*Advanced Engineering Forum* *Online: 2012-06-04*
*ISSN: 2234-991X, Vol. 4, pp 13-18*
*doi:10.4028/www.scientific.net/AEF.4.13*

# Advanced Optimal Buffer Scheduling Policy in Opportunistic Networks

Qilie Liu[1,2,a], Guangde Li[1,b], Yun Li[1,c], Yingjun Pan[2], Fengzhi Yu[1]

[1]Key lab of Mobile Communication technology, Chongqing University of Post and Telecommunications, Chongqing 400065, China

[2] The key lab of OptoElectronics Technology and System, Ministry of Education, Chongqing University, Chongqing 400044,China

[a]liuql@cqupt.edu.cn, [b]liguangdecqupt@126.com,[c]liyun@cqupt.edu.cn

**Keywords:** Opportunistic Networks, buffer management, optimal theory, scheduling policy

**Abstract.** Opportunistic Networks (*ON*s) are the newly emerging type of Delay Tolerant Network (*DTN*) systems that opportunistically exploit unpredicted contacts among nodes to share information. As with all DTN environments *ON*s experience frequent and large delays, and an end-to-end path may only exist for a brief and unpredictable time. In this paper, we employ optimal theory to propose a novel buffer management strategy named Optimal Buffer Scheduling Policy (OBSP) to optimize the sequence of message forwarding and message discarding. In OBSP, global optimization considering delivery ratio, transmission delay, and overhead is adopted to improve the overall performance of routing algorithms. The simulation results show that the OBSP is much better than the existing ones.

## Introduction

Delay Tolerant Networks (DTNs) [1], based on the store-carry-and-forward routing principle [2], are with intermittent connectivity, high latency and without end-to-end path, which makes routing in DTNs a challenge problem. Opportunistic Networks (*ONs*) are a newly emerging type of DTNs systems that can opportunistically exploit unplanned contacts between nodes to share information. In general, there are many recent proposed routing algorithms [3-6] to address the problem. In the case of limited buffer resources, the node carrying a copy of each message will be quickly saturated its cache, and can not continue to store copies of incoming messages. Therefore, the design of efficient cache management strategy to improve opportunities for the overall performance of the network is particularly important.

In this paper, we study the problem of buffer management in ONs. Firstly, we propound an optimal buffer forwarding and discarding strategy considering the delivery ratio, the delivery delay, and overhead. Then, we offer a method based on creating the lists maintained by each node to calculate the number of copies of message *i* at time *t*, $n_i(t)$. Finally, we have implemented many simulations and compared it to some other buffer policies in simulator ONE [7]. The simulation results show that our policy outperforms existing policies.

The rest of this paper is organized as follows. Section II goes over some related work. In section III, we discuss the core idea of buffer optimization. The section IV displays simulation results and compares it with other buffer strategies. Finally, section V concludes the paper.

## Related work

In order to improve network performance, such as to improve message delivery success rate, and reduce the transmission delay, the researchers propose a variety of cache management strategy.

In paper [8], the authors deduce the optimal function for independent metrics including delivery ratio and delay in the case of large enough bandwidth. However it does not considers overhead cost and the bandwidth is often limited in reality. Developing and studying prioritization schemes for messages such that nodes can forward high priority messages and drop low priority messages is proposed in [9].However, this policy is based on the history information of nodes which in reality we can not get easily. In this case, we cannot guarantee that the policy in paper [9] is really useful in real application. K.Shin and S.Kim [10] proposed two utility functions in order to maximize the message delivery success rate and minimize the average message delay based on message properties, such as the number of copies of the message, the message of time to live (TTL) and the remaining TTL. In addition, [10] proposed a method of calculating the number of copies of the message. Since obtained copies of the message are inconsistent after two nodes meet to exchange information, the use of this method is restricted.

In this paper, we employ optimal theory to optimize buffer scheduling strategy including forwarding and discarding policy. And we deduce the optimal functions considering the delivery ratio, the average transmission delay, and the average overhead cost, respectively.

**Buffer optimization Policy**

In this section, we first study the problem of choosing which message to send first and choosing which message to discard first when a node's buffer is full. We deduce the optimal functions for independent performance metrics respectively, such as average delivery ratio, average delivery delay, and average overhead cost. To formulate the optimal buffer scheduling policy (OBSP), we only assume that the meeting time between nodes adheres to exponential distribution. In Table 1 we list the main parameters that we will use in this paper.

Table 1 Parameters in Optimal Function

| | |
|---|---|
| $\beta$ | the rate of messages dropped by the nodes |
| $Ti$ | elapsed time form message *i* |
| $ni(t)$ | number of copies of message *i* at time *t* |
| $ki(Ti)$ | number of nodes (excluding source) that have seen message *i* since its creation until elapsed time *Ti* |
| $SR$ | successful rate of delivery after nodes meet each other |
| $dri$ | delivery ratio of message *i* |
| $\overline{dr}$ | average delivery ratio |
| $Nm$(t) | number of distinct messages in the network at time *t* |
| $Nn$ | number of nodes in the networks |
| $Ri$ | remaining time to live for message *i* |
| $di$ | delay of successfully transmitting message *i* |
| $\overline{d}$ | average delay of successfully transmitting all generated messages |
| $oci$ | overhead cost which means the number of copies when the message *i* reaches the destination |
| $\overline{oc}$ | average overhead cost |

**Buffer strategy for Independent Metrics.** Because of the limited transmission speed, and short constrained connected time, messages may not be delivered successfully when connections are established. Thus, we define the probability of successful message transmitted after nodes meet each other as SR in this paper.

**Maximizing the average delivery ratio.** We assume that a number of messages, which all have the finite TTL value, are propagated in the network using replication. The source of the message keeps a copy of it during the whole TTL duration, while intermediate nodes are not obliged to do so.

Number of nodes (excluding source), *ki(Ti)*, that have seen message *i* since its creation until elapsed time *Ti*, includes the messages which has dropped by some nodes, so we have the following equation:

$$k_i(T_i) = \frac{n_i(t)}{(1-\beta)} \tag{1}$$

Then we get the function for metric average delivery ratio.

The probability that message *i* has been already delivered is equal to:

$$P\{message \text{ i already delivered}\} = \frac{k_i(T_i)}{N_n - 1} \tag{2}$$

The delivery ratio of message *i* is:

$$\begin{aligned} dr_i &= \frac{k_i(T_i)}{N_n-1} + (1-\frac{k_i(T_i)}{N_n-1})(1-e^{-\alpha \cdot n_i(t) \cdot R_i})SR \\ &= \frac{n_i(t)}{(1-\beta)(N_n-1)} + (1-\frac{n_i(t)}{(1-\beta)(N_n-1)})(1-e^{-\alpha \cdot n_i(t) \cdot R_i})SR \end{aligned} \tag{3}$$

The average delivery ratio of all generated messages is:

$$\overline{dr} = (\sum_{i=1}^{N_m(t)} dr_i)/N_m(t) \tag{4}$$

We differentiate $\overline{dr}$ with respect to *ni(t)*, then we get Eq. 5.

$$\begin{aligned} \Delta\overline{dr} &= [\sum_{i=1}^{N_m(t)} (\frac{\partial dr_i}{\partial n_i(t)} \cdot \Delta n_i(t))] / N_m(t) \\ &= \{\sum_{i=1}^{N_m(t)} [\frac{1-SR}{(1-\beta)(N_n-1)} + \frac{SR}{(1-\beta)(N_n-1)} e^{-\alpha n_i(t) R_i} + \alpha R_i SR(1-\frac{n_i(t)}{(1-\beta)(N_n-1)})e^{-\alpha n_i(t) R_i}]\}/N_m(t) \end{aligned} \tag{5}$$

When $\Delta n_i(t)$= 1, we keep and store the new-coming message *i*; when $\Delta n_i(t)$ = 0, we don't drop a message from buffer; when $\Delta n_i(t)$= -1, we drop an existing message from buffer.

In order to maximize $\overline{dr}$, we should get the largest $\Delta\overline{dr}$. Eq.5 is decreasing function about *ni(t)*, so it is equal to send message with the smallest $n_i(t)$ first, and drop the message with the largest $n_i(t)$ first when the buffer is full until the free space of the buffer is available for the new coming message.

**Minimizing the average delivery delay.** By a similar way, we get the optimal function for metric average delivery delay.

The delivery delay of message *i* is:

$$\begin{aligned} d_i &= \frac{k_i(T_i)}{N_n-1}C + (1-\frac{k_i(T_i)}{N_n-1})E[t \mid t > T_i]) * SR + \frac{k_i(T_i)}{N_n-1}(1-SR)E[t \mid t > T_i] * SR \\ &= \frac{Cn_i(t)}{(1-\beta)(N_n-1)} + (1-\frac{n_i(t)}{(1-\beta)(N_n-1)})(\mathrm{T}_i + \frac{1}{\alpha n_i(t)}))SR + \frac{n_i(t)}{(1-\beta)(N_n-1)}(1-SR)(\mathrm{T}_i + \frac{1}{\alpha n_i(t)})SR \end{aligned} \tag{6}$$

The average delivery delay is:

$$\overline{d} = (\sum_{i=1}^{N_m(t)} d_i) / N_m(t) \tag{7}$$

We differentiate $\overline{d}$ with respect to *ni(t)*, then we get Eq. 8.

$$\Delta\overline{d} = \sum_{i=1}^{N_m(t)} (\frac{\partial d_i}{\partial n_i(t)} * \Delta n_i(t)) = \{\sum_{i=1}^{N_m(t)} [(\frac{C - SR^2 \cdot T_i}{(1-\beta)(N_n-1)} - \frac{SR}{n_i(t)^2 \cdot \alpha})\Delta n_i(t)]\} / N_m(t) \tag{8}$$

In order to minimize $\overline{d}$, we should minimize $\Delta\overline{d}$. Eq. 8 is increasing function about $n_i(t)$, so it is equal to send message with smallest $n_i(t)$ first, and drop the message with largest $n_i(t)$ first when the buffer is full.

**Minimizing the average overhead cost.** Using a similar method, we get the optimal function for metric overhead as well.

The overhead cost of message *i* is:

$$oc_i = n_i(t) + \frac{R_i}{\alpha} * \frac{1}{N_m(t)-1}(N_m(t) - n_i(t))n_i(t) \tag{9}$$

The average overhead cost of all messages generated is：

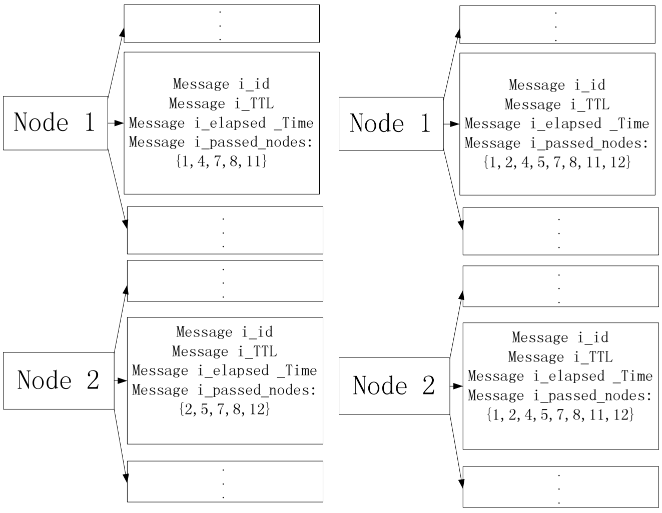
$$\overline{oc}=\left\{\sum_{i=1}^{N_m(t)}\left[n_i(t)+\frac{R_i}{\alpha}*\frac{1}{N_n-1}(N_n-n_i(t))n_i(t)\right]\right\}/N_m(t) \qquad (10)$$

We differentiate $oc_i$ with respect to *ni(t)*， then we get Eq. 11.

$$\Delta oc_i=1+\frac{R_i}{\alpha(N_m(t)-1)}(N_m(t)-2n_i(t)) \qquad (11)$$

K.Shin and S.Kim [10] obtained by simulation using EBMP strategy and other policies such as FIFO, Random, etc., most of the messages were delivered in less than 10 hops to the destination nodes. Because ONs have a lot of nodes and at a time the number of copies of message *i* is far less than the number of nodes in the network that is $N_n-2n_i(t)>0$, and $\Delta oc_i>0$, $oc_i$ is an increasing function on $n_i(t)$. Therefore to minimize $\overline{oc}$, we should discard the copy of message with the largest of $oc_i$ when the buffer is full, equivalent to drop the copy of message with the largest $n_i(t)$ until the remaining buffer space to accommodate the new incoming messages. Contrarily, we should forward the message with the smallest $n_i(t)$ first.

**Establishing the path list for every generated message.** In order to maximize $\overline{dr}$, minimize and minimize $\overline{oc}$, we know that it is reasonable to send message with smallest $n_i(t)$ first, and drop the message with largest $n_i(t)$ first when the buffer is full after the preceding analysis. Then we need to know the value of $n_i(t)$ , the number of copies of message *i* at time *t*. Unfortunately, it is almost impossible for us to find the accurate value of $n_i(t)$ owe to intermittent connectivity, and without end-to-end path in ONs. Our proposed solution is to get the close to the precise value.



1.a Lists before two nodes meet 1.b Lists after two nodes meet

Fig. 1 The changes of lists maintained by each node after they meet

We do this by establishing a path list for every generated message and implementing a learning process to gather the global information about which nodes each of generated message has passed. Every node maintains a list of all the passed nodes of each message carried by them. Two nodes exchange their lists of the messages carried by them, when they meet each other. If message *i* carried by node 1 has passed the nodes including {1,4,7,8,11} and message *i* carried by node 2 has passed the nodes including {2,5,7,8,12}, the nodes that message *i* carried by both of them has passed will change into {1,2,4,5,7,8,11,12} after they meet each other as described in Figure 1.

After a period of time, the nodes that are carrying message *i* will all have the similar value of the number of nodes that message *i* has passed in that time. In the same manner other generated messages will all get the approximate value of the number of nodes that they have passed.

## Simulation results and analysis

In this section, we will verify the optimal message dropping and sending schemes by simulation. We employ epidemic routing protocol and the Random Waypoint (RWP) model to evaluate our policy and other existing classical strategy. In the results of simulation, we focus on the delivery ratio, the average delivery delay and the average network overhead.

**Simulation Environments and Parameter Settings.** We apply the Opportunistic Network Environment simulator (ONE) as the simulator. In this simulator, we assume that all the nodes are with the same communication ranges, and when every two nodes are in communication range, they can setup the connection, and stop transferring when they are out of communication range. We do not consider the impact of signal fading and signal interfering as well. The concrete parameter settings are list in Table 2.

Table 2 Simulation Parameters

| Routing protocol | Epidemic routing | Mobility model | RWP |
|---|---|---|---|
| Transmission speed | 250KB/s | Simulation time | 12h |
| Simulation Area(m) | 1500*1000 | Number of nodes | 225 |
| Node speed | 0.8~2.4m/s | Message size | 50KB~5M |

**Performance Comparison.** In this sub-section, we compare OBSP policy with other classical buffer management strategies, such as Random, FIFO, Youngest and EBMP.

In the certain circumstances of the certain number of nodes in the network and the transmission range, the simulation obtained message delivery rate, average transmission delay and network overhead of several policies as the buffer size in the network changes to changes comparison of the results are shown in Figure 2, 3, 4.

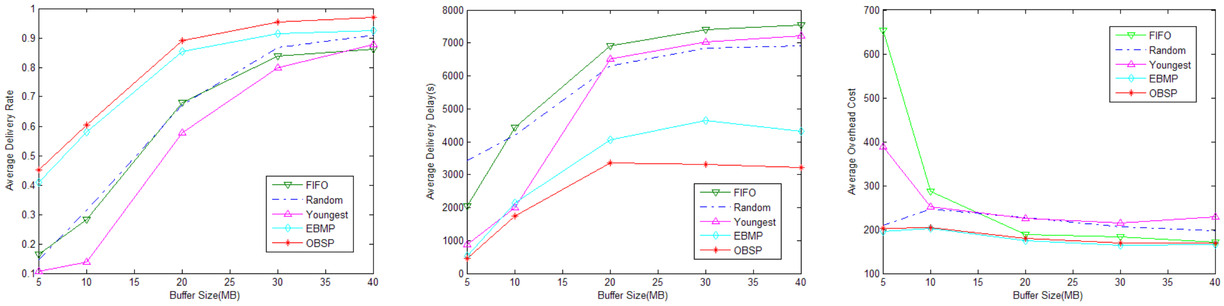


Fig. 2 Fig. 3 Fig. 4

Fig.2-4 Average delivery ratio, average delivery delay and average overhead cost VS buffer size

From Fig. 2, we can find that for every buffer strategy the average delivery rates will almost all increase with the buffer size increasing and our strategy outperforms other buffer policies on the aspect of the average delivery ratio in Epidemic routing algorithm. In this figure, our strategy obviously shows its advantage over other buffer management strategies. That is mainly because our strategy gives the messages with the relative less copies more forwarding opportunities.

On the aspect of average delivery delay shown in Fig. 3, we can find that our policy is also the best among these buffer management strategies. In Epidemic routing algorithm, OBSP obtains the least time delay, because the messages with the relative less copies get more chance to be sent. The messages with the relative more copies have been in the networks for a relatively long time, however, it still does not reach the destination, so we think the old messages has slim probability to be arrived at the destination in the future. Thus, it is reasonable to send the new messages first.

On the aspect of average overhead cost shown in Fig. 4, we can find that our strategy gains the least overhead cost. That is because our policy discards the copy of the message with most copies which contribute to the increasing of overhead cost, while the relatively newly created messages with fewer copies in ONs have little impact on overhead cost.

## Conclusion

In this paper, we propound an optimal buffer forwarding and discarding strategy for ONs. Our policy improves the overall performance of routing algorithms in ONs. In order to testify the superiority of proposed policy, we have implemented many simulations and compared it to some other buffer policies in simulator ONE. The simulation results show that our policy outperforms existing policies with respect to delivery rate, delivery delay and overhead cost.

Note that in this paper, we consider that all messages have the same TTL (time to live). In future we plan to consider different TTL and combine OBSP with other routing algorithms.

## Acknowledgement

This paper is Supported by the National Science Foundation of China (61071118), the Science and Technology Research Project of Chongqing Municipal Education Commission of China (KJ110524, KJ111506), the Natural Science Foundation Project of CQ CSTC (CSTC2010BA4010, cstcjjA40039), Supported by the National Science Foundation of Chongqing University of Posts and Telecommunications (A2011-15). First Author is corresponding author.

## Reference

[1] K. Fall, “A delay tolerant network architecture for challenged internets”, in Proc. ACM SIGCOMM, pp.27-24, 2003.

[2] S. Jain, K. Fall, and R. Patra, “Routing in a delay tolerant network,” In Proceedings of ACM SIGCOMM, Aug. 2004.

[3] Vahdat and D. Becker, “Epidemic routing for partially connected ad hoc networks, ” Duke University, Tech. Rep. CS-200006, 2000.

[4] K. P. Thrasyvoulos Spyropoulos and C. Raghavendra, “An efficient routing scheme for intermittently connected mobile networks,” ACM SIGCOMM workshop on Delay Tolerant Networking (WDTN-05), 2005.

[5] A. Lindgren, A. Doria, and O. Schelen, “Probabilistic routing in intermittently connected networks,” SIGMOBILE Mobile Computing and Communication Review, vol. 7, no.3, 2003.

[6] Etienne C. R. de Oliveria, Celio V. N. de Albuquerque, "NECTAR: a DTN routing protocol based on neighborhood contact history", in Proc. ACM Symposium on Applied Computing, pp. 40-46, 2009.

[7] A. Ker¨anen, J. Ott, “The ONE Simulator for DTN Protocol Evaluation,” in SIMUTools, 2009.

[8] A. Krifa, C. Baraka, and T. Spyropoulos, “Optimal Buffer Management Policies for Delay Tolerant Networks,” in Proc. IEEE SECON, pp. 260-268, 2008.

[9] V. Erramilli and M. Crovella, “Forwarding in Opportunistic Networks with Resource Constraints,” in Proc. ACM, 2008.

[10]K.Shin, S.Kim, “Enhanced buffer management policy that utilizes message properties for delay-tolerant networks”, in the Institution of Engineering and Technology, pp. 753-759, 2011.