2021 年 3 月 第 3 期
Vol. 42 No. 3 2021

机会网络中基于节点相似率的概率路由算法

崔建群,吴淑庆,常亚楠,黄东升

(华中师范大学 计算机学院,武汉 430079)
E-mail: jqcui@126.com

摘 要: 机会网络中的通信设备大多是随着时间的流逝而进行移动的,然而节点之间的移动路径又具有一定的重复性. 因此,可以记录节点移动时与之相遇的节点之间的信息,利用该信息对路由算法做出更合理的决策. 本文根据节点之间的相遇历史信息,提出了一种基于节点相似率的概率路由算法(Probabilistic routing algorithm based on node similarity rate,S-Prophet),对传统的 Prophet 算法的预估节点传输概率阶段进行改进. 首先统计网络中节点与其他节点的相遇集合,定义节点相似率,设计一个新的节点投递概率公式,并根据节点相遇持续时间对 Prophet 路由算法的概率衰减公式进行改变,最后,通过仿真实验验证 S-Prophet 的有效性.

关 键 词: 机会网络; 路由算法; 节点相似率; ACK 确认机制
中图分类号: TP393 **文献标识码:** A **文 章 编 号:** 1000-1220(2021)03-0609-06

Probabilistic Routing Algorithm Based on Node Similarity Rate in Opportunity Network

CUI Jian-qun,WU Shu-qing,CHANG Ya-nan,HUANG Dong-sheng
(School of Computer,Central China Normal University,Wuhan 430079,China)

Abstract: The communication devices in the opportunity network are mostly moved with the passage of time,but the moving paths between nodes are somewhat repetitive. Therefore,it is possible to record the information between the nodes that the node encounters when the node moves,and use this information to make a more reasonable decision on routing algorithm. In this article,according to the historical information of node encounters,a Probabilistic routing algorithm based on node Similarity rate(S-Prophet) is proposed to improve the predicted node transmission probability stage of traditional Prophet algorithm. Firstly,the meeting set of nodes and other nodes in the network is counted,the similarity rate of nodes is defined,a new formula of node delivery probability is designed,and the probability attenuation formula of the Prophet routing algorithm is changed according to the duration of node meeting. Finally,the effectiveness of s-prophet is verified through simulation experiments.

Key words: opportunity network; routing algorithm; node similarity rate; ACK mechanism

1 引 言

随着无线网络的快速发展,机会网络[1]成为了近几年来无线网络研究领域的热点之一. 机会网络采用存储-携带-转发的模式进行消息的传输,不存在固定的端到端传输路径,不同于传统网络. 在传统网络中,源节点和目的节点处于互不连通的状态时无法进行通信. 而机会网络中,由于节点不停的进行移动,因此可以将信息携带到可以进行互相通信的范围,完成消息的传输. 机会网络是为了解决实际自组织网络中,由于网络无法保持长时间连通的情况下的数据通信问题,可在没有固定路由的情况下实现消息的逐跳转发,最终将消息传输到目的节点.

目前机会网络的可应用场景也越来越多. 例如,在灾难应急[2],车载网络[3,4]以及星际网络[5]等领域都具有非常广阔的市场前景和应用价值.

机会网络中的节点普遍具有较高的移动性,而且机会网络具备连接间歇性,网络延迟大等特点,因此寻求合适的中继节点将消息准确快速的投递到目的节点,是路由算法中最迫切关注的问题. 传统的基于历史预测策略的概率路由(Probabilistic routing Protocol using history of encounters & transitivity,Prophet)[6]算法,没有考虑节点的相遇持续时间,只是利用节点相遇频率作为传递概率,这未免考虑不够周全,同时 Prophet 没有对成功传递的消息进行有效的控制,从而导致存在很多已经被成功投递的消息仍然存在网络中,使得网络开销过大,同时成功投递的消息仍然存在节点中,在节点缓存较小的情况下,这些成功投递的消息占用过多的节点缓存,导致节点无法接收新消息. 本文将结合 Prophet 算法,分析其存在的问题,提出了一种基于节点相似率的概率路由算法(Probabilistic routing algorithm based on node similarity rate in opportunistic network,S-Prophet)算法,引入节点相似率进行消息传递概率的设计,并添加 ACK 确认机制删除成功投递的消息,以减轻节点缓存的负担,减少网络开销,从一定程度上弥补了

收稿日期: 2020-03-23 收修改稿日期: 2020-04-25 基金项目: 国家自然科学基金面上项目(61672257)资助; 国家自然科学基金青年项目(61702210)资助. 作者简介: 崔建群,女,1974 年生,博士,教授,CCF 会员,研究方向为机会网络、物联网、移动网络和应用层组播; 吴淑庆,男,1994 年生,硕士研究生,研究方向为机会网络; 常亚楠(通讯作者),女,1984 年生,博士,副教授,研究方向为无线网络、社交网络、物联网; 黄东升,男,1995 年生,硕士研究生,研究方向为机会网络.

Prophet 算法概率计算不准确,以及节点缓存被成功投递的消息过多占用的问题.

2 相关工作

机会网络中经典算法有 Direct Delivery [7]、Epidemic [8] 以及 Prophet 等.

Direct Delivery(直接传输)算法的主要思想是,源节点只有与目的节点相遇的时候,才将消息进行转发给对方. Direct Delivery 是一种单拷贝路由方式,其优点是简单直接,缺点是消息完全依赖源节点和目的节点的相互通信,若源节点始终无法遇到目的节点,则消息将永远无法送达目的节点. 由于机会网络中节点是不断移动的,这将导致消息的延迟较大,消息容易在生存周期过期后仍无法传递到目的节点,从而导致较低的投递成功率. Epidemic 算法则相反,通过提高副本传播的数量来提高数据成功率,不管相遇节点是否为目的节点,携带消息的节点都会将消息传递给与其相遇的所有节点,邻居节点继续通过这种方式进行下一步的传递. 这种方式解决了 Direct Delivery 算法中依赖于源节点与目的节点相遇才将消息转发的问题,但是带来了另外一个问题,网络中存在大量冗余消息,很快便造成网络拥塞节点缓存负担过大,甚至有可能导致网络瘫痪.

Prophet 算法是通过总结节点间历史相遇的规律来预测未来的传输路径,其主要工作在于提出了节点间成功传输消息概率的一个指标——投递预测值(deliver probability). 与传染病路由算法(Epidemic)相比,当两个节点 V_a 和 V_b 相遇时,不仅要交换两者的向量外,而且要交换投递预测值,只有在 V_a 到目的节点的投递预测值大于 V_b 到目的节点的投递预测值时才将消息传递给节点 V_b. 本文将使用投递概率代表投递预测值这一概念.

投递概率的计算通常分 3 种情况讨论: 更新、衰退以及传递性. 当且仅当节点间相遇时,根据不同情况更新概率预测值.

更新: 当节点 V_a 和节点 V_b 相遇,根据公式(1)更新它们之间的投递概率,其中 $P_{init} \in [0,1]$,是一个人为规定的常数,是节点间传输率的初始值.

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) \times P_{init} \quad (1)$$

衰减: 若节点 V_a 和节点 V_b 在某段时间内未相遇,那么他们再次相遇的概率将降低,其投递概率的值按照公式(2)来进行计算:

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

公式中的 γ 是衰减参数,k 则是表示从最后一次相遇到当前时间所经历的时间块的个数,$\gamma \in [0,1]$ 是常数.

传递性: 节点概率的传递性是指不但节点 V_a 经常遇到节点 V_b,而且节点 V_b 也能经常遇到节点 V_c,那节点 V_c 可能是转发消息到达目标节点 V_a 的比较好的中转节点. 节点间的概率传递性计算如公式(3)所示,其中 $\beta \in [0,1]$ 是常数,它的大小是衡量传递性对投递预测概率的影响的一个重要指标.

$$P(a,c) = P(a,c)_{old} + (1 - P(a,c)_{old}) \times P(a,b) \times P(b,c) \times \beta \quad (3)$$

通过总结节点间历史相遇的规律来预测未来的传输路径,通过计算投递概率,选取概率大的节点作为中继节点,一定程度上避免了 Epidemic 算法在选择中继节点的盲目性. 但 Prophet 算法没有考虑节点的相遇持续时间,只是利用节点相遇频率作为传递概率,这未免考虑不够周全. 同时,Prophet 算法没有对成功传递的消息进行有效的控制,从而导致存在很多已经被成功投递的消息仍然存在网络中,使得网络开销过大,同时成功投递的消息仍然存在节点中,在节点缓存较小的情况下,这些成功投递的消息占用过多的节点缓存,导致节点无法接收新消息.

近年来,对于 Prophet 算法的改进有很多种形式. 段宗涛 [9] 等提出概率路由中基于连接时间的机会转发算法,该算法考虑了相遇持续时间,在设计节点投递概率时增加了节点间连接时间占空比的影响因素,并未讨论缓存管理对投递率的影响. 张峰 [10] 等针对上述问题进一步研究,在节点投递概率计算部分引入相遇持续时间重新设计概率计算公式,在选择中继节点时,虽然考虑了节点缓存对投递概率的影响,但是没有对已经成功投递的消息进行删除,未能从根本上解决当节点缓存被占用过大时,无法有效转发消息的问题. 马慧 [11] 等提出基于节点的历史吞吐率的 Prophet 路由策略,在消息传递的过程中,在与目标节点相遇概率相同的节点中选择吞吐率较大的节点作为中继节点. 虽然该算法考虑了消息传递过程中节点的吞吐率问题,但是在计算吞吐率的时候未能考虑到时间片对吞吐率的影响,因此在消息投递率上并未有明显提高.

本文在此基础上,提出了一种基于节点相似率的概率路由算法. 在计算消息转发概率时考虑了节点历史相遇情况,引入了节点相似率的定义. 由于在预估两个节点的投递概率的时候,Prophet 算法中使用的参数 P_{init} 是固定的,不能真实反映两个节点之间的动态变化关系,本文将使用节点相似率来对概率公式(1)中的参数 P_{init} 进行设计,关于节点相似率的定义将在 3.2 节进行阐述. 同时,使用节点相遇持续时间对衰减公式(2)中 γ^k 的参数 k 进行重新设计,为了减少传输成功的消息副本对网络资源的占用,本文采用了 ACK 删除机制,来保证及时清除已经传输成功的消息副本,然后使用 ONE 仿真平台进行实验,对比改进前后算法之间的性能.

3 机会网络中基于节点相似率的概率路由算法

3.1 基于节点相似率的概率路由算法 S-Prophet

定义 1. 节点相遇持续时间

节点相遇持续时间为两个节点之间建立连接后进行通信所持续的时间,在机会网络中由于节点是移动的,因此造成节点之间每次相遇后进行通信所持续时间可能不同,在本文中,统计每个节点与其他节点之间建立连接通信的次数,以及总相遇持续时间,如图 1 所示.

假设节点 V_a 与节点 V_b 总共建立连接 n 次,则它们之间的总相遇持续时间为:

$$T_{(a,b)} = T_1 + T_2 + \cdots + T_{n-1} \quad (4)$$

由于传统的 Prophet 算法中没有考虑节点的接触时间,当两个节点接触频率很高,但是如果接触时间比较短,那么消息就无法完整的传输给对方,导致传输不得不中断,消息投递率

就会受到影响,因此本文中利用节点相遇持续时间$\frac{T_{(a,b)}}{T}$来改变节点公式 $P_{(a,b)} = P_{(a,b)old} \times \gamma^{k}$ 中的参数 k,在本文中,节点投递概率的衰减公式为公式(5):

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^{\frac{T(a,b)}{T}} \tag{5}$$

其中,T 为时间间隔周期,$T_{(a,b)}$ 表示节点 V_a 和节点 V_b 总的相遇持续时间.

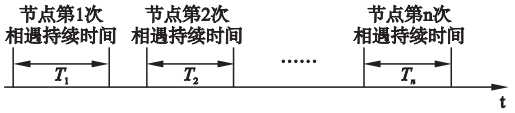

图 1　节点相遇持续时间统计

Fig. 1　Node encounter duration statistics

定义 2. 节点的相似率

节点 V_a 和节点 V_b 的相似率是指节点 V_a 与节点 V_b 拥有的共同相遇节点个数与它们两个节点分别相遇的节点总数的比值. 令集合 $N_a = \{V_c \mid n_{a,c} \neq 0, 1 \leqslant c \leqslant n\}$ 表示节点 V_a 在时间间隔周期 T 内所遇到的相遇节点集合,其中 $n_{a,c}$ 表示节点 V_a 与节点 V_c 的相遇次数,其初始值为 0,每当节点 V_a 与节点 V_c 相遇,则次数加 1. 同理集合 N_b 表示节点 V_b 在时间间隔周期 T 内的相遇节点集合. 当统计的时间超过时间间隔 T 则会将原先的集合清空并重新开始统计,节点 V_a 与节点 V_b 的相似率计算方法见公式(6):

$$Sim(a,b) = \frac{|N_a \cap N_b|}{|N_a \cup N_b|} \tag{6}$$

我们将在图 2 所示的网络中详细演示节点相似率的获得. 如图 2 所示,节点 V_a 和别的节点的相遇集合是 $N_a = \{V_1, V_2, V_3, V_4, V_5, V_6, V_7\}$,节点 V_b 与其他节点的相遇集合为 $N_b = \{V_2, V_3, V_4, V_8, V_9\}$,所以 $N_a \cup N_b = \{V_2, V_3, V_4\}$,$|N_a \cap N_b| = 3$,$N_a \cup N_b = \{V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9\}$,$|N_a \cup N_b| = 9$,因此 $Sim(a,b) = \frac{|N_a \cap N_b|}{|N_a \cup N_b|} = \frac{3}{9} = \frac{1}{3}$.

本文使用公式(7)作为两个节点之间的传输概率公式,不再使用公式(1).

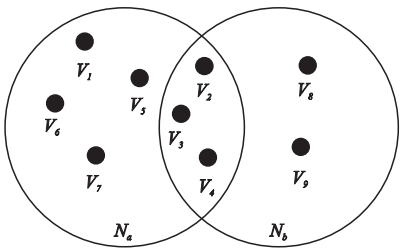

图 2　节点相似率的统计

Fig. 2　Statistics of node similarity rate

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) \times Sim(a,b) \tag{7}$$

3.2　消息转发策略

接下来我们将介绍 S-Prophet 路由算法的消息转发步骤以及消息转发流程.

消息转发流程,如图 3 所示.

步骤 1. 在携带消息的节点 V_a 遇到节点 V_b 时,首先要判断 V_b 是否为目的节点,是则转步骤 4,否则转步骤 2.

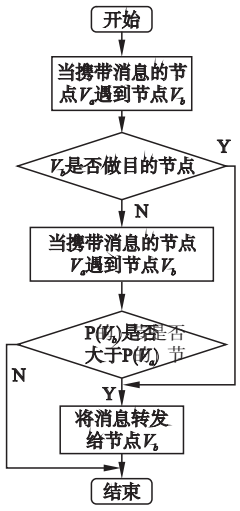

图 3　基于节点相似率的概率路由算法消息转发流程

Fig. 3　Probability routing algorithm based on node similarity rate message forwarding process

步骤 2. 计算节点 V_a 遇到节点 V_b 同目的节点之间的相遇概率.

步骤 3. 要判断节点 V_b 和目的节点的概率 $P(V_b)$ 是否大于节点 V_a 和目的节点的概率 $P(V_a)$,是则转步骤 4,否则结束.

步骤 4. 将消息转发给节点 V_b,结束.

表 1　S-Prophet 算法

Table 1　S-Prophet algorithm

算法 1. S-Prophet 算法

输入: InitPreds( ),InitFreqs( ),InitMeetTime( );
输出: preds,freqs,meetTime;

```
1.  if( con. isup( ) )
2.      startTime = SimClock. getTime( );
3.      if( currentTime-lastEncounterTime) > T
4.       freqs. clear( );
5.      else
6.       updateFreqsFor( );
7.      end if
8.   else
9.       endTime = SimClock. getTime( );
10.   end if
11.   for 集合 Na 和 Nb
12.      updateMeetTime( );
13.      updateSimilarFor( );
14.      updateDeliveryPredFor( );
15.      updateTransitivePreds( );
16.   end for
17.   for 所有属于节点 Va 的消息 msg
18.      if P( Va,Vd) > P( Vb,Vd)
19.       将消息 msg 转发给节点 Vb
20.      end if
21.   end for
```

表 1 中算法第 1 行表示任意两个节点 V_a 与 V_b 相遇并且建立连接,算法第 2 行,记录节点 V_a 与节点 V_b 相遇的开始时

间,算法 3 ~ 7 行进行判断,如果节点 V_a 与 V_b 的相遇时间间隔大于设定的时间周期阈值 T,则将 V_a 的相遇节点列表清空,否则将该相遇节点添加到相遇节点列表中. 因为本文只记录时间周期 T 内的相遇节点信息. 算法第 8 行,记录节点 V_a 与 V_b 断开连接的时间. 算法第 12 行,统计集合 N_a 和 N_b 中节点相遇持续时间. 算法第 13 行,计算集合中节点相似率. 算法第 14 行,根据节点相似率来进行节点投递率的计算,在计算之前会根据节点相遇持续时间先更新节点衰减概率. 算法第 15 行,计算节点传递概率. 算法第 17 ~ 21 行,遍历节点转发列表,根据节点传递概率将消息转发给比当前节点概率大的节点. 由于本算法需要进行节点相遇持续时间的统计以及节点相似率计算,因此本算法时间主要耗费在 11 ~ 16 行,假设共有 n 个节点,则 S-Prophet 算法的时间复杂度为 $O(n^2)$.

3.3 ACK 删除机制

为了减少冗余副本在网络长时间保留造成节点缓存占用过多的不良影响,降低网络开销,本文将使用 ACK 删除机制来删除已经成功传递的消息副本. 给每个节点设置一个 ACK 列表,当消息成功传输到目的节点时,该消息就被加入到 ACK 列表中,当两个节点相遇时,交换它们的 ACK 列表,然后从缓存中删除对应的消息. 具体算法如表2所示,算法第

表 2 ACK 删除机制

Table 2 ACK deletion mechanism

算法 2. ACK 删除机制
输入: InitackMessage( ),消息 msg
输出: ackMessage
1. if(msg. getTo( ) == V_b)
2. V_a. ackMessage. add(msg. getId( ))
3. endif
4. if(节点 V_a 与节点 V_b 相遇)
5. V_b. ackMessage. addAll(V_a. ackMessage) ;
6. V_a. ackMessage. addAll(V_b. ackMessage) ;
7. V_b. deleteAckedMessage( ) ;
8. V_a. deleteAckedMessage( ) ;
9. endif

1 ~ 3行表示,当消息成功投递之后,判断消息是否已经到达目的节点,如果消息到达目的节点,则将消息添加到 ackMessage 列表中. 算法 4 ~ 9 行表示当任意两个节点 V_a 与节点 V_b 相遇,首先互相交换其 ackMessage 列表中的信息,然后删除 ackMessage 列表中已经被成功投递到目的节点的消息. 假设 ackMessage 列表中的消息共有 n 个,则本文的 ACK 删除机制时间复杂度为 O(n) .

4 仿真实验及结果分析

4.1 仿真的参数设置

本文的所有实验都是采用机会网络环境仿真平台 ONE (Opportunistic Network Environment) 来进行,本文对经典路由算法 Epidemic、Prophet 以及本文算法 S-Prophet 利用 ONE 仿真平台分别从节点缓存空间、消息生存周期、仿真时间、消息产生间隔 4 个不同角度对算法性能的影响进行比较分析,主要将消息投递率、传输延迟、平均跳数、网络开销作为路由算法的衡量指标,表 3 为本文仿真时设置的具体参数.

表 3 仿真参数设置

Table 3 Simulation parameter setting

参数名称	默认值	变化范围
仿真时间	12h	6-14h
场景大小	4500m × 3400m	—
移动模型	ShortestPathMapBased	—
消息大小	0. 5 ~ 1MB	—
消息生存周期	300min	100-300min
节点缓存大小	10MB	2-10MB
传输范围	10m	—
传输速度	250kbps	—
消息产生间隔	75s	55-75s
节点数量	126	—
时间周期间隔 T	7200s	—
P_{init}	0. 75	—
β	0. 25	—
γ	0. 98	—

4.2 仿真结果分析

4.2.1 节点缓存对算法性能的影响

图 4 是 3 种算法的路由性能随着节点缓存空间的变化情况,从图中可以看出本文提出的 S-Prophet 路由算法在投递率、网络开销和平均跳数及传输延迟等方面都取得了较好的仿真结果,从而证实了考虑节点间相似率在路由选择方面发挥的主观作用. 图4(a) 的仿真结果表明,当节点缓存为2MB

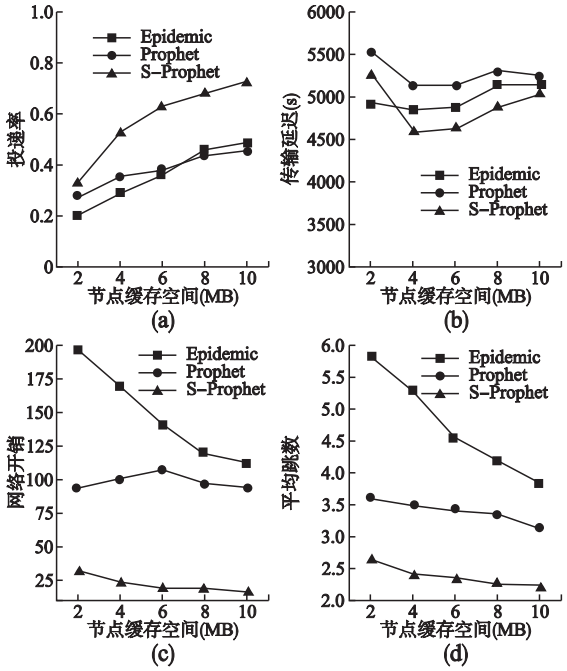

图 4 节点缓存对算法性能的影响

Fig. 4 Effect of node caching on algorithm performance

时,S-Prophet 算法比 Prophet 算法提高了 20%,比 Epidemic 算法提高了 66%. 当节点缓存为 10MB 时,S-Prophet 算法比 Prophet 算法提高了 58%,比 Epidemic 算法提高了 48%. 随着缓存空间的增加,Epidemic、Prophet、S-Prophet 算法的消息投递率一直在持续增长,由于缓存空间的增加,可携带的消息也增多,从而提高了总的消息成功投递率.

图 4(b) 中的仿真结果表明,当节点缓存从 2MB 变化到 4MB 时,3 种算法的传输延迟都有降低,特别是 S-Prophet 算法有明显的降低,这是因为在缓存为 2MB 时,S-Prophet 需要统计节点的相遇记录用于计算节点相似率,会因此占用一部分节点缓存,从而导致消息传输延迟会比较高. 当节点缓存从 4MB 开始增加到 10MB 的过程中,S-Prophet 算法传输延迟逐渐增加,但仍然比 Epidemic、Prophet 要低,这是因为在 S-Prophet 算法中利用了节点相似率改进了两个节点间传输概率更合理的决策出下一跳中继节点.

图 4(c) 的仿真结果表明,S-Prophet 路由算法在网络开销方面表现最优,S-Prophet 比 Epidemic 要小 83% ~85%,比 Prophet 要小 64% ~81%,这是因为在 S-Prophet 算法中不仅增加了节点相似率作为节点传输概率的影响因素,而且增加了 ACK 删除机制,有效的减少了网络中冗余副本的数量,从而降低了网络开销.

从图 4(d) 中可以看出,当缓存空间加大时,消息在网络中的平均传输跳数会随此降低,这是因为节点可存放的消息数量增多. 对于 Epidemic 越有利,对 S-Prophet 和 Prophet 算法的平均跳数影响并不大. 其中 S-Prophet 算法始终维持在 2 ~3跳之间,而 Prophet 算法始终维持在 3 ~4 跳之间. 跳数越少说明路由算法更有效,从图中可以直观看出 S-Prophet 算法性能略优于 Prophet,因为 S-Prophet 在选择下一跳的指标上利用到了节点相似率这一因素.

4.2.2　消息生存周期对算法性能的影响

图 5 描述了各个算法在不同消息生存周期下的路由表现. 从图 5(a) 中可以看出,随着消息生存周期从 100min 增加

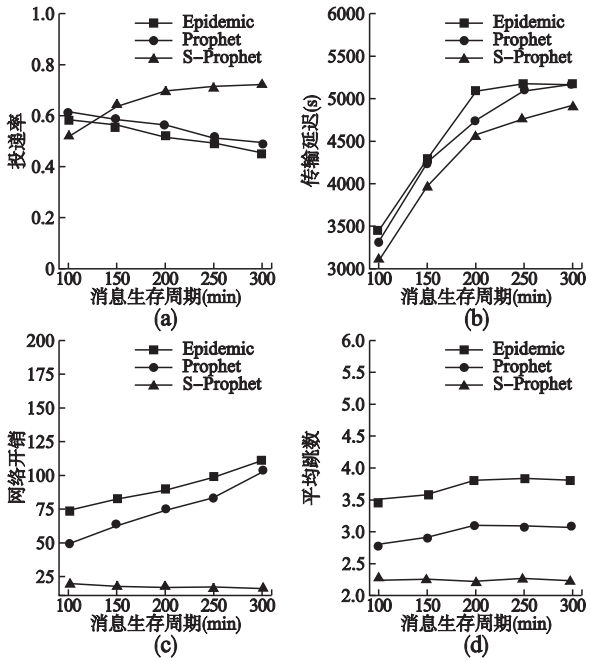

图 5　消息生存周期对算法性能的影响

Fig. 5　Effect of message lifetime on algorithm performance

到 300min 的过程中,Epidemic 算法和 Prophet 算法的投递率在降低,这是因为当节点缓存空间较小的情况下,如果消息生存周期比较高,会使大量已经成功投递的消息仍然存在网络中,这对于 Epidemic 和 Prophet 算法来说是非常不利的,Epidemic 和 Prophet 算法并没有采取有效的措施来控制消息冗余数量,不可避免的产生更多的消息副本,而 S-Prophet 算法的投递率在逐渐上升,且 S-Prophet 算法的投递率在消息生存周期大于 150min 之后一直比 Prophet 和 Epidemic 算法的投递率高,这是因为在 S-Prophet 算法中有 ACK 删除机制,一定程度上减少了冗余副本的数量,从而提高消息投递率. 此外,从图 5 可以看出,随着消息生存周期的增加,S-Prophet 算法仍然保持了它在传输延迟、平均跳数、网络开销上的优势,均低于 Prophet 算法和 Epidemic 算法.

4.2.3　仿真时间对算法性能的影响

图 6 描述了各个算法在不同仿真时间下的路由表现. 从图 6(a) 中可以看到,在随着仿真时间的增多,S-Peophet、Prophet 以及 Epidemic 算法的投递率也在逐渐的提高,但是当仿真时间为 8h 的时候,S-Prophet 算法已经达到饱和状态,本文所提出的 S-Prophet 算法相较于 Epidemic 算法和 Prophet 算法更为稳定,且 S-Prophet 算法的投递率一直高于其他两种算法. 在仿真时间为 6h 的时候,S-Prophet 算法比 Epidemic 算法的投递率高 53.2%,比 Prophet 算法的投递率高 65%,当仿真时间达到 8h 的时候,S-Prophet 算法比 Epidemic 算法投递率高 53.3%,比 Prophet 算法投递率高 74%.

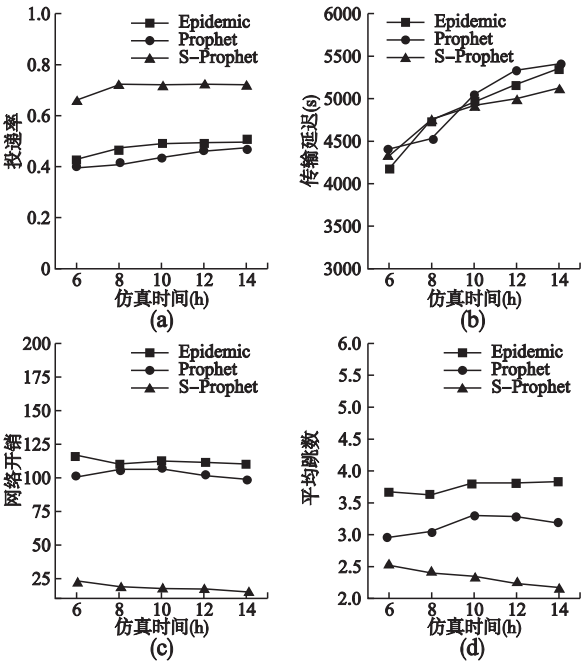

图 6　仿真时间对算法性能的影响

Fig. 6　Influence of simulation time on algorithm performance

从图 6(b) 中可以看出,当仿真时间在 6h 时候,S-Prophet 算法的传输延迟介于 Prophet 与 Epidemic 之间,当仿真时间从 10h 开始,S-Prophet 算法的传输延迟就一直低于 Prophet 与 Epidemic,这是由于随着仿真时间的增加,S-Prophet 算法收集到的节点信息越来越多,更能够准确的预估下一跳节点,将消息准确送达目的地.

从图 6(c) 中可以看出,随着仿真时间的增加,S-Prophet 算法的网络开销越来越低,且均低于 25,而 Prophet 算法和 Epi-

demic 算法两者都比较高，虽然 Prophet 算法网络开销比 Epidemic 的要稍微低一些，但是由于 Prophet 算法没有进行消息冗余控制，因而 S-Prophet 算法在路由性能的表现上更具优势.

从图 6(d) 中可以看出，随着仿真时间的增 S-Prophet 算法的平均跳数均在 3 以下，而 Prophet 算法与 Epidemic 算法的平均跳数均在 3 ~ 4 之间，平均跳数越少，说明路由性能越好.

4.2.4 消息产生间隔对算法性能的影响

图 7 比较了改变消息产生间隔时各个算法的 4 项评估指标. 在不考虑消息传输延迟的情况下，从图 7(a) 中可以看出，S-Prophet 算法的路由性能是最优的，且稳定的保持较高的消息投递率，而 Epidemic 和 Prophet 算法投递率都比较低，且两者都比较接近. S-Prophet 算法的投递率比 Epidemic 的高 53% ~57%，比 Prophet 的高 62% ~67%. 从图 7(b) 中可以看出，在传输延迟方面，S-Prophet 算法与 Prophet 相比，在消息产生间隔为 60s ~70s 的过程中两者较为接近，当消息产生间隔为 75s 时，S-Prophet 的传输延迟比 Prophet 要低，而此时 Prophet 的传输延迟比 Epidemic 高，就图 7(b) 图而言，总体来说 S-Prophet 的传输延迟始终比 Epidemic 要低.

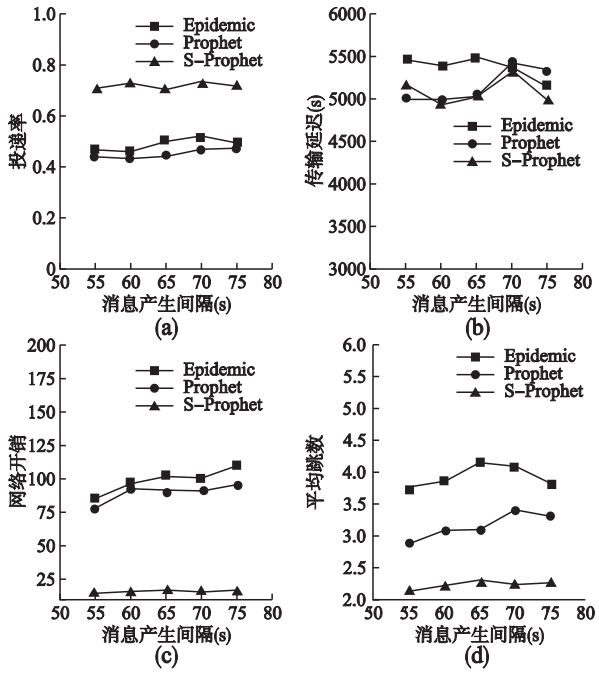

图 7 消息产生间隔对算法性能的影响

Fig. 7 Influence of message generation interval on algorithm performance

从图 7(c) 和图 7(d) 中可以看出，S-Prophet 算法的性能更加稳定，网络开销和平均跳数一直保持较低水平，其网络开销只有 Epidemic 的 17%、Prophet 的 19%. 导致这些路由性能差异的关键原因是，在消息生存时间间隔较小的情况下，整个网络会产生较多的消息，由于缺乏控制消息冗余的机制和受限的缓存空间，Epidemic 会丢弃大量的消息包，Prophet 利用节点间传输概率来筛选更有的中继节点，一定程度上比 Epidemic 的盲目性要好一些，因而路由性能上要由于 Epidemic，但仍然要弱于 S-Prophet. 因为 S-Prophet 借助节点相似率来设计节点传输概率，从而筛选出更好的中继节点，以此获得了更好的路由性能，并且通过 ACK 删除机制，有效避免过多的冗余消息副本，从而提高网络资源的利用率.

5 结束语

本文结合 Prophet 路由算法的优势，针对其中存在的问题，提出了一种考虑相遇节点历史连接情况以及 ACK 确认机制的路由算法，并进行了仿真实验. 实验证明，相对于传统的 Prophet 算法和 Epidemic 算法而言，本文提出的 S-Prophet 算法在路由性能上更好一些，但仍然存在一些不足的地方，比如整体而言，传输延迟均偏大，在这方面仍有较大的改进空间，这是下一步研究工作的重点.

References:

[1] Xiong Yong-ping, Sun Li-min, Niu Jian-wei, et al. Opportunistic networks [J]. Journal of Software, 2009, 20(1): 124-137.

[2] Trono E M, Fujimoto M, Suwa H, et al. Generating pedestrian maps of disaster areas through ad-hoc deployment of computing resources across a DTN [J]. Computer Communications, 2017, 100: 129-142.

[3] Cuka M, Shinko I, Spaho E, et al. A simulation system based on ONE and SUMO simulators: performance evaluation of different vehicular DTN routing protocols [J]. Journal of High Speed Networks, 2017, 23(1): 59-66.

[4] Tornell S M, Calafate C T, Cano J C, et al. DTN protocols for vehicular networks: An application oriented overview [J]. IEEE Communications Surveys & Tutorials, 2015, 17(2): 868-887.

[5] Zhang L, Zhou X, Guo J. Noncooperative dynamic routing with bandwidth constraint in intermittently connected deep space information networks under scheduled contacts [J]. Wireless Personal Communications, 2013, 68(4): 1255-1285.

[6] Lindgren A, Doria A, Schelen Oc. Probabilistic routing in intermittently connected networks [J]. Mobille Computing and Communications Review, 2003, 7(3): 19-26.

[7] Grossglauser M, Tse D. Mobility increases the capacity of Ad-Hoc wireless networks [J]. IEEE/ACM Transactions on Networking 10, 2002, 10(4): 477-486.

[8] Vahdat A, Becker D. Epidemic routing for partially-connected Ad-Hoc networks [D]. Durham, NC: Duke University, 2000: 1-14.

[9] Duan Zong-tao, Yang Yang, Fan Na, et al. Opportunistic forwarding algorithm based on connection time in probabilistic routing [J]. Microelectronics & Computer, 2018, 35(12): 50-54.

[10] Zhang Feng, Wang Xiao-ming, Zhang Shan-shan. Buffer aware prophet routing algorithm for opportunistic networks [J]. Computer Engineering and Design, 2015, 36(5): 1145-1149 + 1218.

[11] Ma Hui, Li Tao. Application of probabilistic routing based on throughput ratio in DTN [J]. Computer Technology and Development, 2018, 28(7): 187-191.

附中文参考文献:

[1] 熊永平，孙利民，牛建伟，等. 机会网络 [J]. 软件学报，2009，20(1)：124-137.

[9] 段宗涛，杨 阳，樊 娜，等. 概率路由中基于连接时间的机会转发算法 [J]. 微电子学与计算机，2018，35(12)：50-54.

[10] 张 峰，王小明，张珊珊. 机会网络中考虑缓存的 ProPhet 路由算法 [J]. 计算机工程与设计，2015，36(5)：1145-1149 + 1218.

[11] 马 慧，李 涛. 基于吞吐率的 Prophet 路由在 DTN 中的应用 [J]. 计算机技术与发展，2018，28(7)：187-191.