Heuristic Congestion Control for Message Deletion in Delay Tolerant Network

Lertluck Leela-amornsin, Hiroshi Esaki

Graduate School of Information Science and Technology,
Information and Communication Engineering, The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
lertlove@hongo.wide.ad.jp,hiroshi@wide.ad.jp
http://www.hongo.wide.ad.jp

Abstract. Delay tolerant network (DTN) challenges on large delay of communication, opportunistic and intermittent connectivity so that data can be transferred over vulnerable network. Most routing protocols in DTN basing on replicating routing work well over infinite buffer assumption but the performance drops when using with finite buffer due to the congestion problem. However, it is still lack of work on congestion control in DTN. Since simple congestion control policies can cause uselessly looping problem and cannot maintain high performance when nodes buffer become overflow, a new buffer management policy for DTN should be considered. In this paper, we identify the looping problem caused by using simple buffer management policies and propose the Effective Looping Control mechanism to solve such a problem. We also propose the Credit-based Congestion Control as for heuristically deleting messages when the buffer congested to retain high delivery rate with low number of replicas.

Key words: Buffer Management, Credit Dropping, Looping Control

1 Introduction

Delay tolerant network (DTN) is a network where communications may experience large delay, and the connectivity is often opportunistic and intermittent. DTN's characteristics are different from existing Internet leading to newly challenging issues as described in [3]. Many works have been studied, used and deployed [1], [5], [6], [7], [12], [22] following the DTN architecture proposed in RFC4838 [2] by overlaying the network with bundle layer to provide delay tolerant service. To construct a high performance system, most of the works [8], [9], [10], [11], [12], [13] route messages in replicating way, and apply custody transfer [4] that each DTN host needs to take action in carrying many of messages in the non-volatile buffer [17] for a long period. These approaches can achieve high delivery rate but trade off with number of replicating messages. The performance is high under infinite buffer and infinite bandwidth assumption, however, in real situations, the performance is worse when the buffer and bandwidth are limited due to the congestion control problem.

Although modern storage devices have large capacity in scale of gigabytes, it may limit a part of memory for storing and forwarding DTN messages as described in [21]. Some applications apply DTN to personal mobile devices, which may contain any kinds of contents such as songs and videos. In cooperative aspects, people normally share their resources greedily as few as they can. Moreover, some applications may generate large size messages, while other generate smaller size but more frequently. All of these reasons can cause a diffusion of congested traffic over the system.

Regarding replicating-based routing in DTN, buffer management policies for congestion control in DTN attempts to drop insignificant messages subjecting to improving the performance essentially to reduce the number of replicas in the network while maintaining high delivery probability. Conventional policies cannot maintain high performance, and some of them can cause uselessly looping problem because of dropping some messages before they are expired. Therefore, a buffer management mechanism for controlling message deletion is required to maintain high performance for the system. We believe that the study of this issue will give a great contribution to DTNs.

In this paper, we analyze the effect of various environments to buffer occupancy of DTN nodes. After that, we study the buffer management policies in the prospect of how we should delete the messages, and classify them into two types:- the proactive policy, and the reactive policy. The proactive scheme deletes the messages depending on the time-to-live (TTL) or feedback acknowledgment as for definitely deleting. On the other hand, the reactive policy deletes some messages due to congestion in nodes' buffer. Our work scopes in the reactive policy to deal with congestion control yearning for high-performance message deletion mechanism in DTN. Along our study, we identify the looping problem, which is caused by using popular reactive policy, as a result of degradation in routing performance. After that, we propose the Effective Looping Control mechanism to prevent such a problem so that routing can achieve better performance in most environments. Moreover, we propose a heuristic congestion control policy called Credit-based Congestion Control (CCC), which pursues the low number of replication messages but still keep high delivery ratio, without using history-based information. Finally, we validate our CCC policy comparing to other policies in related works by performing extensive of the ONE simulation [19], which is the popular DTN simulation.

The rest of this paper is organized as follows. We surveyed the related work in section 2. We identify looping problem and propose Effective Looping Control to relieve such a problem in section 3. In section 4, we propose the new congestion control scheme, Credit-based Congestion Control, which can improve performance comparing to the existing policies. We verify our proposed policies in section 5 with extensive simulation. Finally, we summarize this paper and present our future work in section 6.

2 Related Works

In recent years, DTN has been researched in various aspects to counter with many newly challenging issues as described in [3]. However, the main issue in DTN is to develop the routing strategy [18] to achieve high delivery probability which is the ratio of delivered messages over the total generated messages. Since legacy-forwarding scheme achieves very low delivery rate, many researchers developed their routing protocols in replicating-based approach. Epidemic [8] is the common and simple replicating-based routing protocol. When two nodes get contact, they exchange the information of messages that they are carrying, then each node clones all of the messages another node does not have, and transfers. Although epidemic can achieve high delivery ratio, the replicas are too many resulting in consuming much bandwidth. Thus, other works attempted to reduce the number of replicas, for instance, the Probabilistic ROuting Protocol using History of Encounters and Transitivity, or PROPHET [10] reduces replicas by using knowledge based of probability that a certain message can reach the destination. Spray and Wait [9] controls the maximum number of messages that can be replicated to other nodes. In addition, other routing protocols such as RAPID [12], PEAR [11], Bubble RAP [13] are using other knowledge based metrics to achieve better performance.

Many routing algorithms work well under assumption that buffer is unlimited, but still in doubt if they do work under buffer constraint that may lead to congestion in certain environments. To prevent degrading in performance we need to concern deeply when/which/where and how we should drop the messages to maintain higher performance for any routing protocol. There are many buffer management policies had been proposed in [14], [15], [16], [23], and we can classify into two groups, 1)proactive, and 2) reactive policy.

2.1 Proactive Policy

When DTN hosts update their status (generally in a time unit, or every activating cycle), they delete messages depending on

1. Time-Based TTL [14]: Time-based TTL initiates the time-to-live (TTL) value and the value decreases a unit periodically every a certain amount of time. DTN hosts will delete the messages whose TTLs are less than zero.
2. Hop-Based TTL [14]: Like time-based TTL, original message is implemented with the maximum number of hops in two tuples, breadth and depth. Depth value decreases by one per hop, and the relay node will no further forward the message and keep carrying it for directly transmitting to the destination if the depth value becomes zero. Each node can replicate the message only b copies no more than breadth value, and deletes the message that has been forwarded for b times.
3. Feedback Policy: The destination node broadcasts an acknowledge message as a feedback policy, then, each relay node drops the carrying message corresponding to the acknowledge message.

2.2 Reactive Policy

DTN hosts delete the message when their buffers are congested. Many reactive dropping schemes have been proposed in [15], [16], and [23].

1. Last-In First-Drop (LIFD): is a simple dropping scheme but very low performance. A node just rejects any new coming message, if its buffer overflows.
2. First-In First-Drop (FIFD): drops the message stored in the buffer first.
3. Oldest-Drop or Least Remaining Life First-Drop: concerns about the amount of remaining TTL to decide which message should be dropped. For Oldest-drop, the message that has the least remaining TTL will be dropped first.
4. Youngest-Drop or Most Remaining Life First-Drop: on the other hand, Youngest-Drop drops the message that has the most remaining TTL.
5. Most-Forwarded First-Drop (MOFD): the message that has been forwarded the largest number of times is the first to be dropped, thus giving messages forwarded less times produce more chances of getting forward.
6. Least-Forwarded First-Drop (LEFD): the message that has low delivery probability and has not been forwarded for the fewest number of times is the first to be dropped. This is because it has a little chance to reach the destination.

In addition, in [16] also proposed the history-based policy, which requires history information to estimate an optimum utility for two popular metrics: - maximizing the average delivery rate, and minimizing the average delivery delay. However, it needs the beacon-updating message to learn the status of the network, and needs much time for computation. Besides that, in [23] proposed the mechanism to migrate the custody from congested node to other nodes where still have some vacant rooms. However, this algorithm cannot work well if the nodes congest at the same time, and if there is no encounter node at the time such a node congests.

3 Effective Looping Control Mechanism

To delete messages in an effective way, we usually implement proactive policy like global time-to-live as a final control to guarantee that the messages will be deleted from the system eventually. In parallel, we use the reactive policy to control message deletion when the DTN buffer becomes congested. However, all of reactive policies (except LIFD) generally drop the carrying messages before they are expired, as a result that the same messages in other nodes can be looped back to the DTN node that has just deleted such messages as illustrated in figure 1. It may meet greater impact, if the DTN router inherits the simple-based routing like Epidemic [8]. And even the DTN router uses knowledge-based routing such as PROPHET [10], PEAR [11] and others, which may have less effect, but looping problem may still cause heavy damage in some situations where certain pair nodes may uselessly loop some messages back-and-forth repeatedly. Although

global TTL can eventually delete the message, there is no mechanism to estimate TTL accurately for DTN's application whose TTL is not fixed. Therefore, we generally implement a large TTL value in term of days, weeks or longer resulting in wasting amount of bandwidth if such useless looping occurs many times and repeatedly. Hence, in this section, we propose the Effective Looping Control, as shown in figure 2, to relieve the degradation in performance because of looping problem.

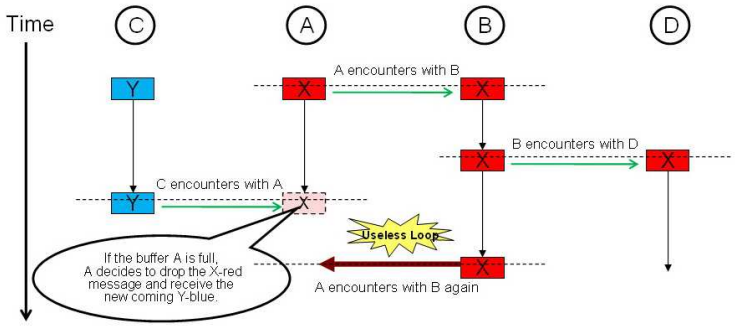

Fig. 1. Looping problem due to congestion control policy leads to wasting of bandwidth.

The key feature of this method is that, instead of keeping track of every whole message, we save space by caching only header of the deleted message in the entry named *junk entry*. And we denote clock down *time in junk*, as for, each caching entry can stay in the junk within a time limit. A node that is receiving any new message will check the junk entry list and ignore the message, if it is listed in the junk. However, after the time in junk period, the node allows the message to come into its buffer again. This soft policy is to relax in some incidents that a message that has a better chance to reach the destination (depending on routing scheme) should have a permission to be looped and replicated again. The Effective Looping Control mechanism can be described in the following steps and illustrated in figure 2.

1) Once any node has deleted the messages, it tracks the header of those messages that can distinguish the messages (usually it is constructed by sequence number of message and the node id the message originally generated from).

2) Each entry of deleted message will be listed in the junk until the time in junk expired.

3) If a message loops back to any node that still has the entry of such a message in the junk, that node will ignore and not request that message.

Although we can eventually delete the message via TTL, in DTN we usually set TTL to very large value because no one knows the appropriate value of it. This Effective Looping Control mechanism not only prevents the useless looping, but also gives an option for the network to adjust whether it will allow the looping

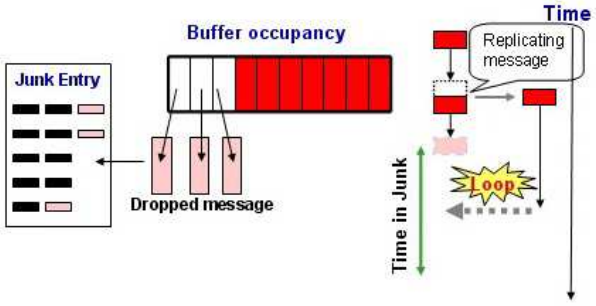

Fig. 2. The Effective Looping Control.

again or not by setting the time in junk parameter. For instance, we can just set the time in junk greater than the TTL of the message or infinite value, if we desire loop-free network.

4 Credit-based Congestion Control (CCC) for Message Deletion

As mentioned in section 2, the simple LIFD cannot achieve high performance, so [15] has presented other congestion control policies to obtain better performance, essentially to reduce the number of replicas. Although the history-based policy in [16] has been studied, it focused on minimizing the delay metric, and needs exchanging in updating message to estimate many parameters for predicting the optimum value using for dropping the message. Another work in [23] attempted to migrate the custody message from congested node to other nodes, but it is still in doubt that if there is no other node at that time or if all nodes are congested, the migration does not work leading dropping mechanism to come out and take into account. Therefore, in this section, we propose the newly heuristic congestion control policy called Credit-based Congestion Control (CCC) to maintain high delivery probability and reduce the number of replicas as our target. CCC is dynamic, independent, and isolated from history information, so there is no need of beacon-updating.

CCC bases on the concepts that the message that is obsolete should be deleted first as in oldest-drop, which performs high performance by using the time-dependent credit. However, we also added the merit of most-forwarded first-drop that the message has been forwarded many times should be dropped first by applying the refilling and refunding technique when pair nodes get contact and exchange their messages. Besides that, different applications or even in one application may have different priorities for different messages, which need to be standardized and is still an open issue. No matter how, getting along priority

aspect we can easily rule the priority by adjusting the initial value of credit as high and low for higher and lower priority, respectively. The DTN router along these credit-based policies will drop the message that has the least credit (can be negative) when the buffer becomes full to allocate vacant space for the new coming message. CCC can be described as follows.

Once a message is generated from any DTN node, it will carry maximum amount of credit as its initial value. In consequence of that, host generating the message can set how much credit should be initiated for different priorities depending on various applications. Later, one credit will be decreased at every time unit similar to age decreasing. However, the key technique of this policy is refilling and refunding amount of credit when pair nodes encounter at any point of transmission. We can construct function for refilling and refunding methods in many approaches, however, in this paper, we propose an additive approach called Simple CCC and Half-Twice CCC as a multiplicative one.

4.1 Simple Credit-based Congestion Control (S-CCC)

Simple Credit-based Congestion Control policy manages amount of refill and refund as follows. When two nodes get contact, they exchange the messages depending on applied routing protocol. After that, each node refunds the sent message's credit for amount of value named *penalty* at the sender side, and refills the received message's (the same id, but the replicated one) credit for amount of value named *reward* at the receiver. As a result, such a message that has been replicated will have lower credit in sender host, but higher credit in receiver one. We can easily formulate it into equation 1,

$$C_{sender}(t+1) = max\left(C_{sender}(t) - "penalty"\ , minimum\_value\right) \tag{1a}$$

$$C_{receiver}(t+1) = min\left(C_{sender}(t) + "reward"\ , initial\_value\right) \tag{1b}$$

where C_{sender} is the credit of message in sender side, $C_{receiver}$ is the credit of replicated message in receiver side, before (time=t) and after (time=t+1) pair nodes encounter. However, the credit after refill cannot exceed the initially maximum value, and in practical, the credit should eventually equal to a certain minimum value depending on the number of bits using as a header, or manually configure.

4.2 Half-Twice Credit-based Congestion Control (HT-CCC)

For S-CCC, we proposed the additive approach; however, HT-CCC on the other hand applies multiplicative function to adjust refunding and refilling credit. With this policy, it defines the multiplicative scheme following equation 2. The functions in equation 2 are constructed similarly from the sense of nature so-called half-life theory. However, the complexity of equation 2 is because the credit value

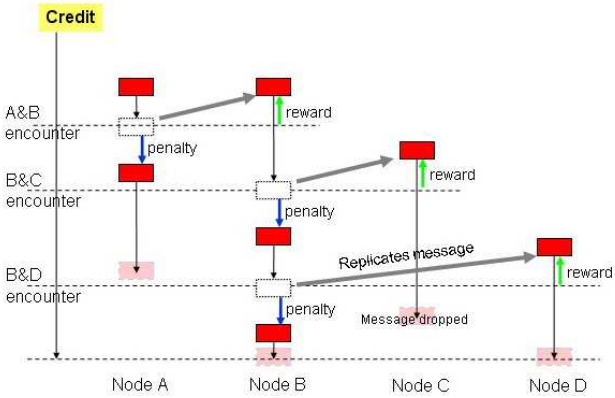

Fig. 3. The Simple Credit-based Congestion Control (S-CCC)

can be negative. Allowing credit to be negative can slow down the rate of refilling and refunding when the credit of the message is near zero and decrease rapidly when the credit is negative, which means that this message has no credit and is going to be in debt so we should ignore it. Whenever pair nodes get contact and completely replicate the message, the credit of the message at the sender is refunded as a half of credit before replicating but no less than a certain value, which is limited by the number of bits we use as a header or we may manually configure. In contrast, the credit of the message at the receiver becomes twice but no more than maximum value, which equals to the initial credit. Both policies can maintain high delivery probability, at the same time, they can decrease the replica ratio as we validate in the next section.

$$C_{sender}(t{+}1) = \begin{cases} max\left(\lfloor C_{sender}(t) \div 2 \rfloor, minimum\_value\right) & ; \text{ if } C_{sender}(t) > 0 \\ max\left(C_{sender}(t) \times 2, minimum\_value\right) & ; \text{ if } C_{sender}(t) < 0 \end{cases} \tag{2a}$$

$$C_{receiver}(t+1) = \begin{cases} min\left(C_{sender}(t) \times 2, initial\_value\right) & ; \text{ if } C_{sender}(t) > 0 \\ min\left(\lfloor C_{sender}(t) \div 2 \rfloor, initial\_value\right) & ; \text{ if } C_{sender}(t) < 0 \end{cases} \tag{2b}$$

5 Simulation and Evaluation

In this paper, we use the ONE simulator [19], NetLab, Helsinki University, which is the simulator for DTN supporting GUI and providing many of libraries for movement models, routing protocols, but no buffer management policies. Hence, we implement relating buffer management policies, our Effective Looping Control and including our Credit-based Congestion Control into the simulator. Later, we

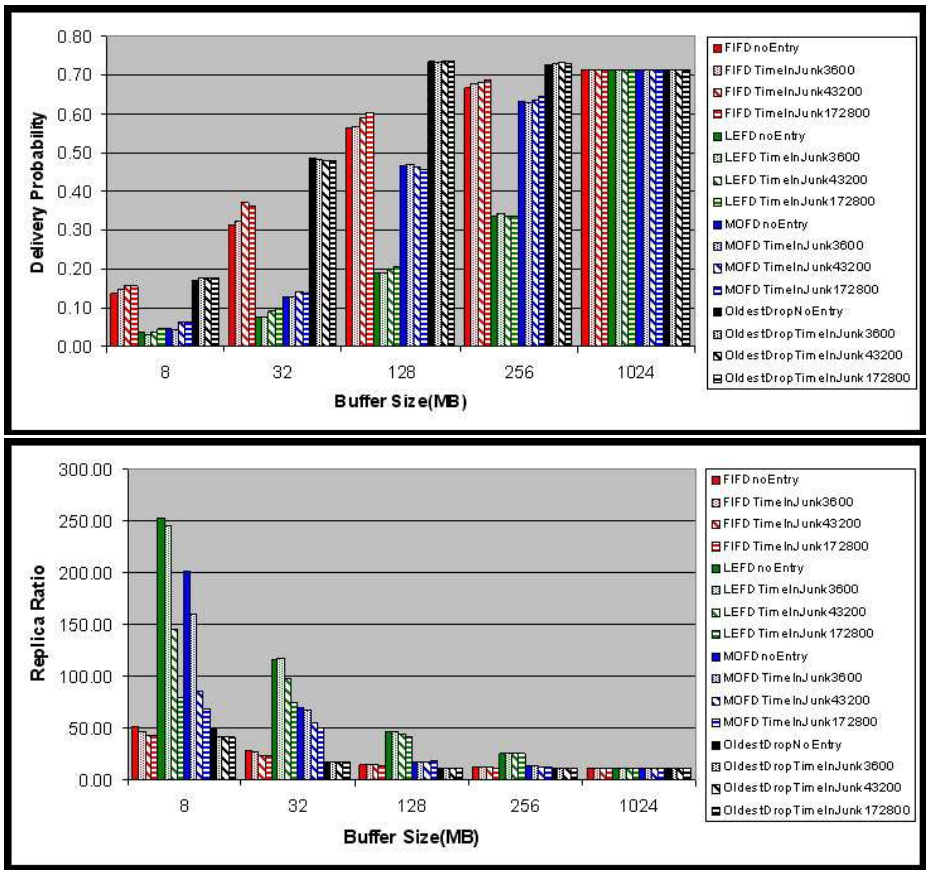

Fig. 4. Performance of Effective Looping Control.

use the random waypoint movement model for 100 nodes over 1 km x 1 km world size, for 2 days simulation time. Each node has 10-meter transmitting range, 2-Mbps bandwidth, and moves with speed uniformly random from 0 km/h to 5 km/h. Besides that, we create message by using MessageEventGenerator class attached in the ONE simulator to randomly generate the source and destination nodes for a 500 kB to 1 MB message every 1 to 30 seconds interval, uniformly. Since we are focusing on the reactive message deletion control, thus we assume the TTL value is equal to the observation time, 172800 seconds (2 days).

5.1 Evaluation in Effective Looping Control Mechanism

In this section, we compare most standing out policies to those policies themselves but applying our heuristic mechanism, effective looping control. We also compare the difference of time in junk parameters ranging from 3600 seconds (1 hour), 43200 seconds (12 hours) to 172800 seconds (2 days). Then, we show

the relationship among the delivery probability, replica ratio and the buffer size ranging from 8MB, 32MB, 128MB, 256MB to 1024MB, as shown in figure 4. From figure 4, we can see that when using Effective Looping Control to Epidemic routing we could increase the delivery rate of FIFD for any buffer size, even we changed the time in junk parameter. In addition, other policies also obtained higher delivery probability in most cases, but not much different in oldest-drop.

However, the replica ratio when applying effective looping policy is reduced in all cases, compared to the cases without this mechanism. Especially, when the traffic is very congested (many messages are generated or small buffer size of a node), our looping control could help reducing replicas up to 70% for LEFD, 65% for MOFD, around 15% for FIFD, and also, 15% for oldest-drop when buffer is very congested. Although the delivery ratio drops in some environments, it is less than 2% comparing to the replica ratio we can reduce. Corresponding to our assumption, we could maintain high delivery probability and reduce the number of replicas for most of all reactive policies. However, the performance is the same for all when the buffer is large enough (1024MB). This is because the network is not congested, so there is no message loss in the network. Remark that, the delivery probability is not equal to one because there are still some messages just generated and may reach the destination out of range of our observation time. In this sense, the Effective Looping Control can relieve such kind of problem by adjusting proper time in junk parameter to that system.

5.2 Evaluation in Credit-based Congestion Control

For evaluating the CCC, we compare among simple congestion control policies and our CCC policies with different initial credit values 3600 credits (HTCredit3600), 43200 credits (HTCredit43200), and 172800 credits (HTCredit172800) for Half- Twice CCC, and at 172800 credits for simple version fixing 10 credits for both penalty and reward. While, the minimum credit value in this simulation is set to the minimum value of integer (for Java programming $= -2^{31}$). We show the performance results in table 1 for different buffer sizes 8 MB, 32 MB, and 256 MB, which easily congest, through Epidemic [8] and Prophet [10] routing.

Table 1. Comparison of Replica Ratio among Reactive Buffer Management Policies.

	Epidemic Routing						Prophet Routing					
	Delivery Probability			Replica Ratio			Delivery Probability			Replica Ratio		
Buffer Size (MB)	8	32	256	8	32	256	8	32	256	8	32	256
FIFD	0.1362	0.3111	0.6664	52.4140	28.4857	12.6230	0.1667	0.3718	0.7033	34.9224	22.1443	11.6573
MOFD	0.0444	0.1288	0.6339	201.6181	70.0129	13.3065	0.0676	0.1647	0.6441	117.1928	54.3162	13.1769
OldestDrop	0.1712	0.4848	0.7249	51.5952	17.7563	11.5215	0.1797	0.5056	0.7301	44.9874	16.7499	11.2092
SimpleCredit172800	0.1777	0.4833	0.7334	41.6650	17.6398	11.3637	0.1834	0.5051	0.7422	31.3373	15.9291	10.9750
HTCredit3600	0.1788	0.4874	0.7313	41.3886	17.4339	11.4217	0.1842	0.5079	0.7413	31.3349	15.8285	10.9874
HTCredit43200	0.1814	0.5335	0.7322	43.5313	15.8513	11.3993	0.1854	0.5449	0.7446	33.7201	14.7224	10.9486
HTCredit172800	0.1810	0.5200	0.7331	43.6944	16.2970	11.3530	0.1847	0.5226	0.7439	33.9030	15.5421	10.9617

We evaluated our algorithm and obtained the result as shown in table 1. Both S-CCC and HT-CCC could maintain high delivery probability, while reduce the number of replicas up to 20% for Epidemic routing, and to 10% for Prophet routing, respectively to the best existing policy of each case. Even if we need to add some information into header of the message to provide the current credit field and initial credit value, it is very small comparing to whole message in DTN whose size can be up to 1 MB. Therefore, it is worth to trade off some field in header to many large replication messages those we can reduce. Although the performance of CCC is close to the oldest-drop policy since its credit behaves like an age of the message, it makes use of the advantage of refilling credit to enhance a chance for replicated message in the next-hop node, and refunding credit to devalue the message at present node, as a result that it can reduce useless replicas but still keep high delivery rate not only in Epidemic routing but also in Prophet routing. In addition, in the Half-Twice approach also gave the same tendency. This means that we can control the different priorities of messages corresponding to the initial credit values, and get high performance, concurrently.

6 Conclusion and Future work

In this paper, we investigated the problem of congestion control in DTN. Since, some reactive buffer management policies have mined the looping problem, we proposed the Effective Looping Control to prevent uselessly looping. We also propose the congestion control policies in the heuristic way. Hence, our parameters used in our work need to be studied and standardized before using in the real system for any general and/or any particular environment following the heuristic process. However, this work presents an intuitive idea but reasonable as a prototype to control message deletion when buffer congests so that we can achieve high performance via reducing overhead without decreasing in delivery rate. Therefore, in the future, we will apply this credit-based policy by refunding and refilling with respect to other resource values and/or knowledge metrics such as remaining buffer, number of forwarded, etc., based on game theory and will further expand to real world implementation.

References

1. S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, and et. al, Delay-tolerant networking: an approach to interplanetary internet, IEEE Communications Magazine, pages 128-136, 2003.
2. RFC4838 : Delay-Tolerant Networking Architecture, IRTF DTN Research Group, 2007.
3. K. Fall, A delay-tolerant network architecture for challenged internets, Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pages 27-34, 2003.

4. K. Fall, W. Hong, and S. Madden, Custody transfer for reliable delivery in delay tolerant networks, Technical Report IRB-TR-03-030,Intel Research, Berkeley, California, 2003.
5. P. Juang, H. Oki, and et. al, Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet, ACM SIGOPS Operating Systems Review, pages 96-107, 2002.
6. T. Small, Z.J. Haas, and et. al, A sensor network for biological data acquisition, Sensor Networks.
7. S. Carrilho and H. Esaki, A Pub/Sub message distribution architecture for disruption tolerant networks, IEICE transactions on Information and Systems, page 1888-1896, 2009.
8. A. Vahdat and D. Becker, Epidemic routing for partially connected ad hoc networks, 2000.
9. T. Spyropoulos, K. Psounis, and C.S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, ACM SIGCOMM workshop on Delay-tolerant Networking, page 259, 2005.
10. A. Lindgren, A. Doria, and et. al, Probabilistic routing in intermittently connected networks, Computer Communication Review volume 7, pages 19-20, 2003.
11. H. Ochiai and H. Esaki, Mobility entropy and message routing in community-structured delay tolerant networks, The 4th Asian Conference on Internet Engineering, pages 93-102, 2004.
12. A. Balasubramanian, B. Levine, and A. Venkataramani, DTN routing as a resource allocation problem, Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, page 384, 2007.
13. P. Hui, J. Crowcroft, and E. Yoneki, Bubble rap: Social-based forwarding in delay tolerant networks, The 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 241-250, 2008.
14. W.H. Yuen and H. Schulzrinne, Performance evaluation of time-based and hop-based TTL schemes in partially connected ad hoc networks, IEEE ICC'06, 2006.
15. A. Lindgren and K.S. Phanse, Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks, IEEE COMSWARE, 2006.
16. A. Krifa, C. Barakat, and T. Spyropoulos, Optimal buffer management policies for delay tolerant networks, IEEE SECON, 2008.
17. M. Seligman, Storage usage of custody transfer in delay tolerant networks with intermittent connectivity, ICWN, 2006.
18. E.P. Jones and P.A. Ward, Routing strategies for delay-tolerant networks, ACM Computer Communication Review (CCR), 2006.
19. A. Keranen, J. Ott, and T. Karkkainen, The ONE simulator for DTN protocol evaluation, The 2nd International Conference on Simulation Tools and Techniques, page 55, 2009.
20. S. Jain, K. Fall, and R. Patra, Routing in a Delay Tolerant Network Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pages 145-158, 2004.
21. M.J. Pitkanen and J. Ott, Enabling opportunistic storage for mobile DTNs Pervasive and Mobile Computing, pages 579-594, 2008.
22. S.B. Eisenman, N.D. Lane, and et. al, Metrosense project: People-centric sensing at scale, First Workshop on World-Sensor-Web, 2006.
23. M. Seligman, K. Fall, and P. Mundur, Alternative custodians for congestion control in delay tolerant networks, Proceedings of the 2006 SIGCOMM workshop on Challenged networks, page 236, 2006.