软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(1):153−163 [doi: 10.3724/SP.J.1001.2013.04218]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于接收阈值的容延网络拥塞控制机制*

赵广松, 陈 鸣

(解放军理工大学 指挥自动化学院,江苏 南京 210007)

通讯作者: 赵广松, E-mail: guangsongzhao@126.com

摘 要: 为了减少容延网络的资源开销,研究者提出了单副本转发路由算法.研究发现,这些转发算法导致节点流量负载极度不均衡,使得那些连接度较大的节点产生了拥塞.针对该问题,提出了一种基于接收阈值的拥塞控制机制,可以有效降低节点拥塞.该机制使每个 DTN(delay tolerant network)节点根据自身的拥塞状况动态调整自己的拥塞控制机制,而且该机制独立于节点所运行的转发路由算法,不影响路由算法对中继节点的选择,具有很好的普适性.为了验证所提出机制的有效性,将该拥塞控制机制与现有的 SimBet 路由算法加以结合,提出了具有拥塞控制功能的SimBetCC算法.实验结果表明,SimBetCC算法在取得很好的拥塞控制的前提下,其递交率和递交时延等性能方面均优于具有拥塞控制功能的 FairRoute 路由算法.

关键词: 容延网络;单副本;拥塞控制;中继节点;接收阈值

中图法分类号: TP393 **文献标识码**: A

中文引用格式: 赵广松,陈鸣.基于接收阈值的容延网络拥塞控制机制.软件学报,2013,24(1):153−163. http://www.jos.org.cn/1000-9825/4218.htm

英文引用格式: Zhao GS, Chen M. Congestion control mechanism based on accepting threshold in delay tolerant networks. Ruanjian Xuebao/Journal of Software, 2013,24(1):153−163 (in Chinese). http://www.jos.org.cn/1000-9825/4218.htm

Congestion Control Mechanism Based on Accepting Threshold in Delay Tolerant Networks

ZHAO Guang-Song, CHEN Ming

(Institute of Command Automation, PLA University of Science & Technology, Nanjing 210007, China)

Corresponding author: ZHAO Guang-Song, E-mail: guangsongzhao@126.com

Abstract: In order to decrease the resource cost in delay tolerant networks, single-copy forwarding protocols have been proposed by researchers. However, recent research has pointed out that single-copy forwarding protocols lead to great unfairness of the nodes' traffic load and make the highly connected nodes suffer congestion. In order to bridge this gap, a congestion control mechanism based on accepting the threshold is put forward in this paper, which makes each DTN node dynamically adjust to its accepting threshold and accordingly to its congestion state. Moreover, the mechanism can be widely adopted, since it is independent of the forwarding protocols and does not affect the choice of the relay nodes made by the forwarding protocol. In order to validate the proposed mechanism, an algorithm named SimBetCC is proposed, which incorporates the congestion control mechanism with existing SimBet protocol. Experimental results show that SimBetCC can cope with congestion and also outperforms FairRoute with congestion control in many aspects, e.g., the message delivery ratio, the delivery delay.

Key words: delay tolerant network; single-copy; congestion control; relay node; accepting threshold

当无线移动节点之间仅存在间歇性连接时,容延网络(delay tolerant network,简称 DTN)仍可以很好地支持节点间的数据传输.由于容延网络存在的潜在优势,该领域吸引了大量的研究工作.这些工作主要集中在 DTN

* 基金项目: 国家自然科学基金(61070173, 61103225); 国家重点基础研究发展计划(973)(2012CB315806); 江苏省自然科学基金(BK2010133)

收稿时间: 2011-09-20; 定稿时间: 2012-04-09

的路由算法设计方面,自从 Epidemic[1]路由算法被提出之后,大量的路由算法不断涌现出来.

从报文副本数目来看,DTN 路由算法可以分为两类:基于复制的路由算法(replication-based routing protocols)和基于转发的路由算法(forwarding-based routing protocols).

基于复制的路由算法通过向网络注入多个报文拷贝来提高报文的递交率和减少端到端的时延.它主要包括 Epidemic[1],PROPHET[2],Spray and Wait[3],RAPID[4],MaxProp[5]等路由算法.虽然这些算法在报文递交率和时延等方面取得了较优性能,但大量的报文拷贝充斥着整个 DTN 网络,使得本身资源并不丰富的 DTN 网络的大量资源被浪费掉.

针对该问题,人们提出了基于转发的路由算法.基于转发的路由算法在网络中为每个报文仅仅保留一个拷贝,并通过一定的启发式策略将报文转发给最优下一跳节点,最终将该报文递交给它的目的节点.基于转发的路由算法主要包括 SocialCast[6],SimBet[7]和 BubbleRap[8]等路由算法.这些算法利用社会网络模型和移动模型来识别最佳的中继节点,在控制网络资源开销的前提下,依然可以取得令人满意的性能.

近年来的研究[9−11]发现,基于转发的路由算法会导致大量的报文流向连接度较高的节点,导致节点负载不均衡,最终使那些连接度较高的节点发生拥塞而导致报文丢弃.例如在 SimBet 算法中,10%的节点承担了 54%的报文转发和 85%的报文递交任务.针对转发路由算法的不足,本文提出了一种基于接收阈值的拥塞控制机制.在该机制下,节点通过感知自己的拥塞程度,以一定的条件接纳到来的报文,并且随着拥塞程度的变化可以实时调整报文接纳条件.除了可以减轻节点拥塞程度,该机制还具有下面几个优点:

(1) 拥塞控制的粒度可以细化到每个节点上,节点可以根据自己的拥塞程度动态调整自身的拥塞控制;

(2) 该拥塞控制机制独立于转发路由算法,不影响路由算法对下一跳中继节点的选取,适用范围更广;

(3) 能有效减少网络资源的浪费.

本文第 1 节介绍当前 DTN 拥塞控制的相关工作.第 2 节详细介绍本文所提出的基于接收阈值拥塞控制的原理和具体设计.第 3 节进行仿真实验,对仿真结果进行分析.第 4 节进行总结.

1 相关工作

基于转发路由算法与基于复制路由算法存在很大差别,因此它们的拥塞控制手段差别也较大.

当前,基于复制路由算法的拥塞控制主要通过限制报文的副本数目来防止网络发生拥塞.基于复制的喷射等待(spray and wait)[3]和喷射聚焦(spray and focus)[3]路由协议都是在喷射阶段,将 L 个报文副本独立分发给 L 个中继节点,对报文的副本数目规定了上限 L.EBR[12]协议也是一种限制报文最大副本数目的路由算法,在 EBR 算法下,每个节点跟踪自身与其他节点发生接触的频繁程度 EV.当节点 A 想要传输报文给节点 B 时,它们通过 EV 值来计算 A 应该传输多少个报文副本给节点 B.与 Epidemic 算法盲目地向所有邻居复制报文副本不同,PROPHET[2]与 RAPID[4]等算法采用一定的启发式策略对报文进行复制,尽量减少报文副本数目以降低网络拥塞.PROPHET 算法通过估计邻居节点与目的节点相遇概率,将报文转交给具有更高递交概率的邻居节点.而 RAPID 是一种基于效用的路由协议,它在给定的有限带宽下,决定对哪些报文进行复制可以优化相应的路由目标.文献[13]设计了一种基于网络局部的方法来检测 DTN 网络拥塞,该方法通过考察报文丢失数与报文复制数之比这个测度来估计网络拥塞程度,然后根据网络拥塞状况动态调整报文副本数目的上限.文献[14]允许拥塞节点将自身缓存中的报文转移给它的一组邻居节点,这样可以避免拥塞节点发生缓存溢出.当这些拥塞节点的拥塞程度减轻之后,它再取回先前转移的报文.该方法的缺点在于,它仅仅暂时缓解了拥塞.

而基于转发路由算法的拥塞控制机制与具体的路由算法紧耦合在一起.在选择最佳下一跳中继节点时,它们通常把节点的拥塞程度融合到下一跳中继节点的选择测度中,从而避免选择拥塞节点作为下一跳中继,使流量绕开拥塞节点.因此,这种拥塞控制机制改变了转发路由算法对下一跳节点的选择.当前,专门针对基于转发路由算法的拥塞控制工作还不多.文献[9]提出了具有负载均衡的 FairRoute 算法.该算法在 interaction strength 测度的基础上又定义了 assortativeness 测度,只允许节点将报文转发给那些具有更短队列长度的节点,减轻了连接度较大节点的拥塞.Grundy 等人在文献[10,11]中通过考虑节点多方面的特性,组合了节点的社交效用值、可

用缓存效用值以及中心网络(ego network)效用值作为转发算法 Café 的核心,将报文转发给效用值最大的下一跳节点,从而将流量导向负载较轻的节点上,以减轻连接度较大节点的拥塞.

针对转发路由算法,本文提出的基于接收阈值的拥塞控制,使每个节点根据自身的拥塞状况动态调整自己的拥塞控制,控制粒度可以具体到每个拥塞节点上.与 FairRoute 算法和 Grundy 等人提出的 Café 算法的拥塞机制不同,基于接收阈值的拥塞控制独立于节点所运行的转发路由算法,它不是通过改变下一跳节点的选取来达到拥塞控制.因此,本文提出的拥塞控制机制并不影响转发路由算法对下一跳中继节点选取的决策,这使得该拥塞控制机制具有广泛的适用性.

2 本文的拥塞控制机制

2.1 基于接收阈值的拥塞控制原理

为了有效地设计基于转发路由算法的拥塞控制机制,首先必须认识到节点发生拥塞的本质原因.当前的转发算法都是采用启发式策略来选择最优下一跳节点,这使得大量报文被转发给那些连接度较大的节点,而这些连接度较大的节点却需要花费较长时间才能把接收的报文递交给目的节点或转发给其他中继节点.当报文接收速率远大于报文离开速率时,这些节点的剩余缓存空间不断减小,最终导致缓存溢出.这就形成了所谓的节点拥塞.因此,发生节点拥塞的本质原因在于:节点缓存的报文填充速率远大于报文的成功离开速率(报文被递交或转发的速率).

当节点 A 与节点 B 相接触时,节点 A 根据转发路由算法,拟将报文 m 转发给 B.为了防止 B 缓存溢出,本文设定了节点接收报文的条件:假设节点 B 在接收报文 m 后,B 在时间范围 C 内将报文 m 成功递交给目的节点或者转发给其他中继节点的概率为 P_C.只有在满足 $P_C \geqslant p_{Thres}$ 时,节点 B 才会接收报文 m;否则拒绝接收(其中,p_{Thres} 为节点 B 的接收阈值,它反映了节点 B 当前的拥塞程度).

通过对该接收条件的限制,我们可以保证节点 B 的报文接收速率与报文成功离开速率达到一种平衡,从而有效避免节点的拥塞.并且,每个节点可以根据自身拥塞程度的高低动态地上调或降低接收阈值 p_{Thres},以达到实时拥塞控制.本文称这种方法为基于接收阈值的拥塞控制.

可以看到,如果一个节点拥塞达到一定程度,由于接收条件的限制,它会拒绝接收某个报文 m.这使得报文 m 的发送节点只能将报文 m 保留在自己的缓存中,这导致该发送节点拥塞程度加重,该节点的拥塞控制策略也会拒绝接收其他报文.随着时间的推进,这种级联效应会使所有节点的缓存负载变得均匀.

在 SimBet 路由算法下,报文需要经过多跳中继节点才会到达连接度较大的节点,然后很可能在连接度较大的节点上被丢弃(由于连接度较大的节点缓存发生溢出),这使得前面用于转发该报文的资源都遭到浪费.而在基于接收阈值的拥塞控制下,只有当下一跳节点 n 在时间范围 C 内能以高于 p_{Thres} 的概率成功转发或递交该报文时,节点 n 才接收该报文.假设报文 m 在时间范围 C 内被节点 n 转发的概率 P_C 服从[0,1]均匀分布,那么报文 m 被节点 n 接收的概率 $P_{rec}=P(P_C \geqslant p_{Thres}^{(n)})=1-p_{Thres}^{(n)}$,$p_{Thres}^{(n)}$ 表示节点 n 的当前接收阈值.

由此可知,报文 m 能够成功经历 h 跳转发的概率 $P_h=\prod_{k=1}^{h}(1-p_{Thres}^{(k)})$,其中,$p_{Thres}^{(k)}$ 表示第 k 跳节点的接收阈值.随着 h 的增大,可以看出 P_h 急剧减小.尤其是当网络负载较重时,每个节点的 p_{Thres} 都很大,这时 P_n 就更小,最终报文更多地存储在源节点附近节点的缓存中.在这种情况下,如果报文发生丢弃,网络资源的浪费相比 SimBet 算法要小很多.

综上所述,基于接收阈值的拥塞控制的优点在于,它不仅可以动态地进行拥塞控制,而且不影响转发路由算法对中继节点的选择,同时还能有效缓解网络资源的浪费.

2.2 基于接收阈值的拥塞控制概述

2.2.1 节点交互过程

这里使用节点 A 与 B 的交互过程来阐述基于接收阈值的拥塞控制的工作过程.

(1) 通过邻居发现,节点 A 与 B 相互发现对方;

(2) 节点 A 和 B 将自身缓存中那些目的地址属于对方的报文递交给对方,然后,A 根据当前运行的路由算法,将自身缓存中那些可以转交给 B 的报文相关信息⟨ID,$DstID$,$ExpTime$,hop⟩放在一个列表 $List_1$ 中,传输给节点 B,其中,ID 表示报文的标识符,$DstID$ 表示报文的目的节点,$ExpTime$ 表示报文的有效期,hop 表示该报文已被传输的跳数(如图 1 中的 Step 1 所示);

(3) 节点 B 计算列表 $List_1$ 中每个报文被 B 接收后能在时间范围 C 内成功离开 B 的概率 P_C(被递交或被转发).若 $P_C \geqslant p_{Thres}$,那么就把该报文的相关信息⟨ID,P_C,hop⟩放到列表 $List_2$ 中;

(4) 节点 B 通过解决 0-1 背包问题,然后发送一个返回列表 $List_3$ 给节点 A,告知 A 应该向节点 B 转发哪些报文,以及按何种顺序转发;

(5) 当节点 A 和 B 交互结束后,节点 B 根据这次交互来更新自身的拥塞程度 $\overline{CV}$,然后重新调整接收阈值 p_{Thres}.

这里仅描述了节点 A 将缓存中的数据包转发给 B 的单方向交互过程,节点 B 也存在类似 A 的转发过程.在 DTN 网络中,一个节点可能同时与多个节点发生接触,本文这里仅仅假设一个节点同时只能与一个节点发生接触通信.

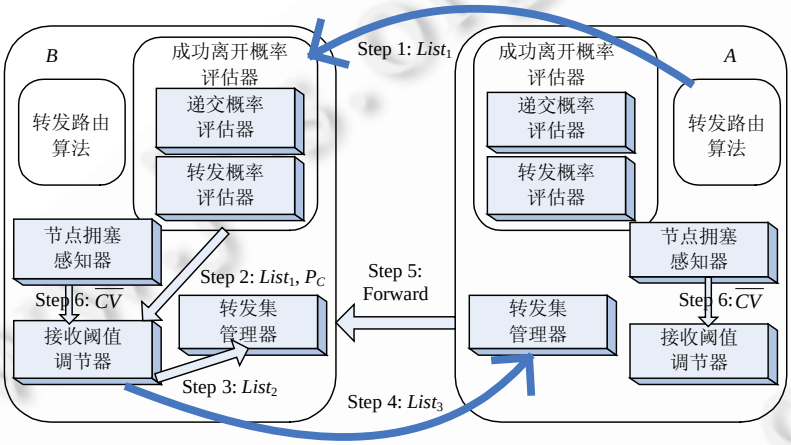


Fig.1 The architecture of DTN nodes based on accepting threshold

图 1 基于接收阈值拥塞控制的 DTN 节点架构

2.2.2 节点的体系架构

基于接收阈值拥塞控制的 DTN 节点架构如图 1 所示,两个 DTN 节点 A 与 B 发生接触.

下面介绍 DTN 节点中每个模块的功能及其它们之间的交互:

- 转发路由算法模块:运行具体的转发路由算法,例如 SimBet,BubbleRap 等转发路由算法;
- 成功离开概率评估器:用于计算列表 $List_1$ 中每个报文在时间段 C 内发生递交或转发的概率,然后给 $List_1$ 中的每个报文添加计算出的成功离开概率值 P_C.它包含两个组件:递交概率评估器和转发概率评估器(如图 1 中的 Step 2 所示);
- 节点拥塞感知器:它根据节点当前缓存的剩余空间和报文接收速率不断评估当前的拥塞程度 $\overline{CV}$, 然后将 $\overline{CV}$ 告知接收阈值调节器;
- 接收阈值调节器:根据节点拥塞感知器告知的拥塞程度 $\overline{CV}$ 动态调整自身的接收阈值 p_{Thres}(如图 1 中的 Step 6 所示).根据成功离开概率评估器计算出的 P_C,决定接收 $List_1$ 中的哪些报文,然后将其放到返回列表 $List_2$,返回给节点 A(如图 1 中的 Step 3 所示);
- 转发集管理器:节点 A 在接收到节点 B 发送过来的列表 $List_2$ 和 B 的剩余缓存空间 l_{free} 之后,通过求解一

个 0-1 背包问题,最终决定向节点 B 转发列表 $List_2$ 中哪些报文以及按何种顺序转发(发送一个返回列表 $List_3$ 给节点 A,如图 1 中的 Step 4 所示).

2.3 基于接收阈值的拥塞控制设计细节

2.3.1 成功离开概率的计算

这里讲解节点 B 在接收到节点 A 发送过来的列表 $List_1$ 后,如何计算列表 $List_1$ 中每个报文在时间范围 C 内被节点 B 递交或转发出去的概率 P_C.

报文 m 离开节点 B 缓存的情况有 4 种:(1) 被递交给它的目的节点;(2) 被转发给下一跳中继节点;(3) 由于缓存溢出被丢弃;(4) 由于报文超时被丢弃.

每个 DTN 节点为上面 4 种情况各自维护一个计数器,每当节点缓存中某个报文发生上面的情况时,对应的计数器就加 1.因此,当该节点接收某个报文 m 后,报文 m 发生上面任意一种情况的概率期望可以采用历史统计数据来近似:

$$p_d=\frac{n_d}{n_d+n_f+n_o+n_t},p_f=\frac{n_f}{n_d+n_f+n_o+n_t},p_o=\frac{n_o}{n_d+n_f+n_o+n_t},p_t=\frac{n_t}{n_d+n_f+n_o+n_t},$$

其中,p_d,p_f,p_o 和 p_t 分别表示递交概率、转发概率、溢出概率和超时概率,n_d,n_f,n_o 和 n_t 分别表示发生递交、转发、溢出和超时的报文数目.

报文 m 在时间范围 C 内成功离开节点 B 缓存的概率可记为 $P_C=1-p(\overline{Cd}\cap\overline{Cf})$, 其中,$Cd$ 表示报文 m 在时间范围 C 内被递交的事件,Cf 表示报文 m 在时间范围 C 内被转发的事件, $\overline{Cd}\cap\overline{Cf}$ 表示报文在时间范围 C 内既没有被递交也没有被转发的事件.本文将 $P_C=1-p(\overline{Cd}\cap\overline{Cf})$ 进行展开:

$$P_C=1-p(\overline{Cd}\cap\overline{Cf})=1-p(\overline{Cd\cup Cf})=1-[1-p(Cd\cup Cf)]=p(Cd\cup Cf)=p_{Cd}+p_{Cf}-p(Cd\cap Cf).$$

因为报文在时间范围 C 内被递交事件 Cd 与被转发事件 Cf 是互不相容的,因此 $p(Cd\cap Cf)=0$,则可推得 $P_C=p_{Cd}+p_{Cf}$,其中,p_{Cd} 表示在时间范围 C 内报文 m 被递交的概率,p_{Cf} 表示在时间范围 C 内报文 m 被转发的概率.

首先计算 p_{Cd}.报文 m 在时间范围 C 内发生递交,其实蕴含了两个条件:(1) 报文 m 是以递交的方式离开缓存的;(2) 在递交的前提下,报文 m 发生递交的时间是在时间范围 C 内.那么,

P(报文 m 在时间范围 C 内发生递交)=P(报文发生递交)×P(递交在时间范围 C 内发生|报文发生递交),

即,$p_{Cd}=p_d\times p_{dC}$.其中,p_{dC} 表示报文 m 递交在时间范围 C 内发生的概率(节点 B 与报文 m 的目的节点 D 下一次接触时间间隔 X 小于 C 的概率),则 $p_{dC}=p\{X\leqslant t+C-t_{contact}\}=1-p\{X>t+C-t_{contact}\}$.其中,$X$ 变量表示节点 B 与报文 m 的目的节点 D 接触时间间隔,t 表示当前时刻,$t_{contact}$ 表示节点 B 与节点 D 的最近一次的接触时刻.

那么,根据马尔可夫不等式 $p\{X>t+C-t_{contact}\}\leqslant\frac{E(X)}{t+C-t_{contact}}$ 可推得:

$$p_{dC}=1-p\{X>t+C-t_{contact}\}\geqslant 1-\frac{E(X)}{t+C-t_{contact}},$$

其中,$E(X)$可以用节点 B 和目的节点 D 之间的平均历史接触间隔来近似.

同理,可计算 p_{Cf}.

P(报文 m 在时间范围 C 内发生转发)=P(报文发生转发)×P(转发在时间范围 C 内发生|报文发生转发),

即,$p_{Cf}=p_f\times p_{fC}$,p_{fC} 表示报文转发在时间范围 C 内发生的概率.

这里使用 $p_{fC}\approx n_{fC}/n_f$ 来近似 p_{fC},n_{fC} 表示节点 B 所有的接收报文中那些接收时间和转发时间的间隔小于 C 的报文数目.这就要求节点在每次进行报文转发时,都要计算该报文的到达时间与转发时间的间隔 $t_{forward}-t_{arrive}$.若 $t_{forward}-t_{arrive}<C$,则 n_{fC} 加 1,否则保持不变.

那么,该报文在时间范围 C 内成功离开节点 B 的概率为

$$\begin{aligned}P_C &= p_{Cd} + p_{Cf}\\ &= p_d \times p_{dC} + p_f \times p_{fC}\\ &= \frac{n_d}{n_d + n_f + n_o + n_t} \times (1 - p\{X > t + C - t_{contact}\}) + \frac{n_f}{n_d + n_f + n_o + n_t} \times \frac{n_{fC}}{n_f}\\ &\geqslant \frac{n_d}{n_d + n_f + n_o + n_t} \times \left(1 - \frac{E(X)}{t + C - t_{contact}}\right) + \frac{n_f}{n_d + n_f + n_o + n_t} \times \frac{n_{fC}}{n_f}.\end{aligned}$$

这里取 P_C 的下限,近似记为 $P_C = \frac{n_d}{n_d + n_f + n_o + n_t} \times \left(1 - \frac{E(X)}{t + C - t_{contact}}\right) + \frac{n_f}{n_d + n_f + n_o + n_t} \times \frac{n_{fC}}{n_f}$.

节点 B 的成功离开概率评估器计算列表 $List_1$ 中每个报文的离开概率 P_C,然后接收阈值调节器将 P_C 与自身当前接收阈值 p_{Thres} 进行比较.若 $P_C \geqslant p_{Thres}$,那么就把该报文 ID 以及对应的离开概率 P_C 放到返回列表 $List_2$ 中.最后,将返回列表 $List_2$ 发送给转发集管理器.

2.3.2 转发集合的优化

前面讲到,当接收到接收阈值调节器发送过来的列表 $List_2$ 后,节点 B 的转发集管理器需要决定节点 A 应该转发哪些报文以及按什么样的顺序转发,最终将结果(列表 $List_3$)发送给节点 A.

首先本文定义报文 m 转发效益值:U_m=hop×P_C,其中,hop 表示报文 m 已经被转发的跳数(可以根据报文的 TTL 获得),P_C 是指报文 m 在时间范围 C 内成功离开节点 B 缓存的概率.

例如,节点 A 缓存中存在两个报文 m_1 和 m_2,并且 m_1 和 m_2 都在列表 $List_2$ 中.m_1 已被转发跳数为 5,节点 B 在时间范围 C 内转发和递交 m_1 的概率 P_C 为 0.8;m_2 已被转发跳数为 3,节点 B 在时间范围 C 内转发和递交 m_2 的概率 P_C 为 0.9. $U_{m_1} = 5 \times 0.8 = 4.0, U_{m_2} = 3 \times 0.9 = 2.7$, 根据 $U_{m_1} > U_{m_2}$,此时 A 应优先转发 m_1 给 B.可以看到,在 P_C 相同的情况下,转发跳数多的报文优先被转发,从而减小这些报文被丢弃的概率,减少网络资源浪费.

节点 B 通过选择存储列表 $List_2$ 中以及自身缓存中的报文使得报文转发效益值达到最大化.本文将该问题形式化为 0-1 背包问题.

$$\begin{aligned}&\max \sum_{k=1}^{n} U_k x_k\\ &\text{s.t.} \quad \sum_{k=1}^{n} l_k x_k \leqslant l\\ &\qquad x_k \in \{0,1\}, \quad k = 1,2,...,n,\end{aligned}$$

其中,

- U_k 表示第 k 个报文转发效益值;
- l 是节点 B 的缓存空间大小;
- n 表示列表 $List_2$ 以及节点 B 缓存中所有报文的总数目;
- l_k 表示第 k 个报文的长度;
- x_k 表示是否选择转发第 k 个报文,若 x_k=1,则表示转发第 k 个报文;若 x_k=0,则表示不转发第 k 个报文.

0-1 背包问题是典型的 NP 难问题[15],本文采用启发式算法进行求解.节点中的转发集管理器采用的启发式算法设计如下:

(1) 节点 B 的转发集管理器计算列表 $List_2$ 中以及自身缓存中每个报文 m_k 的转发效益值:$U_k = hop_k \times P_C^k$;

(2) 按照 U_k/l_k 对每个报文进行降序排序;

(3) 按照这个顺序逐个查看每个报文 m_k.如果 m_k 属于列表 $List_2$,则将该报文的 ID 放到返回列表 $List_3$ 中,直到发生以下情况才终止:① 节点 B 的缓存不能再容纳下一个报文;② $List_2$ 中所有报文已全部被查看.

2.3.3 接收阈值的动态调整

基于接收阈值的拥塞控制机制允许每个节点根据自身拥塞程度的不同,动态地调整自己的报文接收阈值.本节具体阐述节点 B 在离开节点 A 时,节点 B 如何根据自身拥塞程度动态调整接收阈值 p_{Thres}.

如图 2 所示,t_1 表示节点 B 离开上一个接触节点的时刻;t'表示节点 A 与节点 B 开始接触的时刻;t_2 表示节点 B 离开节点 A 的时刻;$[t',t_2]$表示节点 A 和 B 相接触的时间段,本文认为这个时间段长度很小,相对于 C 可以忽略不计(C 的取值通常以天为单位,而接触时间通常以分钟为单位).

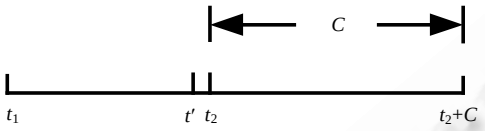


Fig.2 The time coordinate

图 2 时间坐标示意

$r=l_{sum}/\Delta t$ 表示节点 B 在时间段$[t_1,t_2]$的平均接收速率,其中,l_{sum} 表示节点 A 转交给节点 B 的所有报文的长度总和;Δt 表示相邻两次接触间隔,即$\Delta t=t_2-t_1$.

当节点 B 在时刻 t_2 离开节点 A 时,节点 B 的拥塞程度 CV 可以表示为 $CV=r/l_{free}$,其中,l_{free} 表示节点 B 在 t_2 时刻的剩余缓存空间.可以看出,CV 越大,节点的拥塞程度也就越高.

为了抑制拥塞突发峰值的效应,本文采用指数加权移动平均(EWMA)来更新节点的拥塞程度 $\overline{CV}$:

$$\overline{CV}_{new}=\alpha\cdot CV_{sample}+(1-\alpha)\cdot\overline{CV}_{old},\alpha=0.25.$$

每当节点 B 与其他节点发生接触后,节点 B 就计算当前的拥塞程度 $CV=r/l_{free}$,并将该值作为 CV_{sample},对自己保存的 $\overline{CV}_{old}$ 进行更新,获得新的 $\overline{CV}_{new}$. 该工作由节点拥塞感知器来完成.

那么节点 B 如何根据自身拥塞程度值 $\overline{CV}_{new}$ 来调整接收阈值 p_{Thres} 呢?在不破坏报文接收条件的情况下,本文考虑所存在的最坏情况:节点 B 在离开 A 之后(在 t_2 之后),其缓存中所有报文在时间段$[t_2,t_2+C]$内都没有离开.如果在该时间段内,节点 B 的接收速率 r 保持不变,要使节点 B 在时间段$[t_2,t_2+C]$不会发生缓存溢出,则必须满足 $r\cdot C\leqslant l_{free}$,即需要满足:$CV=r/l_{free}\leqslant 1/C$.

若 $CV\geqslant 1/C$,则节点 B 在时间段$[t_2,t_2+C]$内会发生缓存溢出,那么需要提高节点 B 的接收阈值来降低节点 B 在时间段$[t_2,t_2+C]$内的报文接收速率 r,以防止缓存溢出;若 $CV<1/C$,则可以降低节点 B 的接收阈值来提高报文的接收速率.本文采用类似 TCP 拥塞控制策略,对接收阈值进行“加性增,乘性减”,从而实现报文接收速率的调整.该工作由节点的接收阈值调节器来完成.

- 当 $\overline{CV}_{new}>\overline{CV}_{old}$ 且 $\overline{CV}_{new}\geqslant 1/C$ 时,则 $p_{Thres}=p_{Thres}\times mi$($mi>1$,增大阈值来降低报文接收速率);
- 当 $\overline{CV}_{new}\leqslant\overline{CV}_{old}$ 且 $\overline{CV}_{new}\geqslant 1/C$ 时,则 p_{Thres} 保持不变;
- 当 $\overline{CV}_{new}<\overline{CV}_{old}<1/C$ 时,则 $p_{Thres}=p_{Thres}-ad$(降低阈值来提高报文接收速率);
- 当 $\overline{CV}_{old}\leqslant\overline{CV}_{new}<1/C$ 时,则 p_{Thres} 保持不变.

3 实验仿真评估

3.1 实验的设置

本文仿真评估采用 MIT Reality Trace[16].在这个数据集中,97 个 Nokia 6600 手机终端被分发给 MIT 的学生和教员,数据采集时间长达 9 个月.每当两个手机终端发生接触时,就会产生一条接触记录.该记录包含了发生接触的两个节点标识符、接触开始时间和持续时间.整个 Trace 包含了 110K 条记录.

在仿真过程中,每个节点每天随机产生一个数据包,并且数据包的目的地址是随机的.数据包的大小在[50KB,100KB]区间服从均匀分布.并且每个数据包都附有 TTL 值,一旦 TTL 过期,该数据包就要从节点缓存中移

除.本文假定每个 DTN 节点的缓存空间为 5MB.在仿真过程中,Trace 数据集的前 1/3 用于预热,仿真结果从剩余数据集的仿真中获取.

在仿真中,本文对原始 SimBet 路由算法、FairRoute 路由算法和附加上基于接收阈值拥塞控制的 SimBet 算法(SimBet with congestion control,简称 SimBetCC)进行性能对比.下面对这几种转发路由算法和参数设置进行介绍.

3.2 算法介绍与参数设置

(1) SimBet 路由算法:该路由算法结合了分布式的 *betweenness* 测度和 *similarity* 测度,为每个节点定义了新的测度 *SimBetUtil*(*SimBetUtil*=α*SimUtil*+β*BetUtil*,α+β=1).在路由过程中,数据包被转发给具有更高 *SimBetUtil* 的节点.本文在仿真过程中设置α+β=0.5.

(2) FairRoute 路由算法:该算法使用"*interaction strength*"测度和"*assortativeness*"测度来达到流量均衡,降低连接度较大节点的拥塞.该算法只允许节点将报文转发给那些具有更短队列长度的节点.本文在仿真过程中将算法参数设置为 r_λ=5×10$^{-4}h^{-1}$,r_σ=5×10$^{-3}h^{-1}$,r_λ与 r_σ分别表示节点间长期交互强度和短期交互强度的指数衰减速度.

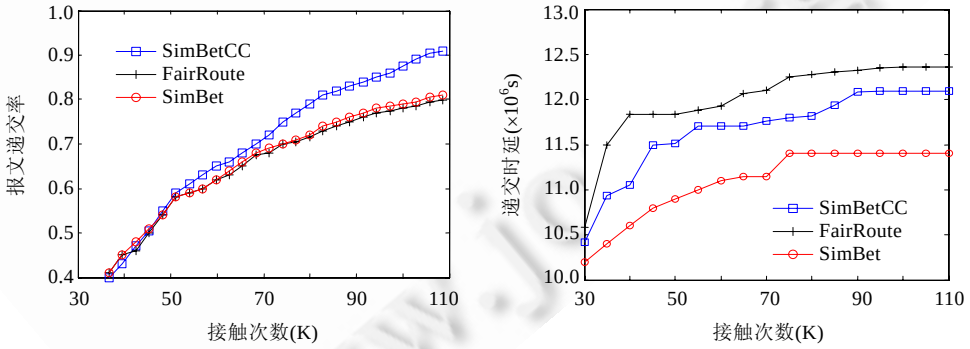
(3) SimBetCC 路由算法:就是在 SimBet 路由算法基础上,增加了基于接收阈值的拥塞控制机制.其中,拥塞控制参数设置为 *mi*=1.1,*ad*=0.1,p_{Thres} 初值为 0.8,*C*=2 天.

3.3 仿真结果与分析

(1) 报文递交率和递交时延

如图 3(a)所示,横轴表示仿真接触次数,纵轴表示报文的递交率.这里将每个数据报的 *TTL* 设置为 9 个月,即不会发生报文超时.随着仿真的不断推进,SimBetCC 的递交率上升得最快.仿真结束时,SimBetCC 的报文递交率达到了 91%.而 FairRoute 和 SimBet 的报文递交率分别为 80%和 81%.

原因分析:由于 FairRoute 只允许节点将报文递交给队列长度比其更短的节点,这导致很多报文被递交给那些连接度较低的节点.这些残留在连接度较低的节点缓存中的报文在很长时间内难以递交.而 SimBet 递交率较低的原因主要是连接度较大的节点缓存溢出,导致大量报文丢失.如本文前面提到的,SimBetCC 算法没有改变 SimBet 路由算法对下一跳节点的选择,报文依然被递交给 *SimBetUtil* 更高的节点,这有利于报文的迅速递交.因此对于相同仿真时刻,SimBetCC 的递交率明显高于 FairRoute.同时,SimBetCC 通过接收阈值的拥塞控制减少了拥塞节点的缓存溢出,大大提高了报文递交率.



(a) 各种算法的报文递交率 (b) 各种算法的报文递交时延

Fig.3

图 3

从图 3(b)看到,SimBet 算法的报文递交时延最小,SimBetCC 次之,而在 FairRoute 下的报文递交时延却最大.

原因在于,SimBet 算法始终选择 *SimBetUtil* 更高的节点来作为报文的下一跳中继,加快了报文的递交速度.SimBetCC 的基于接收阈值的拥塞控制机制抑制了部分报文的转发,这在一定程度上延长了报文递交时延.而 FairRoute 为了进行拥塞控制和负载均衡,它只允许将报文递交给队列长度更短的节点(连接度较低的节点),这使得报文需要花费很长时间才得以递交给目的节点.因此,FairRoute 的递交时延最大.

(2) 转发效率

定义. 转发效率=递交报文数/总的转发次数.

如图 4 所示,横轴表示节点接触次数,纵轴表示转发效率.我们将报文的 *TTL* 设置为 3 个月.从仿真结果可以看出,SimBetCC 的转发效率最高,平均值为 0.25.也就是说,报文平均被转发 4 次被递交给目的节点.而 FairRoute 的转发效率次之,大小为 0.23 左右.而 SimBet 的转发效率最低,仅为 0.17.SimBetCC 和 FairRoute 的转发效率高于 SimBet 的原因在于,它们降低了拥塞,减少了报文的丢失.因为丢失的报文只增加系统的总转发次数,却并不增加报文的递交数.

在网络负载加重的情况下,基于接收阈值的拥塞控制可以将报文的丢失压制在报文源节点附近.并且,在对转发集合进行优化时,本文根据转发效益值对报文进行排序,这使得被转发跳数多的报文尽量优先发送,这也在一定程度上使得以前的转发得到保证.因此,这些措施都提高了 SimBetCC 的转发效率.

(3) 被丢弃数据包的平均转发跳数

为了考察基于接收阈值的拥塞控制机制能否有效降低资源浪费,本文统计了在整个仿真过程中被丢弃报文的平均转发次数.从图 5 中可以看到,SimBetCC 算法下被丢数据包的转发次数平均为 2.1 次,而 FairRoute 平均次数为 3.2 次,SimBet 算法下被丢数据包平均转发次数却高达 6.3 次.

原因分析:在 SimBet 算法下,报文丢弃主要是因为连接度较大节点缓存溢出引起的(高连接度节点通常负载较大),报文丢弃很少发生在低连接度节点上.而报文通常要经过多跳才能转发给连接度较大的节点.因此,被丢弃报文的平均转发跳数很大.而在 SimBetCC 路由下,被丢弃的报文转发次数很小,比 FairRoute 还小 1 跳.这主要归结于本文的拥塞控制机制,使得报文丢弃主要发生在源节点附近.另外,转发集合的优化也在某种程度上降低了被丢弃的报文转发次数.

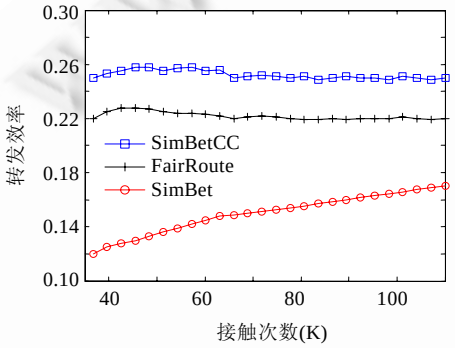


Fig.4 Forward efficiency

图 4 各种算法的转发效率

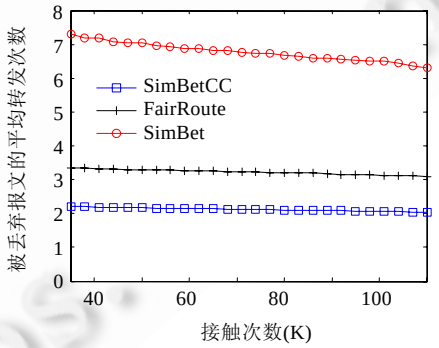


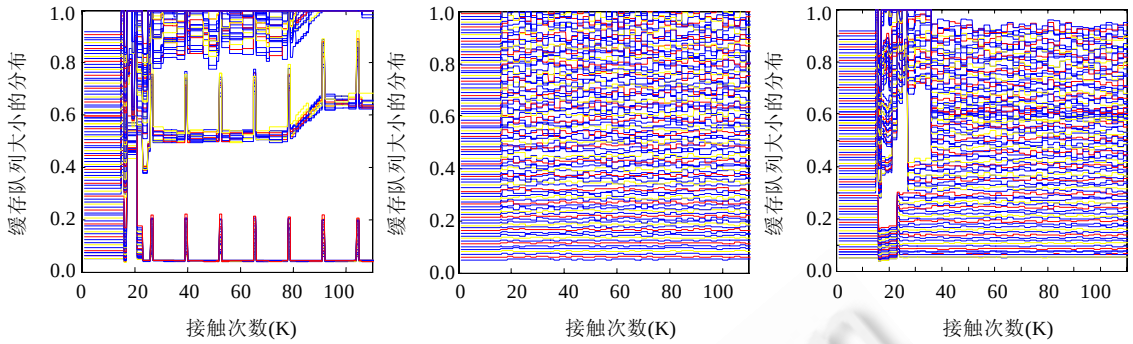
Fig.5 Average forward hops of the messages dropped

图 5 被丢弃数据包的平均转发跳数

(4) 在各种路由算法下,各个节点的缓存队列长度分布

为了观察整个仿真过程中各节点的缓存队列长度的变化,本文统计了在整个 trace 过程中,每个节点缓存队列中的报文占整个网络中处于转发的报文的比例.图 6(a)~图 6(c)中两条相邻线之间的空白部分表示对应节点的缓存报文所占比例.所有节点的报文比例之和为 1.

在 Trace 前 15K 次的接触中,每个节点自己不断产生数据包.在这段时期,节点之间不进行报文的转发.经过 15K 次接触之后,节点间开始进行报文转发.每个节点的缓存队列长度分布随时间演化的情况如图 6 所示.



(a) SimBet 算法 (b) FairRoute 算法 (c) SimBetCC 算法

Fig.6 The queue length of each node's buffer

图 6 各节点的缓存队列

从图 6(a)~图 6(c)中可以看到,在[0,15K]仿真过程中,由于不进行报文的转发,每个节点的缓存队列分布相等.随着后面转发过程的进行,图 6(a)显示,在 SimBet 路由算法下,节点的缓存队列分布极端不平衡,少量几个节点的缓存中的报文占到网络中所有报文的 30%~40%,而绝大部分节点缓存中的报文仅占了不到 1%.而图 6(b)显示,在 FairRoute 算法下节点的缓存队列分布非常均衡,每个节点的队列长度基本相等.而图 6(c)显示,在 SimBetCC 算法下,在转发的初始阶段[15K,28K],少量几个节点出现缓存队列较长的现象.但是随着在后面转发过程中,节点的缓存队列分布变得非常均衡.这是因为在初始阶段,连接度较大节点的初始接收阈值较小,SimBetCC 算法仍然会将大量报文转交给连接度较大的节点,从而导致这些节点缓存队列不断变长.随着它们剩余缓存空间的减少,其拥塞程度 CV 不断增大,从而促使它们的接收阈值 p_{Thres} 逐渐提高.这时,连接度较大的节点会大量地拒绝接收报文.由于级联效应,所有节点的缓存队列分布变得均匀.从实验结果可以看到,SimBetCC 算法的拥塞控制与负载均衡的效果与 FairRoute 算法相类似.

4 总 结

针对转发路由算法,本文提出了一种基于接收阈值的拥塞控制,该机制可以使每个节点根据自身的拥塞状况动态调整自己的拥塞控制.本文的拥塞控制独立于节点的转发路由算法,不影响路由算法对下一跳中继节点的选择.而 FairRoute 以及 Café 路由算法都是对中继节点的选择施加控制来降低拥塞,这在很大程度上影响了网络性能,增大了递交时延.

综上所述,本文的拥塞控制机制在降低节点拥塞的前提下,对网络性能影响较小,并且降低了网络资源的浪费程度.同时,该机制对转发路由算法具有很好的普适性.

针对基于复制的路由算法的拥塞控制,也是一项重要的研究内容.本文下一步的工作就是对我们提出的基于接收阈值的拥塞控制机制进行扩展,将其应用到基于复制的路由算法上.

References:

[1] Vahdat A, Becker D. Epidemic routing for partially connected ad hoc networks. Technical Report, CS-2000-06, Duke University, 2000.

[2] Lindgren A, Doria A, Schelen O. Probabilistic routing in intermittently connected networks. SIGMOBILE Mobile Computing Communications Review, 2003,7(3):19–20. [doi: 10.1145/961268]

[3] Spyropoulos T, Psounis K, Raghavendra CS. Efficient routing in intermittently connected mobile networks: The multiple-copy case. IEEE/ACM Trans. on Network, 2008,16(1):77–90. [doi: 10.1109/TNET.2007.897964]

[4] Balasubramanian A, Levine BN, Venkataramani A. DTN routing as a resource allocation problem. In: Proc. of the ACM SIGCOMM 2007. Kyoto: ACM Press, 2007. 373–384. [doi: 10.1145/1282380.1282422]

[5] Burgess J, Gallagher B, Jensen D, Levine BN. MaxProp: Routing for vehicle-based disruption-tolerant networks. In: Proc. of the IEEE INFOCOM 2006. Barcelona: IEEE Communications Society, 2006. 1−11. [doi: 10.1109/INFOCOM.2006.228]

[6] Costa P, Mascolo C, Musolesi M, Picco GP. Socially-Aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. IEEE Journal of Selected Areas in Communication, 2008,26(5):748−760. [doi: 10.1109/JSAC.2008.080602]

[7] Daly E, Haahr M. Social network analysis for routing in disconnected delay-tolerant MANETs. In: Proc. of the MobiHoc 2007. Montreal: ACM Press, 2007. 32−40. [doi: 10.1145/1288107.1288113]

[8] Pan H, Crowcroft JEY. BUBBLE rap: Social-Based forwarding in delay tolerant networks. In: Proc. of the MobiHoc 2008. Hong Kong: ACM Press, 2008. 241−250. [doi: 10.1109/TMC.2010.246]

[9] Pujol J, Toledo A, Rodriguez P. Fair routing in delay tolerant networks. In: Proc. of the IEEE INFOCOM 2009. Brazil, 2009. 837−845. [doi: 10.1109/INFCOM.2009.5061993]

[10] Grundy A, Radenkovic M. Promoting congestion control in opportunistic networks. In: Proc. of the IEEE WiMob 2010. Niagara Falls, 2010. 324−330. [doi: 10.1109/WIMOB.2010.5645048]

[11] Radenkovic M, Grundy A. Congestion aware forwarding in delay tolerant and social opportunistic networks. In: Proc. of the WONS 2011. Bardoneccia, 2011. 60−67. [doi: 10.1109/WONS.2011.5720201]

[12] Nelson SC, Bakht M, Kravets R, HarrisIII S. Encounter-Based routing in DTNs. Mobile Computing and Communications Review, 2009,13(1):56−59. [doi: 10.1145/1558590]

[13] Thompson N, Kravets R. Poster abstract: Understanding and controlling congestion in delay tolerant networks. ACM SIGMOBILE Mobile Computing and Communications Review, 2009,13(3):42−45. [doi: 10.1145/1710130.1710139]

[14] Seligman M, Fall K, Mundur P. Storage routing for DTN congestion control. Wireless Communications and Mobile Computing, 2007,7(10):1183−1196. [doi: 10.1002/wcm.v7:10]

[15] Dawande M, Kalagnanam J, Keskinocak P, Ravi R, Salman F. Approximation algorithms for the multiple knapsack problem with assignment restrictions. Journal of Combinatorial Optimization, 2000,4:171−186. [doi: 10.1023/A:1009894503716]

[16] Eagle N, Pentland A. Reality mining: Sensing complex social systems. Personal and Ubiquitous Computing, 2006,10(4):255−268. [doi: 10.1007/s00779-005-0046-3]

赵广松(1984－),男,江苏盐城人,博士生,主要研究领域为网络性能建模,时延容忍网络.

E-mail: guangsongzhao@126.com

陈鸣(1956－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络测量,网络性能分析与建模.

E-mail: mingchen@public1.ptt.js.cn