DTN中基于Spray and Wait的综合缓存管理策略

**崔建群，余东海，常亚楠，孙佳悦，邬尧**



华中师范大学计算机学院，湖北 武汉 430079

(E-mail：[jqcui@126.com](mailto:jqcui@126.com))

**摘 要** DTN(Delay Tolerant Network)区别于传统Internet网络，无法提供稳定端到端的连接，因此采用“存储-携带-转发”路由机制进行数据传输，而这使得消息需要长时间驻留在节点的缓存中。由于节点的缓存空间和处理能力受价格、体积和功耗的限制，因此如何对缓存进行管理成为影响路由性能的重要因素。本文根据Spray and Wait路由算法的特点，提出一种基于Spray and Wait的综合缓存管理策略(Integrated Buffer Management Strategy Based on Spray and Wait,IBMS)。该策略综合考虑消息大小、消息生存时间和消息副本数三种消息属性，确定消息的优先级，并根据优先级对消息进行转发和丢弃，同时增加了ACK确认机制，删除冗余消息，提升网络资源利用率。仿真结果表明，IBMS缓存管理策略能够在消息投递率、网络开销和消息传输延时方面有明显的提升。

**关键词**：DTN；消息属性；优先级；缓存管理策略

**中图法分类号** TP393

**Integrated Buffer Management Strategy Based on Spray and Wait in DTN**

CUI Jian-qun, YU Dong-hai, CHANG Ya-nan, SUN Jia-yue and WU Yao

School of Computer Central China Normal University, Wuhan 430079, China

**Abstract:** Delay Tolerant Network (DTN), unlike traditional Internet networks, does not provide a stable end-to-end connection, so a storage-carrier-forward routing mechanism is used for data transmission, which causes messages to reside in the node's buffer for a long time. Since the buffer space and processing capacity of nodes are limited by price, volume and power consumption, how to manage the buffer becomes an important factor affecting the routing performance. According to the characteristics of the Spray and Wait routing algorithm, an Integrated Buffer Management Strategy Based on Spray and Wait (IBMS) is proposed in this paper. This strategy comprehensively considers three message attributes, namely message size, message TTL and message copy number, determines the priority of the message, forwards and discards the message according to the priority, and at the same time adds ACK-check mechanism to delete redundant messages and improve network resource utilization. The simulation results show that IBMS buffer management strategy can improve the message delivery rate, network overhead and message transmission delay.

**Keywords:** DTN; Message attributes; Message priority; Buffer management strategy

1 引言

延迟容忍网络[1](Delay Tolerant Networks，DTN) 是指能在长时延、连接频繁断开等极端网络环境下进行通信的一种新型网络体系。不同于传统Internet网络，DTN具有间歇连接、资源有限、时延高以及异构互联等特点[2]，在环境监测、军事战略、深空探测、灾难救援和偏远地区以及发展中国家网络基础设施建设等方面具有广泛的应用前景和实用价值[3]。

基于DTN网络的特点，DTN采用“存储-携带-转发”[4]路由机制进行数据传输，然而这种传输方式就使得消息需要长时间驻留在节点的缓存中，又由于缓存空间的大小有限，因此选择合适的缓存管理策略显得尤为重要。

传统的缓存管理策略只考虑一种消息属性对消息进行丢弃，具有一定盲目性，且只考虑消息的丢弃顺序而忽略了消息的转发顺序。因此，不同于传统的缓存管理策略，本文针对Spray and Wait算法提出一种综合缓存管理策略IBMS，综合考虑消息大小、消息生存时间和消息副本数三个消息属性确定消息的效用值，根据消息的效用值对消息进行转发和丢弃，不仅如此，本文算法还加入了ACK确认机制，以删除已经投递成功的冗余消息，提高网络资源利用率。

本文从以下几个部分对IBMS算法进行介绍：第二部分介绍了相关工作，列举了包括传统缓存管理算法在内的缓存管理策略；第三部分介绍了消息阈值设置和消息效用值定义，详细描述了三种消息属性如何进行结合；第四部分具体描述IBMS算法流程和相关伪代码；第五部分是将IBMS算法与其他缓存管理算法进行比较，并对实验结果进行分析；第六部分是对全文内容的总结以及下一步改进的方向。

2 相关工作

本节中将按照时间顺序介绍过去几年中出现的DTN缓存管理策略，并对其优缺点进行分析。

最早由文献[5]提出的下列四种丢弃策略：

DRA(Drop-Random): 随机选择消息进行丢弃，相对比较公平，但有一定的盲目性。

DLR(Drop-Least-Recently-Received)：丢弃缓冲区中存在时间最长的消息，思想是认为在缓冲区中存在时间越长，转发的次数越多，可以进行丢弃。

DOA(Drop-Oldest)：丢弃整个网络中存在时间最长的消息，因为在网络中停留较长时间的消息有更高的可能性已经被成功投递。

DLE(Drop-Least-Encountered)：基于消息位置和移动丢弃到消息目的节点投递概率最小的消息，相比之前三个算法具有较强的自适应能力。

文献[6]针对Epidemic[7]算法提出了两个消息丢弃策略：drop tail和drop head，分别是丢弃新到达的消息和丢弃最旧的消息，并根据仿真结果表明drop head的策略能够取得更好的路由性能。

文献[8]提出了下列五种丢弃策略：

FIFO(First-In-First-Out)：首先删除第一个进入队列的消息。

MOFO(Evict-Most-Forwarded-First)：首先删除转发次数最多的消息，这样做可以给转发次数少的消息更多转发机会，可以得到较高的投递率。

MOPR(Evict-Most-Favorably-Forwarded-First)：首先丢弃最有利转发的消息，具体做法是每个节点保存一个FP值，每当消息被转发，FP按照FP=FPold+P进行更新，其中P是接收节点对应该消息的投递概率，是根据Prophet[9]算法进行计算。

SHLI(Evict-Shortest-Life-Time-First-Time-First)：首先丢弃最短生存时间的消息，类似之前DOA丢弃策略，在平均延时方面是最好的。

LEPR(Evict-Least-Probable-First)：首先丢弃投递概率最小的消息，与Prophet算法的思想一致：节点最不可能传递投递概率值低的消息。

文献[10]针对Binary Spray and Wait[11]路由算法提出了基于二分散发和等待路由的自适应拥塞控制策略DLC。主要思想是用消息副本数作为转发次数、存在时间和散发能力的综合表现，优先丢弃消息副本数少的消息来达到提高投递率和减少开销的目的。

文献[12]针对Epidemic算法提出了依据报文质量的拥塞控制策略DmQ，通过估计报文拷贝数并综合消息生存时间确定报文丢弃的优先级，根据优先级丢弃优先级低的报文得到更好的网络性能。

文献[13]针对Epidemic路由算法综合消息生存时间和转发次数，将消息生存时间和消息在网络中的转发次数进行归一化处理，提出一种基于消息特性的缓存管理算法MPBBM，将消息生存时间取对数处理使之与消息转发次数处于同一数量级，并将两者叠加作为消息的优先级，根据消息的优先级决定丢弃顺序，有效提高Epidemic算法的性能。

文献[14]不同于以上缓存管理策略，考虑了消息大小因素，提出了可感知大小的丢弃策略SAD，其主要思想是根据消息大小尽可能最小化丢弃副本数量来达到最小化网络开销的目的，以避免频繁丢弃消息浪费网络资源。

综上所述，消息大小主要影响网络开销、消息生存时间主要影响传输时延、消息转发次数主要影响投递率。然而，很少有缓存管理策略能够综合考虑这三种消息属性，且大多数缓存管理策略都是针对Epidemic算法进行改进，很少有根据Spray and Wait消息转发特性专门针对Spray and Wait算法的缓存管理策略。因此，本文提出一种基于Spray and Wait的综合缓存管理策略，通过同时考虑消息大小、消息生存时间和消息转发次数决定消息的转发和丢弃顺序，以平衡Spray and Wait路由算法的投递率、传输时延和网络开销三种指标。

本文通过与SHLI、DLC、MPBBM和SAD的仿真数据进行分析和对比，CCMQ在投递率、传输时延和网络开销等网络性能方面有了较大的提升。

3 消息阈值设置和消息效用值定义

由于消息大小、消息生存时间和消息转发次数随着仿真时间的增长变化幅度不同，因此简单地进行叠加来定义消息的优先级是不科学的，本文采用定义消息大小和消息生存时间的阈值来将三种属性进行结合。

3.1 消息大小阈值

消息大小直接影响需要丢弃的消息数量，因此定义消息大小阈值并对缓存中消息进行筛选可以尽可能减少丢弃消息数量，以达到降低网络开销的目的。消息大小阈值设置如下式(1)所示：

式子中表示将要接收消息大小，表示缓存空闲大小。消息大小阈值表示接收消息需要空出的缓存空间大小。消息丢弃策略中首先筛选出消息大小大于阈值的消息进行丢弃，可以最小化消息丢弃数量，从而降低网络开销。

3.2 生存时间阈值

消息生存时间影响消息传输时延，因此本文定义生存时间阈值。消息生存时间阈值如下式(2)所示：

式子中表示消息初始生存时间，为0~1之间的数，后文中通过实验确定在当前仿真环境下的值，从而确定生存时间阈值。

3.3 消息效用值

消息效用值根据生存时间阈值进行定义，决定了消息的转发和丢弃顺序。消息效用值如下式(3)所示：

式子中表示节点携带消息的消息副本数，表示该节点中消息的剩余生存时间。

在消息生存时间小于生存时间阈值时，则消息生存时间成为丢弃消息所需要考虑的主要因素，优先丢弃消息生存时间小的消息，以降低传输延时。

当消息生存时间大于阈值时，消息生存时间就不再是比较丢弃顺序的主要因素，本文根据Spray and Wait的算法特性(消息初始副本数与消息副本数差值即为消息的转发次数)，将消息副本数作为效用值，以增大消息投递率。

4 基于Spray and Wait的综合缓存管理策略IBMS

本文提出的基于Spray and Wait的综合缓存管理策略IBMS综合了节点携带消息大小、生存时间和副本数，对Spray and Wait算法的消息的转发和丢弃策略进行改进，本节从ACK确认机制、消息转发和丢弃顺序、消息转发策略以及消息丢弃策略四个方面进行介绍。

4.1 ACK确认机制

DTN的ACK确认机制不同于传统Internet的TCP确认机制，其目的并不是为了提供可靠传输，而只是为了根据确认信息丢弃已经投递的冗余消息，缓解网络压力，达到降低网络负载，提高投递率的目的。

本文中ACK确认机制的实现方式：

(1)每个节点维护一个ACK消息列表，记录节点中已经投递成功的消息ID。

(2)每当两个节点建立连接时，就会交换彼此的表，并将对方的表中的消息ID添加到自己的表中，然后检索各自的表，判断缓存中是否含有表中消息ID对应的消息，若有则删除。

(3)节点有消息成功投递到目的节点时，记录消息ID添加到表中，并将消息从缓存中删除。

具体ACK确认算法如算法1所示。

|  |
| --- |
| **算法1 ACK确认算法** |
| 输入：节点与相遇  Initialize ,   1. exchange and for each other 2. IF != null THEN 3. .addAll() 4. FOR each id in DO 5. IF .hasMessage(id) and !isSending(id) 6. .deleteMessage(id) 7. END IF 8. END FOR 9. END IF 10. IF != null THEN 11. .addAll() 12. FOR each id in DO 13. IF .hasMessage(id) AND !isSending(id) 14. .deleteMessage(id) 15. END IF 16. END FOR |

4.2 消息转发和丢弃顺序

根据之前定义的消息效用值对缓存中消息进行排序，得到如图1所示的消息转发和丢弃顺序。

图1 消息转发和丢弃顺序

当消息效用值相同时则比较消息大小，消息大小较小的消息具有更高的效用值，这是为了在丢弃时腾出更多的缓存空间容纳更多消息，接收时也可以接收更多的消息。若消息大小也相同则按照进入缓冲区时间按照FIFO顺序转发和丢弃消息。

具体消息效用值排序算法如算法2所示。

|  |
| --- |
| **算法2 消息效用值排序算法** |
| 输入：缓存中未排序的两个消息,  输出：与排序顺序相同返回1，否则返回-1   1. IF > and >  4. ELSE  7. diff = - 8. END IF 9. IF diff == 0 10. diff = - 11. IF diff == 0 12. diff = - 13. END IF 14. END IF 15. RETURN (diff > 0 ? -1 : 1) |

4.3 消息转发策略

按照上述消息转发和丢弃顺序，优先转发消息效用值大的消息，效用值相同则比较消息大小，优先转发消息大小较小的消息；消息大小也相同则优先转发较早进入缓冲区的消息。

4.4 消息丢弃策略

当两节点相遇时，首先进行ACK确认机制，删除两节点中已经成功投递的消息；当接收节点空闲缓存不足以容纳邻居节点即将传输过来的消息时,根据消息大小筛选出消息大小大于或等于消息大小阈值的消息，优先丢弃效用值低的消息；若消息大小均小于则优先丢弃效用值低的消息并重新计算。具体消息丢弃策略图2所示。



图2 消息丢弃策略

需要补充的是消息效用值相同，则优先丢弃消息大小较大的消息，以空出更多空间容纳更多的消息；若消息大小仍相同则按照FIFO算法思想优先丢弃较早进入缓存的消息。

具体消息丢弃算法如算法3所示。

|  |
| --- |
| **算法3 消息丢弃算法** |
| 输入：即将进入消息队列消息大小size  输出：丢弃成功返回TRUE，失败返回FALSE   1. IF size > BufferSize 2. RETURN FALSE 3. END IF 4. WHILE 6. FOR each m in Messages 7. IF m.getSize() ≥ 8. Selected.add(m) 9. END IF 10. END FOR 11. IF !Selected.isEmpty() 12. oldest = getLeastUtility(Selected) 13. IF oldest == null 14. RETURN FALSE 15. END IF 16. deleteMessage(oldest.getId(), true) 17. freeBufferSize += oldest.getSize() 18. ELSE 19. oldest = getLeastUtility(Messages) 20. IF oldest == null 21. RETURN FALSE 22. END IF 23. deleteMessage(oldest.getId(), true) 24. freeBufferSize += oldest.getSize() 25. END IF 26. RETURN TRUE |

5 仿真结果与分析

5.1 仿真环境配置

本实验采用ONE仿真平台分别在Spray and Wait路由算法的基础上对DLC，MPBBM，SHLI，SAD和本文提出的IBMS算法进行仿真，采用ONE平台自带的赫尔辛基市地图作为移动场景，大小为4 500 m × 3 400 m。仿真参数设置如表1所示，其余参数均采用平台默认配置文件default\_settings.txt中的默认设置。

表1 仿真参数

Table1 Simulation parameters

|  |  |
| --- | --- |
| Simulation parameters | Parameter value |
| simulation time | 12hours |
| Bluetooth bandwidth | 250kBps |
| Message generation rate | 25-35 seconds |
| TTL | 300 minutes |
| Node cache | 5MB |
| Message size | 500kB - 1MB |
| Transmission range | 10m |
| Number of nodes | 126 |
|  | 0.35 |

5.2消息生存时间阈值参数设置

由于生存时间阈值直接影响何时比较副本数，何时比较消息生存时间，即效用值的确定，进而影响消息投递率和传输延时，因此确定固定仿真条件下生存时间阈值显得尤为重要。消息生存时间阈值由消息初始生存时间的比例确定，即为确定比例系数，使得算法在投递率和传输延时方面均能表现较好。

首先在[0,1]区间，以0.05为间距确定参数的大致范围，投递率和平均延时分别如图3和图4所示。

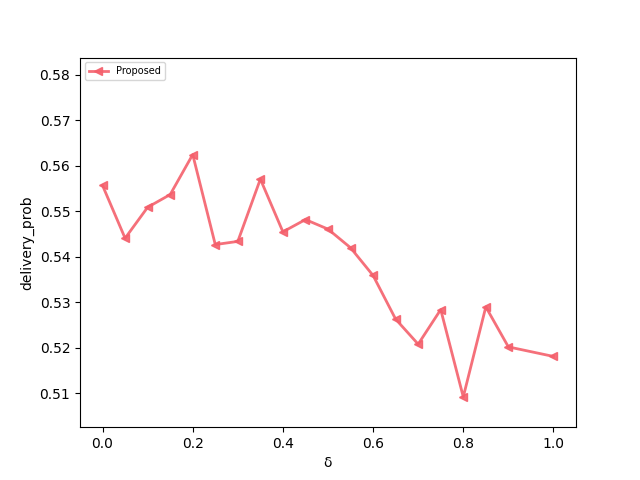


图3 不同下的消息投递率

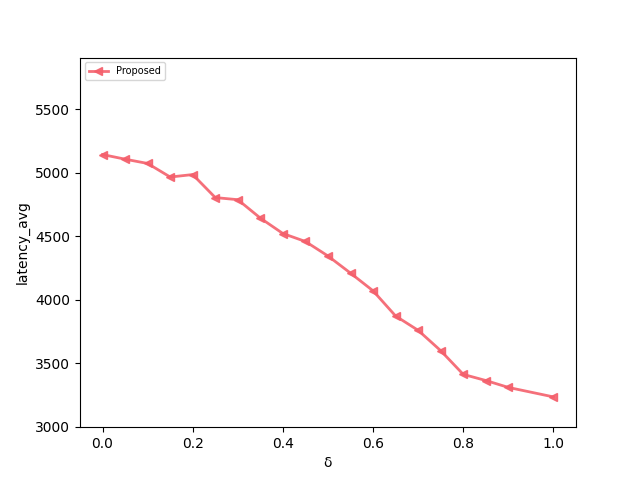


图4 不同下的消息平均时延

可以看出在0.2和0.35时算法投递率相对较高，相差不是很明显，但在传输延时方面为0.35比为0.2下降了很多，因此将参数的取值范围确定在0.35附近。为了进一步精确确定参数的值，在[0.29-0.36]之间以0.01为间距，得到对应算法投递率和平均延时，分别如图5和图6所示。

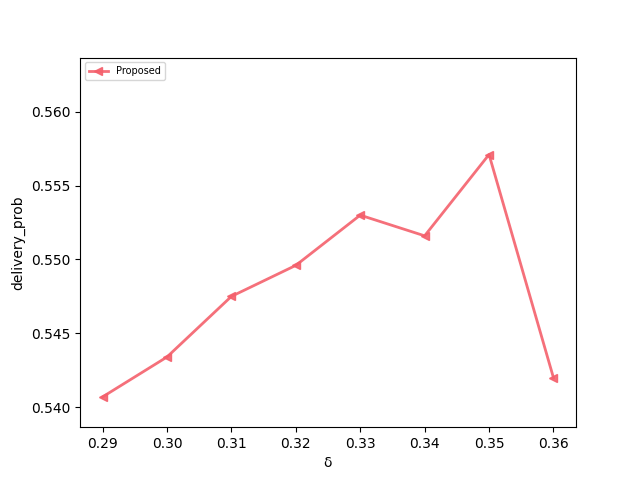


图5 不同下的消息投递率

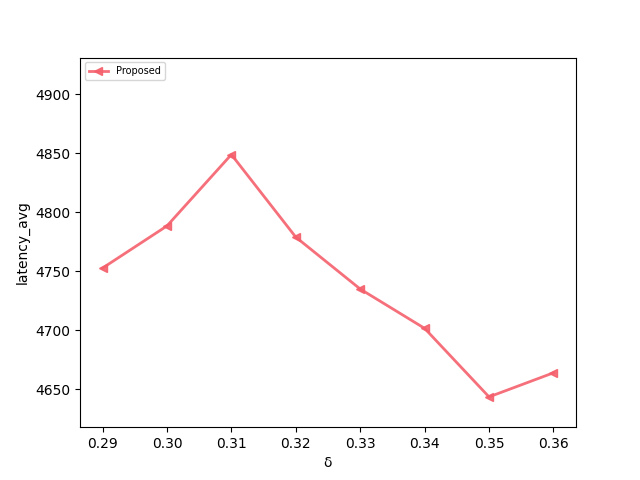


图6 不同下的消息投递率

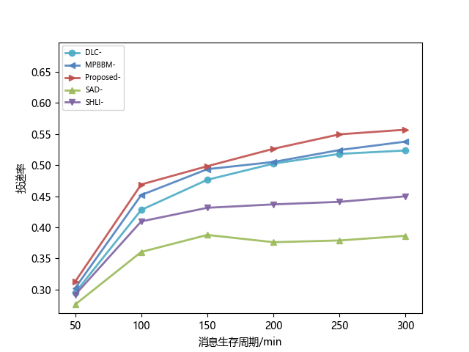
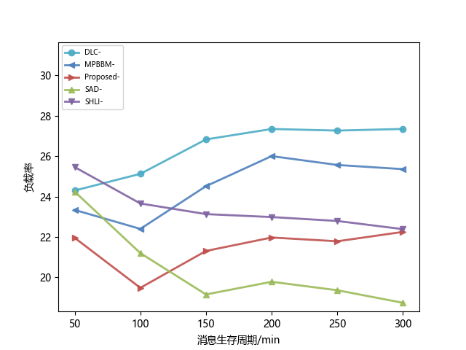
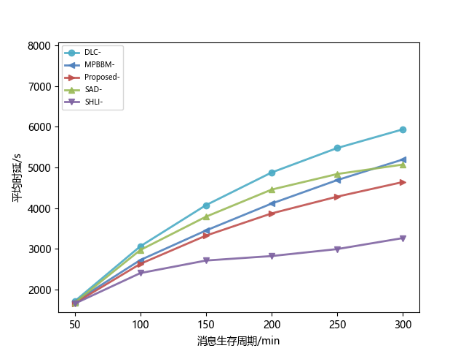
可以很明显看出当参数取0.35时算法在投递率和传输延时综合方面表现最优。

5.3仿真结果及分析

为了验证IBMS算法的高效性，进行了三组仿真实验，在不同的消息生存时间、不同的仿真时间以及不同的消息产生速率下对五种算法在消息投递率、网络负载率和平均传输时延等网络性能方面进行评估，并且除了传统算法SHLI均加入了ACK确认机制以达到控制变量的目的。实验结果表明，IBMS对网络的性能提升最大。

(1)不同消息生存时间

按照表1所示的仿真环境，通过改变消息的生存周期，对五种算法进行对比实验，得到的投递率、负载率和平均时延结果如图7所示。

(a) 消息投递率 (b) 网络负载率 (c) 平均时延(s)

图7 不同的消息生存时间下的性能

图2(a)显示了各个算法在不同的消息生存时间变化下消息投递率的变化情况。由于随着消息生存时间的增加，许多消息不会因为消息生存时间短被丢弃而能够成功投递，因此各个算法的投递率基本呈现上升趋势。其中本文提出的算法IBMS投递率最高：IBMS的消息投递率相较MPBBM平均提高了3.46%，相较DLC平均提高了6.24%，相较SHLI平均提高了17.72%，相较SAD提升了33.59%。

IBMS的消息投递率高于MPBBM的可能原因有两个，一是MPBBM算法未考虑消息的大小，因此可能多次出现接收一个消息而删除多个消息的情况；二是MPBBM算法仅仅将消息生存时间和消息副本数保持在一个数量级，而未考虑两者的占比情况，一般说来消息副本数占比越多则消息投递率越高。

SAD的消息投递率最低的原因是该算法只考虑消息的大小，保证每次丢弃消息的个数最小，并没有对消息生存时间和消息副本数进行考虑，因此可能丢弃消息副本数大即还未得到转发的消息，导致投递率很低。

图2(b)比较了各个路由算法的网络负载率。其中SAD算法的网络负载率最低，这是由于该算法为了使每次丢弃的消息个数最少，因此优先将消息大小大的消息进行丢弃，而这样会为缓冲区腾出大量的空间，使得整个网络的负载率较低。

DLC算法的负载率最高的原因是该算法丢弃副本数最低的消息，即转发次数最多的消息，使得总的消息转发次数大大增加，网络负载率最高。

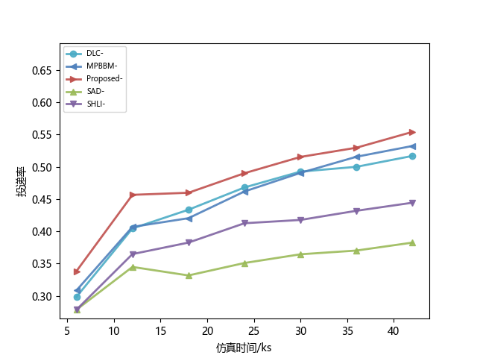
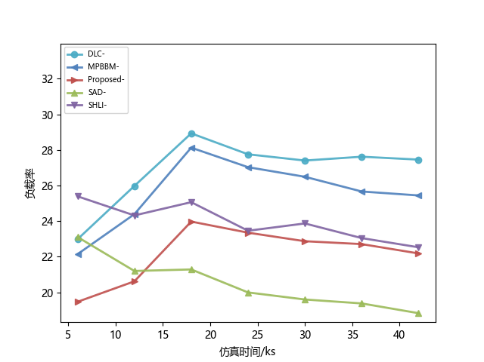
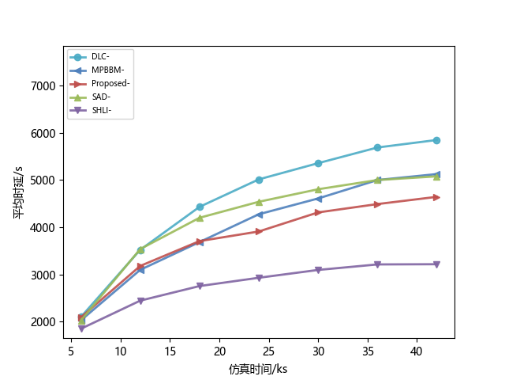
图2(c)评估了各个算法的平均时延的变化情况。从图中可以看出SHLI的平均时延最低，IBMS第二低，这是由于SHLI算法只考虑丢弃生存时间最短的消息，使得成功投递的消息生存时间偏高，计算出的平均时延较低，然而由于可能丢弃一些消息生存时间低但能成功投递的消息，导致该算法的投递率偏低；本文中算法IBMS能够在保持高投递率的情况下，平均延时方面性能仅次于SHLI。

DLC的平均时延是最高的，这是因为该算法 使得转发次数少的消息得到更多的转发机会，这就会导致转发次数少且需要长时间到达目的节点的消息得到转发，造成整体的平均延时较高。

综上所述，IBMS在消息投递率方面优于其他四种算法，在网络负载率和平均时延方面均是次优，因此可以得出在不同的消息生存时间IBMS的整体性能是最好的。

(2)不同的仿真时间

同样的仿真环境，通过改变仿真时间，比较各种算法三种指标，得到的实验结果如图8所示。

(a) 消息投递率 (b) 网络负载率 (c) 平均时延(s)

图8 不同的仿真时间下的性能

图3(a)显示了各个算法在不同的仿真时间下消息投递率的变化情况。随着仿真时间的增加，有更多的消息被成功投递，因此各个算法的投递率基本呈现上升趋势。其中本文提出的算法IBMS投递率最高：IBMS在不同仿真时间下的投递率相较MPBBM平均提高了7.03%，相较DLC平均提高了7.82%，相较SHLI平均提高了22.32%，相较SAD提升了37.33%。

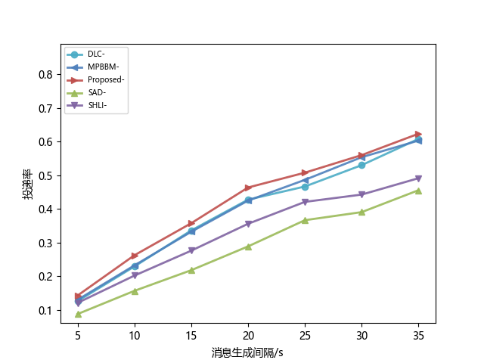
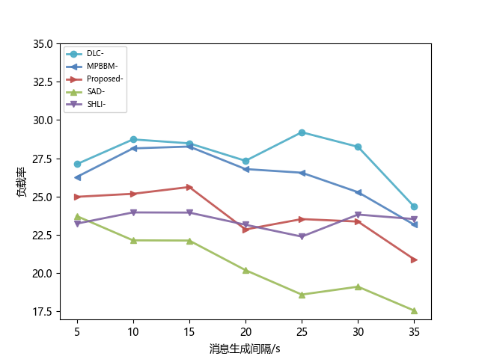
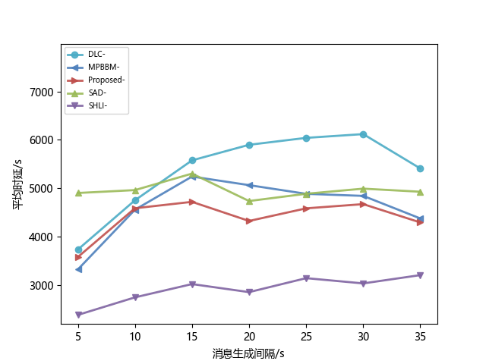
图3(b)比较了各个算法的网络负载率。结果依然是SAD算法网络负载率最低，IBMS算法次优。IBMS算法的网络负载率随仿真时间的增加先增加后趋于稳定，是因为先开始消息的生存时间基本大于消息生存时间阈值，因此大都按照消息副本对消息进行丢弃，使得消息得到大量转发，网络负载率升高；随着时间的推移，消息生存时间占主要因素，因此网络负载率趋向于SHLI算法。

图3(c)评估了在不同仿真时间下网络的平均时延的变化。从图中可以看出随着仿真时间的增加，IBMS算法的平均时延逐渐增加并趋于平缓，仍然是高于SHLI算法，比其他三种算法低，处于次优。

综上所述，随着仿真时间的增加，由于IBMS综合考虑了消息大小、消息生存时间和消息副本数，与其他算法的优势逐渐明显，从总体上来看，IBMS在各个方面都有显著优势。

(3)不同的消息生成间隔

消息生成间隔与消息产生速率呈反比，通过改变消息生成间隔，即改变消息产生速率，来比较五种算法的三种指标，得到实验结果如图9所示。

(a) 消息投递率 (b) 网络负载率 (c) 平均时延(s)

图9 不同消息生成间隔下的性能

图4(a)显示了各个算法的消息投递率的变化。五种算法的投递率都呈现上升的趋势，这是因为随着消息生成间隔的增加，消息产生速率逐渐变小，网络中的消息逐渐变少，资源利用率变高，减少大量消息因为缓存不足而频繁丢弃的情况，因此投递率均呈现上升的趋势，这时对于消息的转发和丢弃策略直接影响着消息是否能成功转发到目的节点。对比图中的数据可以得出，IBMS的消息投递率相较MPBBM平均提高了6.93%，相较DLC平均提高了8.52%，相较SHLI平均提高了26.00%，相较SAD提升了53.45%。

图4(b)比较了各个算法的网络负载率随消息产生速率的变化情况。与之前不同的是当消息生成间隔低的时候，IBMS算法的网络负载率会高于SHLI算法，这是因为消息生成间隔低，就会有大量新消息的产生，由于新产生的消息的生存时间基本大于消息生存时间阈值，所以基本按照消息副本数对消息进行丢弃，许多消息被转发，网络负载率提高，而随着消息生成间隔增加，新消息的生成速率降低，网络负载率逐渐降低，由于同时考虑了消息大小和消息生存时间，因此比单一考虑消息生存时间的SHLI网络负载率低。

图4(c)评估了各个算法的平均时延在不同的消息产生速率下平均时延的变化情况。随着消息产生速率逐渐变小，IBMS的平均时延方面的性能仍仅次于SHLI算法，DLC算法的平均时延依然为最高。

综上所述，在不同的消息产生速率条件下，综合考虑投递率、负载率以及平均时延方面，IBMS对网络的性能提升最大。

6 结束语

本文基于Spray and Wait路由算法提出了考虑了消息生存时间、消息大小和消息副本数三种消息属性的综合缓存管理策略。该算法首先通过消息大小尽可能减少消息丢弃的数量，减少因丢弃消息大小不足造成频繁丢弃的情况，降低网络负载率，再按照消息生存时间和消息副本数对消息优先级进行排序，实现消息投递率和平均传输时延的平衡，最后通过加入ACK确认机制，删除网络中的大量冗余消息副本，缓解网络拥塞情况，进一步提升网络整体性能。仿真结果显示，对比其他四种算法，IBMS在投递率上始终是最高的，在网络负载率和消息传输时延方面是次优的，因此从整体上来看IBMS要优于其他四种缓存管理策略，但是由于消息生存时间阈值是根据本文仿真环境实验进行确定，经过组内研究讨论，下一步将研究如何动态确定消息生存时间阈值，从而使得IBMS能够适应不同缓存等其他仿真环境。

**参考文献**

[1] H Guo, X Wang, M Huang, et al. 2010, 37(011): 12-18.(In Chinese)

郭航，王兴伟，黄敏，等. 容延容断网络研究及进展[J]. 计算机科学. 2010, 37(011): 12-18.

[2] L Zhang, X Zhou, J Wang, et al. 2010(10): 164-182.(In Chinese)

张龙，周贤伟，王建萍，等. 容迟与容断网络中的路由协议[J]. 软件学报. 2010(10): 164-182.

[3] Fall, Kevin. A Delay-Tolerant Network Architecture for Challenged Internets[J]. Sigcomm03 Aug. 2003: 27-34.

[4] Fall K, Scott K L, Burleigh S C, et al. Delay-Tolerant Networking Architecture[J]. Hse Ztschriften Verlag. 2007.

[5] Davis J A, Fagg A H, Levine B N. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks[Z]. 2001141-148.

[6] Zhang X, Neglia G, Kurose J, et al. Performance modeling of epidemic routing[C]. NETWORKING 2006 - Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, 5th International IFIP-TC6 Networking Conference, Coimbra, Portugal, May 15-19, 2006, Procee, 2006. 2006.

[7] Vahdat A, Becker D. Epidemic Routing for Partially-Connected Ad Hoc Networks[J]. Handbook of Systemic Autoimmune Diseases. 2000.

[8] Lindgren A, Phanse K S. Evaluation of Queuing Policies and Forwarding Strategies for Routing in Intermittently Connected Networks[C]. Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on, 2006. 2006.

[9] Lindgren A, Doria A, Schelén O. Probabilistic routing in intermittently connected networks[J]. Acm Sigmobile Mobile Computing & Communications Review. 2003, 7(3): 19.

[10] G Wang, B Wang, J Zhang. 2010, 27(011): 4237-4241.(In Chinese)

王贵竹，王炳庭，张家勇. DTN中基于二分散发和等待路由的自适应拥塞控制策略[J]. 计算机应用研究. 2010, 27(011): 4237-4241.

[11] Spyropoulos T, Psounis K, Raghavendra C S. Spray and wait: An efficient routing scheme for intermittently connected mobile networks[J]. Proceedings of ACM SIGCOMM workshop on Delay-tolerant networking. 10.1145/1080139.1080143. 2005.

[12] G Wang, Z Xu, X Li. 2012, 48(9): 74-77.(In Chinese)

王贵竹，徐正欢，李晓峰. DTN中依据报文质量的拥塞控制策略[J]. 计算机工程与应用. 2012, 48(9): 74-77.

[13] H Wang, H Hu, J Zhu, et al. 2015, 44(03): 403-409.(In Chinese)

王慧强，胡海婧，朱金美，等. 面向DTN感染路由协议的缓存管理算法[J]. 电子科技大学学报. 2015, 44(03): 403-409.

[14] Moetesum M, Hadi F, Imran M, et al. An adaptive and efficient buffer management scheme for resource-constrained delay tolerant networks[J]. Wireless Networks. 2016, 22(7): 1-13.