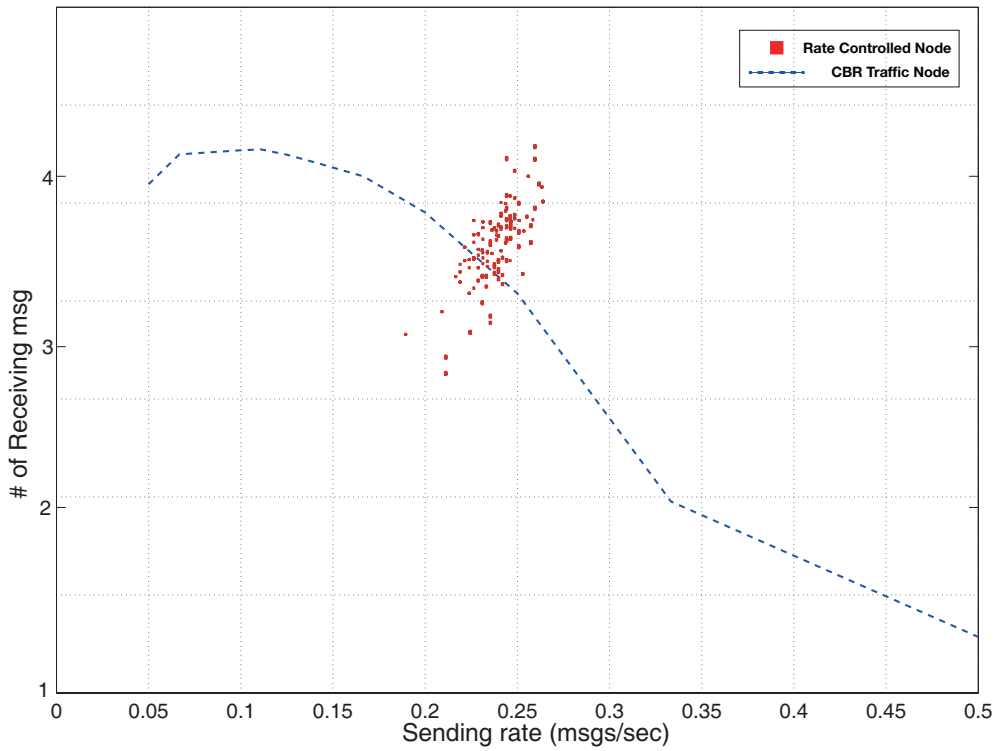

Fig. 8: Sending and receiving messages.

B. Simulation study

To observe the performance of the proposed scheme, we set the following simulation parameters: a) number of nodes, 100; b) mobility model, a random way-point model; c) map, 3000 m $\times$ 3000 m; d) node speed, 30 m/s; e) packet transmission range, 50 m; f) one-hop bandwidth, 10 messages/s; and g) buffer size, 100 messages. The buffer forwarding and drop policies are forward-youngest and drop-oldest, respectively. We use epidemic routing.

The first simulation is to verify the proposed rate control scheme under a basic topology where 100 flows exist in the network. In order to figure out the optimal throughput in a given network environment, we plot the number of received messages by changing the CBR sending rate of all nodes from 0.05 to 0.5. As being indicated by the blue dotted line in Fig. 8, the optimal throughput is achieved when the sending rate of a node is maintained around 0.11, and at a higher sending rate, the achieved throughput tends to decreases due to congestion. The red dots show the results obtained by applying the proposed rate control scheme. Without any explicit feedback from the network, the proposed scheme achieves 91% of the optimal throughput by holding the sending rate of a node at around 0.23. It is also confirmed that throughput fairness is also provided by the proposed scheme. The delivery ratio is approximately 98% and the one-way delay is approximately 135 s.

In the second simulation, we have examined the proposed scheme under a dynamic topology where 99 flows

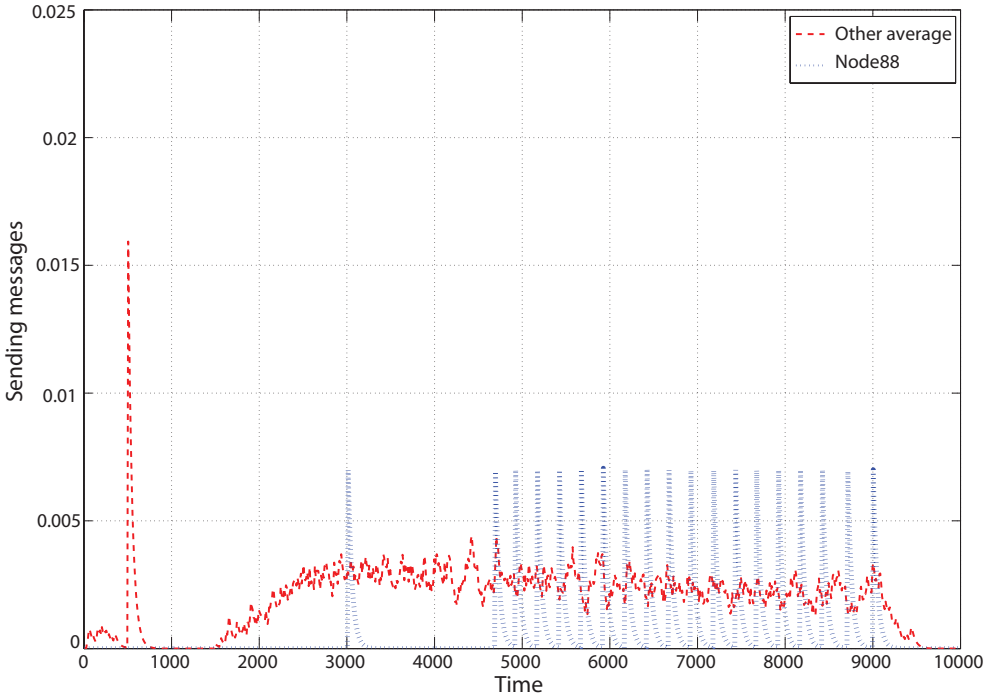

Fig. 9: Flow insertion results.

have been in the network and a single flow (node88) is newly inserted into the network. Here, we note that node88 is randomly selected among 100 nodes. Simulation result is shown in Fig. 9. The red dashed line shows the average number of messages being sent by the existing 99 flows. The blue dotted line shows the number of messages being sent by the new flow. A new flow starts at 3000s , but it does not send more messages for the next 1500s. This is because node88 cannot receive any message destined for itself, or it does not deliver any message to the final destination during that period. At around 4700s, node88 resumes to send messages and its sending rate successfully converges to the average sending rate of the other flows in the network, while the existing flows adjust their sending rates for fair access.

In the last simulation, we observe flows sending rate with a massive change in the number of flows. From 0 to 10000s, 50 flows send messages. From 10000 to 40000s, 100 flows send messages. Finally, from 40000 to 50000s, only 30 flows send messages. We conducted 10 different simulations with different mobilities. In Fig. 10, the red line shows the average number of messages being sent. The light blue, black and dark blue lines represent the optimal number of messages sent by 100, 50 and 30 flows, respectively. As shown in Fig. 10, the change of flows are successfully detected and sending rates are properly controlled by the proposed scheme. Here we note that the optimal sending rate with 100 nodes is different from that in the first simulation. This is due to the randomness in node mobility.

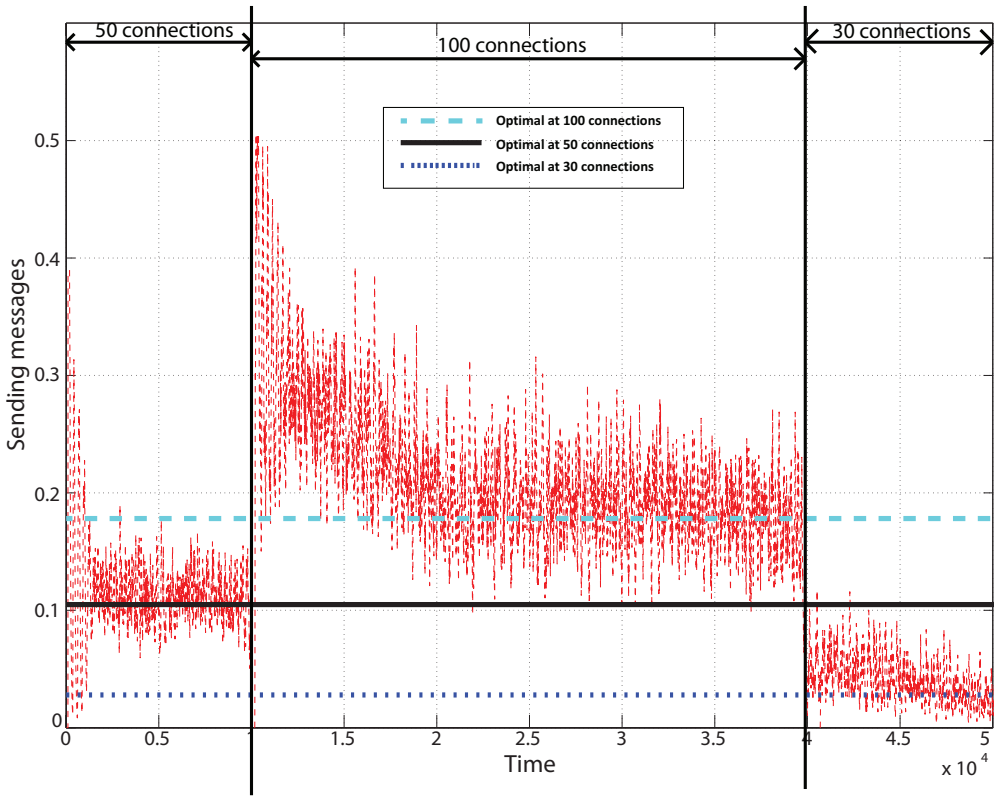

Fig. 10: Change the number of flows

V. Conclusion

We proposed a new end-to-end rate control algorithm to avoid network congestion in ICNs. Current congestion control algorithms for ICNs are implemented in layer 3 and cannot resolve network congestion when each node inputs many messages to the network. According to the proposed algorithm, each node adjusts its sending rate to maximize network utilization while keeping the delivery ratio high. It also guarantees fairness among nodes. To verify the proposed algorithm, we developed a new event-driven network simulator. Simulations demonstrated that the proposed algorithm is effective.

VI. Appendix

This research was supported by Next-Generation Information Computing Development Program (2012-0006423) and Basic Research Promotion Fund (2008-313-D00883) through the National Research Foundation of Korea (NRF) funded by the Korean government.

References

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

[2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 27–34.

[3] D. Niyato and P. Wang, "Optimization of the mobile router and traffic sources in vehicular delay-tolerant network," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5095 –5104, November 2009.

[4] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 458–472, 1999.

[5] M. Seligman, K. Fall, and P. Mundur, "Alternative custodians for congestion control in delay tolerant networks," in *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*. New York, NY, USA: ACM, 2006, pp. 229–236.

[6] S. Burleigh, E. Jennings, and J. Schoolcraft, "Autonomous congestion control in delay-tolerant networks," in *Proceedings of the AIAA 9th International Conference on Space Operations (SpaceOps)*, 2006.

[7] M. Seligman, K. Fall, and P. Mundur, "Storage routing for dtn congestion control: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 10, pp. 1183–1196, 2007.

[8] Y. Li, L. Zhao, Z. Liu, and Q. Liu, "N-drop: congestion control strategy under epidemic routing in dtn," in *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*. New York, NY, USA: ACM, 2009, pp. 457–460.

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

[10] N. Thompson, S. Nelson, M. Bakht, T. Abdelzaher, and R. H. Kravets, "Retiring replicants: Congestion control for intermittently connected networks," in *Proceedings IEEE INFOCOM*, March 2010. [Online]. Available: http://www.igh.com.tr/tr/index.php?c=page&section=gozhastaliklari

[11] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, and D. Wu, "Energy-efficient optimal opportunistic forwarding for delay-tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 9, pp. 4500 –4512, november 2010.

[12] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen, "Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 8, pp. 4628 –4639, oct. 2009.

[13] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Netw.*, vol. 6, pp. 362–373, August 1998. [Online]. Available: http://dx.doi.org/10.1109/90.720870

[14] E. Ko, H. Park, and I. Yeom, "A new event-driven network simulator for delay-tolerant networks (dtns)," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 59:1–59:6. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8650