基于平均传递概率的容迟网络路由算法的设计*

吕杰林[1]，张珊珊[2]
（1. 浙江商业职业技术学院 继续教育学院，杭州 310053；2. 北京理工大学 软件学院，北京 100081）

摘 要：为了提高容迟网络的传递率、降低传输延迟、对节点缓存进行更有效的管理，结合已有的 PROPHET 和 Spray and Wait 算法，提出了一种基于平均传递概率的容迟网络路由算法 RAB-ADP。在该算法中设置了一个与时间有关的平均传递预测概率参数进行消息转发的决策，解决了 PROPHET 算法容易产生路由抖动的缺点。算法综合利用了复制和知识两个属性，采用{MOPR；FIFO}队列策略组，通过消息传送完毕的 ACK 确认信息进行缓存管理和网络中冗余消息副本的删除。仿真实验表明，该算法在节点缓存大小不同以及网络中节点数目不同的两种情况下，传递率和路由开销比率的性能均优于其他经典路由算法。

关键词：容迟网络；路由算法；平均传递概率；消息转发
中图分类号：TP393 **文献标志码**：A **文章编号**：1001-3695(2014)01-0248-05
doi:10.3969/j.issn.1001-3695.2014.01.058

Design of DTN routing algorithm based average delivery probability

LV Jie-lin[1], ZHANG Shan-shan[2]
(1. *Dept. of Continual Education, Zhejiang Business College, Hangzhou* 310053, *China*; 2. *School of Software, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: In order to improve the delivery ratio, reduce the average latency, manage the buffer effectively in delay tolerant network, this paper proposed RAB-ADP, a routing algorithm based on the average delivery probability, by combining the existing PROPHET and Spray and Wait algorithm. This algorithm set a time-related average delivery prediction probability parameters to make the message forwarding decisions, solved the problem of route flap in PROPHET. RAB-ADP comprehensively used two properties of replication and knowledge, used {MOPR; FIFO} queuing policy group, managed the buffer as well as delete the redundant copies of the message in the network by generating ACK confirmation message when the message was delivered to the destination. Simulation results show that the algorithm has a better performance than other classic algorithms in delivery ratio and overhead ratio in the case of different buffer size and the number of nodes.

Key words: delay tolerant network(DTN); routing algorithm; average delivery probability; message forwarding

容迟网络(DTN)时断时续、延迟比较大、节点缓存空间和能量有限，传统路由算法无法应用于容迟网络[1]。目前比较典型的容迟网络路由算法有传染性路由算法(epidemic routing)、Spray and Wait 算法和 PROPHET 算法。Epidemic 算法是将消息分发给同一连通子网内的节点，称为载体，通过载体与其他连通子网中节点的接触，将消息传播到其他连通子网，最终到达接收节点[2]。该算法具有快速传播、传输方式简单直接等优点，但其消耗资源较多，节点间竞争有限的内存资源和网络资源，从而导致许多消息被丢弃和重发，并且网络中消息的拷贝数与网络规模成正比[3]。Spray and Wait 算法结合了 Epidemic 算法的优势，同时也解除了消息拷贝数与网络规模的耦合，可以获得更好的性能[4]。在 Epidemic 算法中假设节点的运动是完全随机的，而事实上节点通常以一种可预测的方式运动，其移动模型可以根据过去一段时间内的重复模型来进行推断[5]。基于这种假设，基于相遇历史和可传递性的概率路由算法 PROPHET 被提出[5]。该算法在带宽、缓存等资源受限的情况下性能优于 Epidemic 算法，但该算法只是将消息转发到了少数节点，会导致消息的传递延迟增加[3]。

针对现有 DTN 路由算法的不足，本文重点关注网络拓扑结构无法事先得知的随机动态 DTN 网络的路由策略，消息在何时以何种方式转发到何地是该类网络中路由策略研究的关键。本文设计了一种基于平均传递概率的 DTN 路由算法 RAB-ADP，通过将 PROPHET 算法中的传递预测概率值优化为一个与时间有关的平均值，解决了其具有的路由抖动问题。该算法综合利用复制和知识两个属性，并不需要事先得知网络的知识信息，而是通过记录节点的相遇时间间隔、传递预测概率来计算出一个平均值用于转发决策。同时采用一定的复制策略和缓存队列管理策略，进而达到提高传递率、降低延迟时间、减少网络资源消耗的目的。

1 算法的设计

在容迟网络环境中，需要很多中继节点，而这些中继节点又是随机运动的，整个网络的拓扑结构是未知的。同时网络中可能会存在许多消息，甚至出现网络中消息拥塞的情况。所以在进行路由算法设计时需要重点考虑以下几个问题：a）当两个节点相遇时如何决定是否进行消息的转发；b）在消息转发

收稿日期：2013-04-21；**修回日期**：2013-05-27 **基金项目**：国家自然科学基金资助项目(61101214)

作者简介：吕杰林(1973-)，男，浙江永康人，副院长，讲师，硕士，主要研究方向为数据挖掘技术和网络路由技术(lvjielinzj@163.com)；张珊珊(1989-)，女，硕士研究生，主要研究方向为容迟网络路由技术.

过程中如何最大程度地保证消息成功传递到目的节点;c)当网络中的消息过多时如何在中继节点有限的缓存中进行管理;d)当消息已经到达目的节点时如何通知网络中该消息的其他副本。整个算法的核心思想是如何消耗最少的网络资源、使用最短的延迟时间实现最高的消息传递率。

1.1　与时间有关的衡量参数

在 RAB-ADP 算法中,借鉴了 PROPHET 算法基于概率的策略,对消息传递成功的概率进行估算,选择性地对消息进行复制。在算法中定义了一个传递预测值作为衡量参数来描述节点间传递成功的概率。该衡量参数的计算方法与 PROPHET 算法不同,它不是单纯地根据节点的相遇历史对节点间的传递成功概率进行预测,而是一个与时间有关的平均值,从而使该衡量参数能够维持在一个相对稳定的范围内,避免因外界因素如网络故障等导致衡量参数波动而造成路由抖动的问题。

平均传递预测值 P_{avg}的计算是在传递预测值 P 的基础上进行的。传递预测值 P 的计算方法如下[5]:

a)在每个节点 a 上建立一个对每个已知目的节点 b 的传递预测概率值 $P_{(a,b)} \in [0,1]$,表示 a 将消息传递给 b 的可能性。

b)每当节点 a 遇见一个节点 b 时,更新相应的传递预测概率值,使得经常相遇的节点具有较高的传递预测概率值。更新时的依据如式(1)所示,其中 $P_{\text{init}} \in [0,1]$是一个初始常量。

$$P_{(a,b)} = P_{(a,b)\text{old}} + (1 - P_{(a,b)\text{old}}) \times P_{\text{init}} \tag{1}$$

c)当节点 a 在很长时间内没有遇见节点 b 时,相应的传递预测概率值会减小(称为衰减)。遵循的公式如式(2)所示,其中 γ 是衰减常量,k 表示该度量值自上次衰减经历的时间单位的数量。

$$P_{(a,b)} = P_{(a,b)\text{old}} \times \gamma^{k} \tag{2}$$

d)传递预测概率值具有传递性。如果节点 a 经常遇到节点 b,节点 b 经常遇到节点 c,那么可以认为节点 a 向节点 c 能够以较高成功率转发消息。具体的计算式如式(3)所示,其中 $\beta \in [0,1]$是一个比例常数,它决定了传递性对传递预测概率值的影响程度。

$$P_{(a,c)} = P_{(a,c)\text{old}} + (1 - P_{(a,c)\text{old}}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{3}$$

为了避免路由抖动,两个节点之间的传递预测概率应该趋于一条平滑的变化曲线。每个节点的传递预测概率应该反映网络中一段较长时间的性能,而不是某一个相遇时间点的峰值。为此需要为每个节点的传递预测概率取一个平均值,代替每次相遇时更新的传递预测概率值,节点依据该平均值决定是否进行消息转发。但是,这个平均值不能简单取网络中所有值的平均,通过测试发现该值与最近两个到三个点的关系较明显,因此,采用局部平均、整体相关的策略,即每次计算平均传递概率时都要综合考虑上次相遇时的平均传递概率以及本次相遇时的传递概率。具体计算方法如下:

a)当节点 a 和节点 b 第一次相遇时,节点 a 到节点 b 转发消息概率如式(4)(5)所示。

$$P_{1(a,b)} = P_{\text{init}} \tag{4}$$

$$P_{1(a,b)\text{avg}} = P_{1(a,b)} \times t_1 / t_1 = P_{1(a,b)} \tag{5}$$

节点 a 需要记录从网络初始化到与节点 b 第一次相遇所经历的时间长度 t_1,以及在相遇时间点 T_1 获得的新传递概率 $P_{1(a,b)}$。

b)第二次相遇时,节点 a 在相遇时间点 T_2 所获得的传递概率值如式(6)所示。

$$P_{2(a,b)} = P_{1(a,b)} + (1 - P_{1(a,b)}) \times P_{\text{init}} \tag{6}$$

同时,节点 a 还要记录两个节点从第一次相遇时间点 T_1 到第二次相遇时间点 T_2之间的间隔时间 t_2。

$P_{2(a,b)}$是节点 a 在相遇时间点 T_2 获得的向节点 b 转发消息的传递预测概率值,但两个节点并不根据这个值来转发,而是通过 $P_{1(a,b)\text{avg}}$、$P_{2(a,b)}$、t_1、t_2 生成一个与时间有关的平均传递概率 $P_{2(a,b)\text{avg}}$。该平均传递概率既考虑了节点 a 和 b 第一次相遇时的平均传递概率,也考虑了第二次相遇时的传递概率,如式(7)所示。

$$P_{2(a,b)\text{avg}} = (P_{1(a,b)\text{avg}} \times t_1 + P_{2(a,b)} \times t_2) / (t_1 + t_2) \tag{7}$$

c)每个节点在决定是否进行转发时的衡量参数就是平均传递预测概率值。

首先需要计算节点 a 在相遇时间点 T_n 所获得的传递概率值,如式(8)所示。

$$P_{n(a,b)} = P_{n-1(a,b)} + (1 - P_{n-1(a,b)}) \times P_{\text{init}} \tag{8}$$

该平均传递预测概率值是由前面节点所得到的平均传递预测概率值 $P_{n-1(a,b)\text{avg}}$、上一次相遇距离其上一次相遇的时间间隔 t_{n-1}、本次相遇距离上一次相遇所经历的时间 t_n,以及当前节点的传递预测概率值 $P_{n(a,b)}$共同决定的。时间间隔的选择如图 1 所示。具体的计算如式(9)所示。

$$P_{n(a,b)\text{avg}} = (P_{n-1(a,b)\text{avg}} \times t_{n-1} + P_{n(a,b)} \times t_n) / (t_{n-1} + t_n) \tag{9}$$

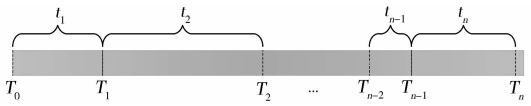

图 1　时间间隔选择示意图

PROPHET 算法利用一个统一的参数来对待长时间两个节点未相遇的传递率衰减,这种方法在高延迟的网络环境中会大大降低原本传递率较高的两个节点的传递率。同时在传递过程中又要将长时间未连接成功的情况考虑进去。为了解决高延迟情况下某些节点传递概率较低的问题,同时将这些特殊节点对其他节点的影响保持在一个较低水平,提出了基于平均传递概率的路由算法。由于该算法的影响因子中只有前一节点和当前节点的一些信息,所以占用内存较少,同时由于每个节点的取值都是与前一节点计算后的平均值,这样可以降低某个特殊节点的高延迟影响。由于这种算法在计算中只关心前一节点的计算结果和与前一节点的时间间隔,如果与特殊节点相距超过两个节点,就不会有过多的影响。所以该算法不会将特殊节点的高延迟进行过多扩散,一般在经过三个节点以后,这种影响效果会大大降低。

1.2　转发策略

在转发策略上,RAB-ADP 综合利用了复制和知识两个属性,并且借鉴 Spray and Wait 算法的路由方案,以达到提高传递率、降低传输延迟的目的。

Spray and Wait 算法是由 Spray 和 Wait 两个阶段组成的。在 Spray 阶段,对于源节点产生的每一个消息将产生 L 个副本,并将这 L 个副本转发给不同的中继节点[4]。在 Wait 阶段,如果在 Spray 阶段没有发现目标节点,携带着消息副本的这 L 个节点将进行直接传递操作(即将传递到目标节点)[4]。

但是如何对消息产生的这 L 个副本进行传递呢?经实验证明,Binary Spray and Wait 算法具有最好的性能。具体传递

方式为:对于源节点起始时拥有一个消息的 L 个副本,任何一个拥有 $L>1$ 个消息副本的节点 a(源节点或中继节点)遇到一个没有该消息副本的节点 b 时,将 $L/2$ 个消息副本传递给节点 b,当一个节点只有消息的 1 个副本时,进行直接传输,即进入 Wait 阶段[4]。

RAB-ADP 算法的转发阶段借鉴的是 Binary Spray and Wait 算法,但是当遇到没有消息副本的节点 b 时,不会直接进行传递操作,而是先进行相对于目标节点的平均传递预测概率值的比较。同时使用的是知识属性中开始对网络的状态信息不了解,但是在转发的过程中自主地进行学习。

具体的过程为当持有消息 msg 的 L 个副本的节点 a 与节点 b 相遇时,首先节点 a 和节点 b 分别更新各自的传递预测概率值,并计算出各自的平均传递预测概率值;然后需要判断当前的情况是否满足 spray 阶段的条件,即节点 a 中的消息 msg 的副本数 L 是否大于 1,并且节点 b 中没有消息 msg 的副本,如果不满足则不进行消息的传递;同时还要判断节点 b 的平均传递预测概率值是否大于 a,只有大于 a 的情况下才进行传递,否则消息 msg 还将继续被保存在节点 a 的缓存中,以等待更好的传递机会。如果节点 a 中只有消息 msg 的一个副本,则进入 wait 阶段,直到遇到消息 msg 的目的节点时才进行传递。

在 RAB-ADP 算法中通过设定消息的属性来记录节点具有的消息副本数,每次进行消息传递时,只需要更新该属性的值即可。具体的操作为首先获取当前连接的所有消息,并获取消息的副本数 nrofCopies,如果 nrofCopies > 1,则进入 spray 阶段;如果连接中的另外一个节点拥有该消息的副本,则跳过不进行传递操作,否则比较两个节点的平均传递预测概率值,如果连接中的另外一个节点的平均传递预测概率值大于该节点,则将消息加入到本次要传递的消息列表中,当遍历完成该连接中的所有消息后统一进行消息的传递操作。

1.3 节点缓存队列策略

当网络阻塞时,节点可能需要在缓存中长时间地保存消息,而节点的缓存资源是有限的,当需要保存的消息数目过多时,需要决定将哪些消息从缓存队列中删除,并决定将哪些消息转发给其他节点。这就需要一定的队列策略来实现对节点缓存的有效管理。

常用的队列策略有:a)先进先出(first in first out,FIFO),先进入队列的消息先被删除[6];b)优先删除最多转发次数(evict most forwarded first,MOFO),通过记录每个消息已被转发的次数,优先删除转发次数最多的消息,使得转发次数较少的消息获得更多的转发机会[6];c)优先删除最高转发概率(evict most favorably forwarded first,MOPR),每个节点为其缓存队列中的每条消息记录一个 FP 值(初始为 0),每当一条消息被转发,则更新该条消息对应的 FP 值,$FP = FP_{old} + P$,其中 P 为接收节点对该条消息的传递预测概率值,FP 值最大的消息优先被删除[7];d)优先删除最短存活时间(evict shortest life time first,SHLI),在 DTN 体系结构中[5]为每条消息设定了一个存活时间值,剩余最短存活时间的消息被优先删除[8]。

选取哪种队列策略与采用的路由算法密切相关,如 MOPR 策略需要计算传递预测概率值,不适于应用到在进行转发决定时不考虑传递预测概率值的 Epidemic 算法中,但却可以很好地应用于 PROPHET 算法。而单一的队列策略很难做到全面地对节点缓存进行管理,很容易出现顾此失彼的情况。

在 RAB-ADP 算法中,由于在衡量参数方面借鉴了 PROPHET 算法,在进行转发决定时使用了与时间有关的平均传递预测概率值,所以队列策略中可以采用 MOPR 策略,但是当两条消息的 FP 值相同时,很难进行队列的管理,因而混合使用了{MOPR; FIFO}的队列策略组。首先根据 MOPR 策略进行消息的删除,当遇到两条消息的 FP 值相同时,删除优先进入队列的消息。

1.4 ACK 确认机制

当一条消息到达目的节点时,网络中其他具有该消息副本的中继节点并不知道其已经到达,还会继续转发复制这些副本。而这些冗余副本的转发会增大路由开销,并且大大增加对网络资源的竞争,导致网络阻塞和资源浪费[9]。

为了提高算法的性能,减小路由开销,在 RAB-ADP 算法中采用了 ACK 确认机制。消息到达目的节点时生成一个 ACK 确认信息,当两个节点建立连接时交换彼此具有 ACK 信息的消息列表,并删除已经具有 ACK 信息的消息副本。这样能够有效地通知网络中的中继节点删除冗余的消息副本,对降低网络资源消耗,减轻网络拥塞起到了很好的作用。

所有具有 ACK 信息的消息 id 保存在称为 ackedMessageIds 的表中,当成功执行了消息的传递操作后,接收节点会判断传递的这条消息是否已经到达目的节点。如果已到达,则将该消息的 id 加入到 ackedMessageIds 表中。

当两个节点建立连接时,会互相交换彼此的 ackedMessageIds 表,并分别调用 deleteAckedMessages 函数来执行删除自己所持有的具有 ACK 信息的消息副本,即网络中冗余的消息。deleteAckedMessages 函数执行删除操作的具体步骤为遍历当前节点的 ackedMessageIds 列表中的所有消息,将自己持有并且不是正在进行传递操作的消息删除。

2 算法的执行流程

假设在一个容迟网络中,节点 a 与节点 b 相遇建立连接,节点 a 是消息传递节点,节点 b 是消息接收节点,消息需要传递到目的节点 d。RAB-ADP 算法的执行流程如图 2 所示。

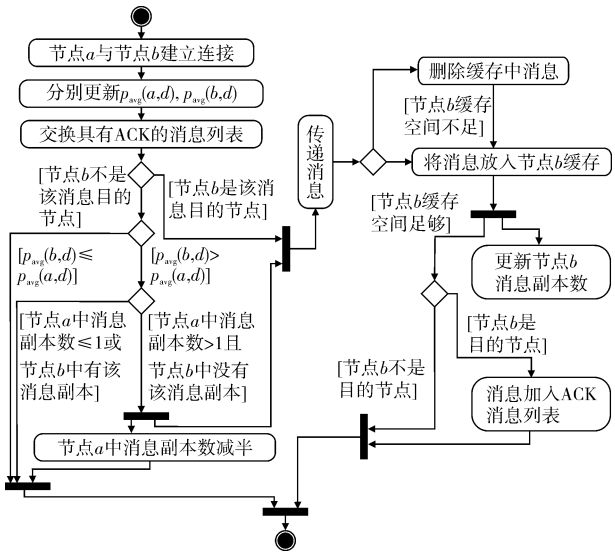

图 2 算法的执行流程

a)节点 a 和节点 b 分别更新传递预测概率值,并根据公式计算出平均传递预测概率值。

b)双方传递具有 ACK 信息的消息列表,并删除已经到达目的节点的消息副本。

c)如果接收节点 b 是消息的目的节点,则直接传递消息;否则只有当发送节点 a 具有某条消息的副本数大于 1 且节点 b 没有该消息的副本,接收节点到达目的节点平均传递预测概率大于发送节点,才进行消息的传递,并且只传递发送节点拥有消息副本数的一半。

d)当接收节点 b 收到发送节点 a 传递的消息时,首先要检查缓存是否有足够的空间接收该消息,如果没有,需要先按照{MOPR; FIFO}队列策略组删除缓存中的消息,然后将该消息的副本放入缓存中,并且更新本节点具有该消息的副本数;然后判断当前节点是否是该消息的目的节点,如果是,则将该消息加入 ACK 列表中。

3　实验分析

为了客观地对 RAB-ADP 算法的性能进行评估,本文采用 ONE 平台[10]对所设计的算法进行仿真实验。实验中采用控制变量的方法,分别在缓存大小不同和网络中节点数目不同的两种情况下,将 RAB-ADP 算法与其在进行算法设计过程中借鉴的 PROPHET 和 Binary Spray and Wait 算法以及传统的 Epidemic 算法进行性能上的比较。在比较的过程中关注的性能指标是传递率(delivery ratio)、平均延迟时间(average latency)和路由开销比率(overhead ratio)三个方面。

3.1　仿真场景设置

仿真实验中选取的场景是赫尔辛基市 4500 m × 3400 m 的无线网络区域[10]。整个仿真过程持续时间为 43000 s,大约相当于真实世界中的 12 h[1]。仿真实验中共设置了 126 个节点,分为 6 组:2 组行人,每组分别有 40 个节点;1 组汽车,有 40 个节点;3 组电车,每组分别有 2 个节点。为每个节点设置的缓存大小为 10 MB,节点的移动速度为 0.5 ~ 1.5 m/s,并有 0 ~ 120 s 的等待时间。采用消息产生类来生成消息,每次间隔25 ~ 35 s 产生一个新消息,每个消息的大小为 500 KB ~ 1 MB。

3.2　不同缓存大小下算法性能的比较

节点的缓存资源是有限的,当消息过多超出缓存容纳的范围时需要根据一定的队列策略删除消息。而不同的缓存大小影响着删除的消息数量,从而影响了算法的性能。在本组实验中,缓存大小分别取 2 MB、5 MB、10 MB、15 MB、20 MB 五种情况进行比较。

由图 3 可以看出,随着缓存大小的增加,PROPHET 和 Epidemic 算法的传递增长率较慢,Binary Spray and Wait 和 RAB-ADP 算法增长较快,而 RAB-ADP 算法的传递率一直高于 Binary Spray and Wait 算法,当缓存大小达到较高水平时,逐渐趋近于 Binary Spray and Wait 算法。分析其具体原因为:a)Epidemic 算法采用的是洪泛策略,随着缓存大小的增加,缓存中能够保存的消息数量增加,但是网络中产生的消息副本也在大规模增加,缓存会不断地根据相应的队列策略删除旧消息,为新消息提供更多的空间,这样使得节点间随机相遇的概率降低,导致传递率降低;b)PROPHET 算法是基于历史和可传递性的概率路由算法,在转发时会根据相遇历史计算出传递预测概率值来决定是否转发,因而在缓存较小的情况下优于 Epidemic 算法,然而随着缓存大小以及网络中产生的消息副本的大量增加,节点间相遇的概率降低,可能出现节点长时间不能相遇而导致传递预测概率值大幅度衰减,从而降低传递率;c)Binary Spray and Wait 和 RAB-ADP 算法对转发过程中的消息副本数进行了限制,从而节省了缓存空间用来保存更多的消息,因此这两种算法随着缓存大小的增加传递率增长较快。RAB-ADP 算法不会进行盲目的转发,在转发过程中根据相遇历史计算出传递预测概率并利用最近两次的相遇时间间隔计算出平均传递预测概率值,因而传递率一直高于 Binary Spray and Wait 算法。然而随着缓存的继续增加,可能出现 PROPHET 算法遇到的问题,所以当缓存大小达到 15 MB 和 20 MB 时,传递率趋近于 Binary Spray and Wait 算法。

图 4 显示了随着缓存大小增加四种路由算法的平均延迟时间变化情况。可以看出 RAB-ADP 算法的平均延迟时间高于 Binary Spray and Wait 算法,但是低于 PROPHET 算法和 Epidemic 算法。这是由于在进行消息转发时,RAB-ADP 算法要根据平均传递预测概率值来决定是否进行转发,因而需要在缓存中停留更长的时间来等待更好的转发机会,所以平均延迟时间较长,这个指标也是 RAB-ADP 算法在继续研究过程中需要改进的方向。

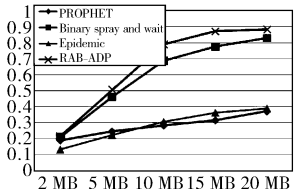

图 3　不同缓存大小下各路由算法传递率对比图

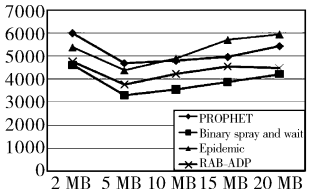

图 4　不同缓存大小下各路由算法平均延迟时间对比图

图 5 显示了随着缓存大小的增加四种路由算法的路由开销比率的变化情况。可以看出,随着缓存大小的增加,PROPHET 和 Epidemic 算法的路由开销比率较高,而 Binary Spray and Wait 和 RAB-ADP 算法一直维持在较低的水平。路由开销比率从某种程度上来说反映了节点对缓存的有效利用程度,由于在 PROPHET 和 Epidemic 算法中,缓存的增大会造成消息副本过多,不能使消息最大程度地进行传递,降低了成功传递消息的数量,所以路由开销比率较高。Binary Spray and Wait 和 RAB-ADP 算法通过限制消息的转发数量,对缓存进行了有效管理,因而成功传递消息的数量较多,缓存的有效利用程度较高,路由开销比率较低。而 RAB-ADP 算法又凭借其不盲目转发的优势,路由开销比率略低于 Binary Spray and Wait 算法。

3.3　不同节点数目下算法性能的比较

随着网络中节点数目的增加,转发的消息数目也会相应增长。为了评测 RAB-ADP 算法在不同网络规模下的性能,设置了本组实验。在本组实验中,网络中的节点数目分别取 66、96、126、156、186 五种情况。网络中节点的缓存大小限定为 10 MB。

图 6 显示了随着节点数目的增加四种路由算法的传递率

变化情况。可以看出,PROPHET 和 Epidemic 算法随着节点数目的增加,传递率增长缓慢,甚至当节点数目达到 126 及其以上时,这两种算法的传递率开始缓慢下降。这是由于节点数目的增加使得网络中产生的消息副本数也大量增加,增大了消息到达目的节点的可能性。而当节点数目过多导致网络中的消息过多时,超出了节点缓存能够处理的范围,造成了网络阻塞,使得传递率下降。Binary Spray and Wait 和 RAB-ADP 算法由于对消息的转发方式进行了限制,消息数目的增加为其传递成功率的增加增大了可能性,而对于缓存的有效管理使得出现网络阻塞的概率大大降低。其中 RAB-ADP 算法凭借其基于平均传递预测概率的转发策略优势,传递概率一直高于 Binary Spray and Wait 算法。

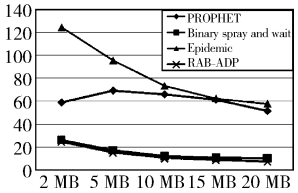

图 5 不同缓存大小下各路由算法路由开销比率对比图

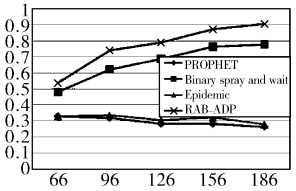

图 6 不同节点数目下各路由算法传递率对比图

图 7 显示了随着节点数目的增加四种路由算法平均延迟时间的变化情况。可以看到,随着节点数目的增加,四种路由算法的平均延迟时间都处于下降的趋势。这是由于节点数目的增加,使得网络中的节点密度变大,这样消息从源节点到达目的节点的可能性也就越大,保存在缓存的时间也就相应变短,从而使得平均延迟时间下降。而 RAB-ADP 算法受其需要等待进行转发的合适平均传递预测概率值的限制,平均延迟时间高于 Binary Spray and Wait 算法。

图 8 显示了随着节点数目的增加四种路由算法路由开销比率的变化。可以看到,随着节点数目的增加,PROPHET 和 Epidemic 算法的路由开销比率急剧增大,这是由于这两种算法在节点数目增大的情况下,中继节点转发的消息总数在增加,但又因为网络阻塞造成成功到达目的节点的消息数目无法相应增加而导致资源的浪费。Binary Spray and Wait 和 RAB-ADP 算法能在节点数目增加的情况下,有效地管理缓存,使得路由开销比率维持在一个较低的水平。其中 RAB-ADP 算法的路由开销比率略低于 Binary Spray and Wait 算法。

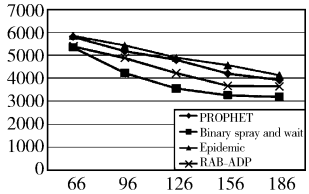

图 7 不同节点数目下各路由算法平均延迟时间对比图

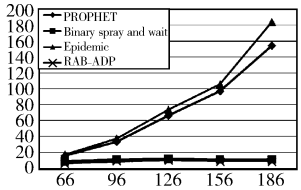

图 8 不同节点数目下各路由算法路由开销比率对比图

3.4 实验结果分析

通过以上两组实验可以看出,在缓存大小不同以及网络中节点数目不同的两种情况下,RAB-ADP 算法的传递率一直高于 PROPHET、Epidemic 和 Binary Spray and Wait 三种算法,路由开销比率一直处于四种算法中最低的水平,性能优于其他三种算法。而平均延迟时间的性能不够理想,始终较高,需要在以后的研究过程中继续改进。

4 结束语

本文结合已有的 DTN 路由策略的优势,针对其中仍旧存在的问题,提出了基于平均传递概率的 DTN 路由算法 RAB-ADP,对其性能进行了客观的评价。本文的主要成果如下:a)在 RAB-ADP 算法中通过利用一个与时间有关的平均传递预测概率值代替 PROPHET 算法中传递预测概率值来进行转发时的决策,有效解决了当出现网络故障等因素时,节点之间长时间不能相遇导致 PROPHET 算法中的传递预测概率值衰减而造成的路由抖动问题,达到了提高传递率的目的;b)综合利用了复制和知识两个属性,在进行消息转发时,借鉴 Binary Spray and Wait 算法中对转发消息的数目进行了限制,每次传递只是传递已有消息副本数量的二分之一,并且利用平均传递预测概率值来决定是否进行转发,避免了网络中出现严重阻塞的可能,有效利用了网络资源;c)为了删除网络中已经到达目的节点的消息存留在网络中的冗余副本,采用了 ACK 确认机制,实现了对缓存的有效管理,降低了网络资源消耗。

参考文献:

[1] 薛静锋, 范志安, 李建胜,等. 基于历史信息预测转发概率的 DTN 路由算法[J]. 北京理工大学学报, 2011, 31(1): 49-53.

[2] VAHDAT A, BECKER D. Epidemic routing for partially-connected Ad hoc networks[R]. Durham, North Carolina: Duke University, 2000.

[3] VOYIATZIS A. A survey of delay and disruption tolerant networking applications[J]. Journal of Internet Engineering, 2012,5(1): 331-344.

[4] SPYROPOULOS T, PSOUNIS K, RAGHAVENDRA C S. Spray and wait: an efficient routing scheme for intermittently connected mobile networks[C]//Proc of ACM SIGCOMM Workshop on Delay-Tolerant Networking. 2005: 252-259.

[5] LINDGREN A, DORIA A, SCHELEN O. Probabilistic routing in intermittently connected networks[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2003,7(3):19-20.

[6] LINDGREN A, PHANSE K. Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks [C]//Proc of COMSWARE. 2006: 1-10.

[7] XIAN Yi, HUANG C, COBB J. Look-ahead routing and message scheduling in delay-tolerant networks[J]. Computer Communications,2011,34(18):2184-2194.

[8] CERF V. RFC 4838, Delay-tolerant network architecture[S]. [S. l.]:IETF,2007.

[9] 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(11): 4237-4241.

[10] KERANEN A, OTT J, KARKKALINEN T. The ONE simulator for DTN protocol evaluation[C]//Proc of the 2nd International Conference on Simulation Tools and Techniques. 2009: 1-10.