# 二进制指数退避的Gossip算法研究

成卫青*①② 张 蕾①
①(南京邮电大学计算机学院 南京 210023)
②(东南大学计算机网络和信息集成教育部重点实验室 南京 211189)

**摘 要**：为减少Gossip算法进行信息传播的通信开销，该文提出一个将二进制指数退避算法与经典Gossip算法相结合的二进制指数退避的Gossip算法(BEBG)，其信息传播策略是一个节点收到同一信息的次数越多，继续传播该信息的概率就越低。理论分析与仿真实验表明，BEBG能够有效减少信息传播冗余，网络中有$10^4$个节点时比经典Gossip算法减少了约61%网络负载。为解决BEBG存在的边缘节点问题，进一步提出了两个BEBG改进算法，引入Pull的PBEBG和引入向邻居节点Push的NBEBG。实验结果表明，两个算法能够消除边缘节点，当网络中有$10^4$个节点时，它们与相应的分别引入相同Pull和Push的经典Gossip算法相比，分别减少了约34%和37%的网络负载。

**关键词**：分布式系统；信息传播；Gossip算法
**中图分类号**：TP391; TP393 **文献标识码**：A **文章编号**：1009-5896(2021)12-3486-10
**DOI**: 10.11999/JEIT200094

# Research on Gossip Algorithms with Binary Exponential Backoff

CHENG Weiqing①② ZHANG Lei①
①(*School of Computer*, *Nanjing University of Posts and Telecommunications*, *Nanjing* 210023, *China*)
②(*Key Laboratory of Computer Network and Information Integration* (*Ministry of Education*), *Southeast University*, *Nanjing* 211189, *China*)

**Abstract**: In order to reduce the communication cost of classic Gossip algorithm for information dissemination, an improved Gossip algorithm BEBG (Gossip with Binary Exponential Backoff) is proposed, which combines the binary exponential backoff algorithm with Gossip algorithm. Its information dissemination strategy is that the more times that a node has received the same information, the lower probability it continues to spread the information. Theoretical analysis and simulation results show that the BEBG can effectively reduce the redundancy of information propagation, and compared with the classic Gossip algorithm, the network load is reduced by about 61% when there are $10^4$ nodes in the network. In order to solve the problem of edge nodes in the BEBG, two improved BEBG algorithms PBEBG that introduces Pull operations and NBEBG that introduces pushing information to a Neighbor node are further proposed. Experimental results show that the two algorithms can eliminate the edge nodes, and when there are $10^4$ nodes in the network, they reduce the network load by about 34% and 37% respectively compared with the corresponding improved classic Gossip algorithms which introduce the same pull and push respectively.

**Key words**: Distributed systems; Information dissemination; Gossip algorithm

## 1 引言

Gossip算法是一个依靠感染行为传播信息的算法，具有简单、高效、健壮、可扩展性和抗干扰能力强的特点，应用场景十分广泛。大型分布式系统常使用Gossip算法进行信息扩散，随着对等网络的发展，其应用领域不断扩大。Gossip算法的应用有数据库复制[1]、聚合计算[2]、网络拓扑构造[3,4]、数据挖掘[5]、故障检测[6]、网络监控[7]等。

Gossip算法常用于聚合计算和达成共识。文献[8]提出了一种新的基于Gossip算法的聚合计算方案，在$O(\log_2 n)$轮通信中发送$O(n\log_2 n)$的消息。文献[9]提出了一种局部平均信息交换算法来计算网络中每个节点的初始测量值的全局一致性；在每一

收稿日期：2020-02-11；改回日期：2021-03-23；网络出版：2021-06-03
*通信作者： 成卫青 chengweiq@njupt.edu.cn
基金项目：国家自然科学基金(61170322)，江苏省研究生教育教学改革课题(JGZZ19_038)

轮扩散过程中，每个节点与所有相邻节点交流，计算和交换局部平均值，使所有节点都能很快地以分布式方式逐渐达到全局一致。针对现有共识算法对拜占庭节点的容错能力低且对区块链系统可扩展性差的问题，文献[10]提出了基于Gossip的拜占庭共识算法，使系统可以容忍小于一半的节点为拜占庭节点。文献[11]提出了一种在不可靠网络中估计未知参数的异步广播Gossip算法，该算法允许新的传感器加入，旧的传感器离开，并能容忍链路故障。

Gossip算法可用于网络拓扑构造。如文献[4]提出了一个支持多个交互参与者和大量消费者的自组织流媒体解决方案，该方案利用基于Gossip的覆盖网的可伸缩性和健壮性来构建延迟感知的流播树，以便向多个用户提供多个不同的流。Gossip算法可用于数据挖掘。例如文献[5]针对淘宝等大型推荐系统中冷启动和稀疏评价严重降低推荐性能的问题，提出了一种基于圆桌流言算法的稀疏信任挖掘方法。文献[12]提出了基于Gossip的非结构化P2P网络分布式聚类算法，不依赖中央服务器来执行数据聚类任务，也不需要同步操作。文献[13]提出了一种能够提高测量精度的基于Gossip算法的数据聚类算法。还有其他一些应用，比如文献[14]提出了一种新的基于Gossip的信令传播方法，文献[15]以无线传感器网络中的分布式定位问题为研究背景，成功地将成对和广播Gossip算法引入到原有定位算法框架之中。为了加速服务器群间的多批数据传输，文献[16]提出了一种编码排列流言算法，研究了最佳的块划分、批划分和分批调度策略，以达到最小的广播完成时间。

影响Gossip算法性能的因素有很多，主要因素有时间模型、网络结构、信息交互模式以及每个周期、每个节点选择联系的节点个数等。因此Gossip算法只要稍作改变就可以适应很多应用场景。传统Gossip算法最大的缺点是冗余，冗余通信会造成对网路带宽和计算机资源的过多占用，因此降低Gossip算法的冗余度非常重要。本文的目的是降低Gossip算法的冗余度并提高算法的收敛速度。本文的主要贡献是：(1)提出一种二进制指数退避的Gossip算法(Gossip with Binary Exponential Backoff，BEBG)，有效降低了算法的网络负载，但BEBG存在边缘节点问题。(2)提出一个引入Pull的BEBG改进算法PBEBG，以及一个增加向邻居节点Push的BEBG改进算法NBEBG，解决边缘节点问题并提高算法的收敛速度。

## 2　Gossip算法概述

Gossip算法是一种非常著名的信息传播算法，其信息传播过程和图的着色类似。假设网络中只有两种颜色的节点，一种是已着色节点，一种是未着色节点，已着色节点是当前网络状态下已经收到信息的节点，未着色节点为网络中没有收到信息的节点。一个节点在网络中只有两种状态，要么是已着色的，要么是未着色的。刚开始，网络中只有一个已着色节点，第1轮这个节点会随机选择一个节点发送信息。接下来的每一轮次，网络中已着色的节点会随机选择另一节点发送信息。假设未着色节点为红色，已着色节点为蓝色。刚开始网络中只有一个蓝色节点，随着网络状态的变化，蓝色节点会变得越来越多。当蓝色完全覆盖了整个网络，意味着信息已经传播到网络中的所有节点。

### 2.1　时间模型

时间模型是Gossip算法的基础，不同的时间模型有着不同的研究方式。目前常用的时间模型有两种，即同步时间模型和异步时间模型。时间模型是对系统时间的简单假设，现实的时间模型更为复杂。

同步时间模型：将时间分为均匀的时隙，每个时隙，节点选择与自己的邻居节点进行通信，邻居节点选择的过程是随机且相互独立的。同步模型下，节点与节点之间的信息交换均在同一时刻进行，信息的发送与接收都在该时隙内同步完成，因此整个分布式网络中需要一个全局的时钟来同步各个节点的时间。为了简化分析过程，本文设定的时间模型是同步的。

异步时间模型：网络中每个节点都有自己的时钟，按照自己的节奏发送信息。节点根据自己时钟来发起信息交换过程，节点之间的信息发送是互相独立的。异步时间模型不需要全局时钟的同步，这种异步的Gossip算法在分布式环境下更容易实现。

### 2.2　Gossip算法信息交换方式

假设$A$和$B$是大规模分布式系统中的两个节点。Gossip算法中$A$节点和$B$节点信息交换的方式一般有3种：Push, Pull, Push&Pull。

Push：拥有更新信息的节点$A$发起信息交换，在网络中随机选择一个节点$B$，$A$节点把自己的更新信息发送给$B$节点，$B$节点收到信息后更新本地信息。

Pull：未着色节点$A$在网络中随机选择一个节点$B$，要求从$B$节点那里获得更新信息。

Push&Pull：节点之间互相发送更新信息给对方。$A$节点在网络中随机选择一个节点$B$，$A$节点将自己的更新信息发送给$B$节点，$B$节点接收到信息后更新本地信息，并将自己的更新信息发送给$A$节点，$A$节点收到$B$节点的信息后也要更新本地信息。

总的来说，采用Push方式，信息传播先快后慢，Pull方式相反，先慢后快。Push&Pull将两种方式结合起来，这种方式速度是最快的，但也是最消耗网络资源的。

### 2.3 相关研究

文献[17]为Gossip算法提供一个简单而通用的框架，并使用该框架来描述Gossip算法在不同领域中的解决方案。该文献将Gossip算法的过程抽象为3个步骤：节点的选择、信息交换和数据处理。选择节点的方式有很多种，一般情况下，每个周期节点随机选择另一个节点向其发送信息。但是，节点选择向哪些节点发送信息，每个周期发送几条信息，节点发送信息的概率都是可以随着应用场景的改变而改变的。

Gossip算法可以容忍数据包丢失和链路崩溃，尽管流言传播具有很高的容错性和可扩展性，但它仍会受到网络性能下降和网络流量负载过大的影响。因此对于必须同时提供可靠性和及时性的应用，尤其在易拥塞的网络中它可能不是最优的。文献[18]通过确定哪些节点应该接收流言消息来改进流言传播方案，提出采用分布式策略学习逻辑来高效地确定这些节点，文中提出了两个启发式方案：一个是基于QoS(Quality of Service)选择，依据损失模式，从根节点开始沿着路径选择链路质量差的节点；另一个是基于拓扑，根据位置选择组播树内更偏离根的节点。

为了提高算法的收敛速度，以及将信息传播到网络上所有的节点，文献[19]提出了Corrected Gossip算法，将蒙特卡罗式流言和确定性修正相结合，构造一个拉斯维加斯式的可靠广播，在执行Gossip算法有限周期后进入修正阶段，保证以低成本到达所有节点。该文献将未被感染的节点称为未着色的节点。文献[20]在布尔网络模型的基础上提出了布尔流言网络模型。在布尔网络中，每个节点只有两种状态“开” 和“关”，而在某个时刻每个节点只能处于其中一种状态。节点之间相互作用，节点的状态随之发生改变。文献中的网络模型只考虑一对节点间的相互作用。布尔流言网络模型是布尔网络模型的一种特殊形式，可以应用在基因调控、社会意见问卷、病毒传播等多个领域。文献[21]研究了一种网络中节点数目为奇数的Gossip模型，深入研究了同步奇Gossip算法模型和异步奇Gossip算法模型的时间下界，给出了$n=2^k-1$ 时的一个最优同步算法。

为了解决C/S架构信息系统采用的中心化数据同步模式导致的服务器性能下降，并影响信息系统整体性能的问题，文献[22]改进了Gossip算法以解决其冗余通信量大且收敛速度慢的问题，并提出了基于改进Gossip算法的数据同步方法。该方法将Gossip算法与选举算法中的环算法相结合，提高了数据同步效率，减少数据同步对系统整体性能的影响。文献[23]研究和分析了两种自然的基于流言传播的发现过程，即基于Push和Pull的发现过程，对这两个随机过程在无向$n$节点图及在有向$n$节点图中的收敛时间进行了较严密的分析。

## 3 基于二进制指数退避的Gossip算法

以太网介质访问控制协议CSMA/CD要求站点在发送数据之前要先监听信道，监听到信道空闲才发送数据，还需要边发边进行冲突检测，一旦检测到冲突，就采用二进制指数退避算法来决定站点再次尝试发送数据前要退避的时间，经过退避时间后再重复上述步骤。退避时间设定方法：从整数集合$[0, 2^0,2^1,\cdots,(2^{k-1})]$中随机取出一个数$r$，其中$k=\min$(冲突次数,10)；退避时间$=2\tau r$，其中$2\tau$是局域网端到端往返时延。使用随机退避时间是为了降低冲突站点再次同时尝试发送数据而又发生冲突的概率。

为降低冗余，减少Gossip算法中节点在信息传播过程中发送的信息量，本文借鉴二进制指数退避算法的思想对经典Gossip算法(classic Gossip Algorithm, GA)做改进，提出二进制指数退避的Gossip算法BEBG。该算法的思想是节点重复收到同一信息时降低传播信息的概率，重复收到同一信息的次数越多，传播该信息的概率越低。此外，针对BEBG存在边缘节点的不足，从两个方面对其进行改进，进一步提出了PBEBG和NBEBG两个算法。

### 3.1 二进制指数退避的Gossip算法

BEBG与GA两算法最大的不同在于，GA中，每一轮已着色节点向外传播更新信息的概率恒为1，而BEBG中已着色节点向外传播更新信息的概率是变化的。这个概率的大小与节点的网络状态有关，一条新的更新信息在网络中流传，节点重复收到这条信息的轮数越多，节点向外传播这条信息的概率越低。这与现实中人们转发新闻情形类似，对于广为流传的信息人们会失去传播它的欲望。

BEBG中，节点向外传播信息的概率$p$是一个变量，每个轮次一个节点的$p$值可能不一样；未着色节点的$p$值设为0，刚收到信息的节点的$p$值设为1。根节点和第1次收到信息的节点向外传播信息的概率为1，而未着色节点向外传播信息的概率为0。

#### 3.1.1 BEBG的信息发送过程

第1轮，只有根节点有新的更新信息，根节点随机选择一个其他节点以概率$p=1$向其传播更新信

息。第1轮结束时，网络中一共有两个已着色节点。第2轮，这两个已着色节点各自先根据自己向外传播信息的概率$p$决定是否向外发送更新信息，如果是，随机选择一个除自己以外的节点向其传播更新信息。以此类推，若干轮后，网络中所有节点可能都被着色了。节点发送信息的过程是严格全局时钟同步的，每个周期节点只随机选择1个节点向其传播1次更新信息。表1为BEBG的信息发送过程，节点向外传播信息的概率$p$由节点接收更新信息的情况决定，初始化时，根节点的$p$值为1，其他节点的$p$值为0。

### 3.1.2 BEBG的信息接收过程

算法的接收过程比发送过程复杂，见表2。如果本轮中收到了更新信息，第1次收到信息的节点将自己的$p$值由0变为1，而其他收到信息的节点(已着色节点)将自己的$p$值调为原来的1/2，即下一轮向外传播信息的概率变为原来的1/2。例如，如果节点$A$在第4轮中收到更新信息，并且它在第1和第3轮中也曾收到过相同的信息，即收到相同信息的轮次一共为3轮，则节点$A$在第5轮中向外传播信息的概率$p$为1/4，节点$A$在第1—第5轮的$p$分别为0, 1, 1, 1/2, 1/4。以此类推，如果某节点收到相同信息的轮次一共为4轮，那么在下一个轮次该节点的$p$为1/8。为了避免算法后期无节点发送信息导致出现边缘节点(算法结束后还是未着色节点)的情况，本文给已着色节点传播信息的概率$p$设置了一个下限值：1/32。

每一轮中，节点传播信息的概率只有两种可能，不变或变为原来的1/2。如表2，设置一个节点接收超时时间，超过一定时间节点未收到信息，表示本轮该节点的接收过程已经结束，然后线程根据本轮是否收到信息、是否是第1次收到、$p$有无达到下限来将$p$减半或维持不变。

每一轮BEBG先执行信息发送过程(算法1)再执行信息接收过程(算法2)。在接收过程中，当节点发现本轮重复收到之前轮次收到过的信息时，除非$p$已达到下限，否则$p$减半，这里改变的是节点下一轮传播信息的概率。每一轮一个节点收到信息的次数是不确定的，可能零次、一次或多次，但是每轮至多只改变一次$p$的值。

**表 1　BEBG算法的信息发送过程(算法1)**

**Data:** 节点集合 $V$，本节点 $v$ 向外传播信息的概率 $p$;
**Result:** 如果本节点 $v$ 已着色，它可能发送待传播更新信息 info 给另一个节点。

```
(1) 初始化
(2) if 本节点 v 已着色(已获得 info) then
(3)     根据传播概率 p 决定是否传播 info
(4)     if 本节点决定发送 then
(5)         从 V-{v} 中随机选择一个节点
(6)         向该节点发送 info
(7)     end if
(8) end if
```

**表 2　BEBG算法的信息接收过程(算法2)**

**Input:** 本轮中可能接收到所关注的信息 info;
**Result:** 如果本轮中收到了所关注的信息 info，则本节点的传播概率 $p$ 可能被改变，且本节点为已着色。

```
(1) first ← false
(2) reflag ← true, changP ← false
(3) while reflag do
(4)     执行接收操作
(5)     if 如果接收操作没有超时 then
(6)         接收消息
(7)         if 消息包含关注的信息 info then
(8)             changeP ← true
(9)             if 本节点未着色 then
(10)                p ← 1
(11)                first ← true
(12)                colored ← true
(13)            end if
(14)        end if
(15)    else
(16)        if first then
(17)            changeP←false
(18)            first ←false
(19)        end if
(20)        if changeP and p > 1/32 then
(21)            p ← p/2
(22)        end if
(23)        reflag ← false
(24)    end if
(25) end while
```

### 3.1.3 BEBG算法理论分析

本小节分析采用BEBG传播信息，每一轮所有节点发送的总信息量以及信息传播规律。设未着色节点为0类节点，$C_0(T)$为在$T$轮次初始尚未着色的节点总数，它们向外传播信息的概率$p_0=0$；一个节点若在$T$轮次之前累计在$i$个轮次中都接收到了信息，则为$i$类节点，$C_i(T)$表示在$T$轮次$i$类节点总数，$i$类节点向外传播信息的概率$P_i=1/2^{i-1}$；其中$i$=1,2,3,4,5；一个节点若在$T$轮次之前累计在大于等于6个轮次中都接收到了信息，则为6类节点，$C_6(T)$表示在$T$轮次6类节点总数，6类节点向外

传播信息的概率$P_6$=1/32。$C(T)$为在$T$轮次初始已着色节点的总数，等于1到6类节点的总和。$C(1)$=1, $C_1(1)$=1, $C_2(1)$=$C_3(1)$=$C_4(1)$=$C_5(1)$=$C_6(1)$=0。

设当前轮次为$T$，在$T$轮次向外传播信息的节点数$S(T)$预期为

$$S(T)=C_1(T)+\frac{1}{2}\times C_2(T)+\frac{1}{4}\times C_3(T)+\frac{1}{8}\times C_4(T)+\frac{1}{16}\times C_5(T)+\frac{1}{32}\times C_6(T) \tag{1}$$

显然，$S(1)$=1。

设网络中共有$N$个节点，假设节点$A$是一个将向外发送信息的已着色节点，节点$B$是网络中的任意一个节点，节点$B$是否着色是未知的。节点$A$随机选择一个除自己之外的节点发送信息，所以网络中任意节点$B$在当前轮次收到节点$A$发送的信息的概率为1/($N$–1)。

任意节点$B$在当前轮次收不到节点$A$发送的信息的概率为

$$P=1-\frac{1}{N-1} \tag{2}$$

任意节点$B$在当前轮次收不到来自任何将向外传播信息的已着色节点的信息的概率$P_{\mathrm{f}}$为

$$P_{\mathrm{f}}=\left(1-\frac{1}{N-1}\right)^{S(T)} \tag{3}$$

任意节点$B$在当前轮次至少收到一条来自将向外传播信息的已着色节点的信息的概率$P_{\mathrm{s}}$为

$$P_{\mathrm{s}}=1-\left(1-\frac{1}{N-1}\right)^{S(T)} \tag{4}$$

又由于当前网络中未着色节点个数$C_0(T)$即为$N$–$C(T)$，所以未着色节点在当前轮次被着色(至少收到一条信息)的预期个数为

$$(N-C(T))\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right] \tag{5}$$

对于BEBG，当前轮次结束即下一轮次初始已着色节点的总数预期为

$$C(T+1)=C(T)+(N-C(T))\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right] \tag{6}$$

下一轮次的$C_i(T+1)$，其中$i$=1, 2, 3, 4, 5，由两部分组成，当前轮次$i$类节点的个数减去$i$类节点在本轮收到信息而转变为$i$+1类节点的个数，以及当前$i$–1类节点在本轮收到信息而转变为$i$类节点的个数。即在本轮未收到信息的$i$类节点的个数，加上$i$–1类节点在本轮收到信息的个数。其计算公式为

$$\begin{aligned}C_i(T+1)=&C_i(T)-C_i(T)\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right]\\&+C_{i-1}(T)\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right]\\=&C_i(T)\left(1-\frac{1}{N-1}\right)^{S(T)}\\&+C_{i-1}(T)\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right],\\&i=1,2,3,4,5\end{aligned} \tag{7}$$

下一轮次的$C_6(T$+1)由当前轮次6类节点的个数，加上5类节点收到信息的个数。其计算公式为

$$C_6(T+1)=C_6(T)+C_5(T)\left[1-\left(1-\frac{1}{N-1}\right)^{S(T)}\right] \tag{8}$$

而对于GA，当前轮次结束即下一轮次初始已着色节点的总数预期为

$$C(T+1)=C(T)+(N-C(T))\left[1-\left(1-\frac{1}{N-1}\right)^{C(T)}\right] \tag{9}$$

$S(T)$是在$T$轮次向外传播信息的节点数，也是在$T$轮次发送的总信息量，显然BEBG能够有效降低网络负载，但是BEBG降低信息量是以增加周期为代价的。在算法的后期，大量节点对外传播信息的概率为1/32，这可能导致有极个别未着色节点长期收不到更新信息。很多周期后，某些未着色节点可能还是没有收到信息，本文称这些仿佛被隔绝了的，一直(或长期)收不到信息的节点为边缘节点。为了解决BEBG存在的边缘节点问题，3.2节和3.3节对BEBG进行了改进。

### 3.2 PBEBG算法

Gossip信息交换的方式有3种：Pull, Push, Push&Pull。BEBG实质上是对Push模式的GA的改进。在Pull模式的GA中，未着色节点会一直向其他节点发送更新请求消息，直到找到一个已着色节点，基于Pull的方法理论上不会出现边缘节点。因此，本文将Pull与BEBG相结合来解决BEBG存在的边缘节点问题，此BEBG改进算法称为PBEBG(Binary Exponential Backoff Gossip with Pull)。如图1所示，在算法的后期，边缘节点随机选择一个节点向其发送请求消息，接收到请求消息的节点将更新信息(如果有)发送给该边缘节点。图1中灰色节点为已着色节点，白色节点为未着色节点。假设节点5随机选择向节点9发送更新请求消息，节点9收到后响应请求，将更新信息发送给节点5。

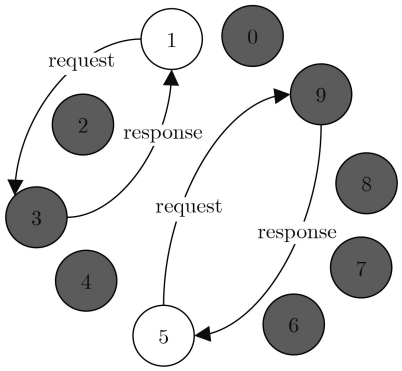



图 1 PBEBG算法中的“拉”

### 3.2.1 PBEBG的发送过程

PBEBG的发送过程比较复杂。初始化时，根节点的$p$值为1，其他节点的$p$值为0。发送信息之前节点要先确定是否在上一个轮次收到其他节点的更新请求。如果节点是已着色的且在上一个轮次收到了请求消息，那么本轮中节点将直接把自己的更新信息发送给请求节点，如果请求节点不止一个，节点将会从请求节点中选取一个节点向其发送更新信息。

如果节点没有收到更新请求，那么信息发送过程与BEBG的相似，先根据节点向外传播信息的概率$p$决定是否向外发送更新信息，如果是，则随机选取一个除自己以外的节点向其发送更新信息。PBEBG的发送过程见表3。

PBEBG算法还引入了一个新的变量“Pull”。这里的“Pull”不是GA算法信息交换的一种方式，而是代表未着色节点向外发送更新请求的时机。设网络中一共有$N$个节点，刚开始网络中有$N$–1个未着色节点，不可能第1轮就让$N$–1个节点向外发送请求，这会导致第1轮时未着色节点发出$N$–1条请求消息，然而只有一条请求能得到根节点的回复，这会造成网络资源的极大浪费。因此，从哪一个轮次开始发送更新请求就至关重要了。节点决定发送更新请求的前提是：(1)当前节点未被着色；(2)当前的轮次$T$大于等于“Pull”值(见表4)，表3中，request为真时发送更新请求，随机选取一个除自己以外的节点向其发送更新请求。

### 3.2.2 PBEBG的接收过程

PBEBG的接收过程比BEBG的接收过程多一个判断，因为这时网络中有两种消息，一种是来自已着色节点的传播感兴趣信息的更新消息，一种是来自未着色节点的更新请求消息。

每1轮，1个节点$v$可能既接收到网络中未着色节点发来的更新请求，也接收到已着色节点传播的更新信息。节点$v$有可能收到多条更新请求消息，这时，线程只会存储最后一条请求消息的来源，记为$s$，在下一轮如果节点$v$是已着色节点则会将更新消息发给节点$s$。如果节点$v$收到了多条包含相同信息的更新消息，这时的处理方法与BEBG算法的一样，每1轮只改变1次$p$的值。PBEBG的接收过程见表4。

**表 3　PBEBG算法的发送过程(算法3)**

**Data:** 节点集合 $V$，当前轮次 $T$，本节点传播概率 $p$，响应标记 response，请求标记 request ，最后一个发来请求的节点 $s$；
**Result:** 如果本节点 $v$ 未着色且 request 为真，则发送请求给另一个节点；如果本节点已着色，可能发送待传播信息info给另一个节点；或者如果response为真则发送info给最后一个发来请求的节点 $s$。

```
(1) 初始化
(2) if 本节点 v 未着色且决定pull (即request为真) then
(3)         从 V-{v} 中随机选择一个节点
(4)         向该节点发送一个请求
(5) end if
(6) if 本节点 v 已着色(已获得 info)then
(7)         if 本节点在上一轮收到了请求(即response为真) then
(8)             向最后一个发来请求的节点 s 发送info
(9)         else
(10)            根据传播概率 p 决定是否传播 info
(11)            if 本节点决定发送 then
(12)                从 V-{v} 中随机选择一个节点
(13)                向该节点发送 info
(14)            end if
(15)        end if
(16) end if
```

PBEBG算法有效减少了网络负载，并解决了边缘节点问题，但是该算法需要未着色节点在一定时间(轮次)后，向网络中的其他节点发送更新请求消息。然而，除非有约定或每个节点定期地去询问，否则未着色节点不知道网络中什么时候会有新的信息，也就无法估计当前处于什么轮次了。另外，Pull参数的设置需要事先做约定。这些因素减少了该算法的适用场景。因此本文又提出了一种不需要Pull的BEBG改进算法。

### 3.3 NBEBG算法

本节提出一个基于邻居节点的二进制指数退避Gossip算法(Neighbor-based Binary Exponential Backoff Gossip, NBEBG)，该算法增加了一个定向“推送”。与BEBG算法不同的是，BEBG算法随机选择网络中的节点向其传播信息，而NBEBG算法则是先判断本节点是否为第1次接收到更新信息，如果是，则将信息传播给前1个邻居节点，如果不是第1次，则将信息传播给从网络中随机选择的一个节点。算法运行到最后，其实相

**表 4 PBEBG算法的接收过程(算法4)**

**Data:** 节点集合 $V$，当前轮次 $T$，阈值 Pull；

**Input:** 本轮中可能接收到所关注的信息 info 和请求；

**Result:** 如果本轮中收到了所关注的信息 info，则本节点的传播概率 $p$ 可能被改变，且本节点为已着色；如果本轮中收到了请求，则设置响应标记 response 为真，并记录最后一个发来请求的节点 $s$；如果本节点在本轮最终仍是未着色的，且当前轮次$T$>=Pull，则设置请求标记 request 为真。

```
(1) first ← false, response ← false, request ← false
(2) reflag ← true, changP ← false
(3) while reflag do
(4)         执行接收操作
(5)         if 如果接收操作没有超时 then
(6)                 接收来自节点src的消息
(7)                 if 这是一个请求消息 then
(8)                         response ← true, s ← src
(9)                 else if 消息包含关注的信息 then
(10)                        changeP ← true
(11)                        request ← false
(12)                        if 本节点未着色 then
(13)                                p ← 1
(14)                                first ← true
(15)                                colored ← true
(16)                        end if
(17)                end if
(18)        else
(19)                if 本节点未着色 且 T>=Pull then
(20)                        request←true
(21)                end if
(22)                if first then
(23)                        changeP←false
(24)                        first←false
(25)                end if
(26)                if changeP and p > 1/32 then
(27)                        p ← p/2
(28)                end if
(29)                reflag ← false
(30)        end if
(31) end while
```

当于每一个着色节点都向前一个节点发送了一次更新信息，如果没有丢包，理论上是不会存在边缘节点的。

如图2所示，给网络中的节点编号，灰色节点为已着色节点，白色节点为未着色节点。满足条件的已着色节点将更新信息发送给前一个节点，是否传播信息与前一个节点是否着色无关。

### 3.3.1 NBEBG的发送过程

NBEBG的发送过程比BEBG的多了一个判断。每个已着色节点每轮至多只传播一次信息，如果它向前一个邻居节点发送更新信息则不会再向网络中的其他节点发送更新信息。根据之前的经验，从哪一个轮次开始可能对总信息量产生影响，因此NBEBG算法引入了一个新的变量“Push”，代表已着色节点向邻居节点发送信息的时机，也就是说，已着色节点向前一个节点发送更新信息是在到达规定的轮次之后才会发送。算法会记录已着色节点是否已经向前一个节点发送过更新信息。只有当前轮次已达到规定轮次，且还没有向前一个节点发送过更新信息，已着色节点才会向前一个节点发送。NBEBG的信息发送的具体过程如表5所示。初始化时，根节点的$p$值为1，其他节点的$p$值为0，所有节点的hasPush标志为false。

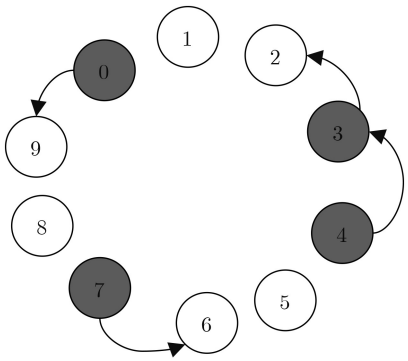



图 2 NBEBG中向前1个邻居节点推送信息

**表 5 NBEBG算法的信息发送过程(算法5)**

**Data:** 节点集合 $V$，当前轮次 $T$，本节点 $v$ 向外传播信息的概率 $p$，标记 hasPush，阈值Push；

**Result:** 如果本节点 $v$ 已着色，它可能发送待传播更新信息 info 给另一个节点，或者如果 hasPush 为假且 $T$>=Push 则发送 info 给本节点的前一个节点。

```
(1) 初始化
(2) if 本节点 v 已着色(已获得 info) then
(3)         if!hasPush and T>=Push then
(4)                 发送 info 给本节点的前一个节点
(5)                 hasPush ←true
(6)         else
(7)                 根据传播概率 p 决定是否传播 info
(8)                 if 本节点决定发送 then
(9)                         从 V-{v} 中随机选择一个节点
(10)                        向该节点发送 info
(11)                end if
(12)        end if
(13) end if
```

### 3.3.2 NBEBG的接收过程

NBEBG算法接收过程与BEBG算法接收过程完全一样，未做任何修改。

## 4 实验

第3节提出了两个BEBG改进算法：引入Pull的PBEBG和向前一个邻居节点Push的NBEBG。为了做对比实验，本文也将这两种改进方法引入经典Gossip算法。将引入Pull的GA称为PGA，将向前一个邻居节点Push的GA称为NGA。本文研究并实现了6种Gossip算法，如图3所示。PGA与NGA是GA的变种，PBEBG和NBEBG是BEBG的变种。

本文实现的程序是基于JAVA的高并发程序，

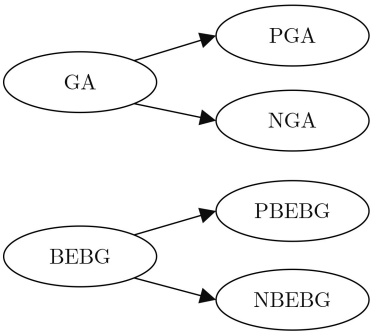
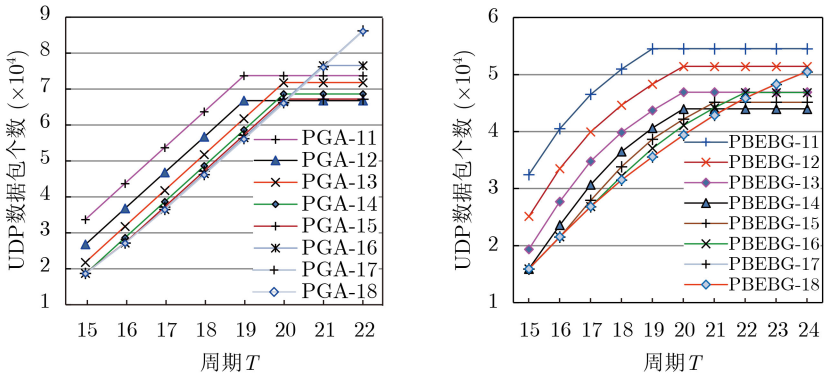



图 3 6种算法名称

通过使用JAVA的一些同步机制来模拟大规模分布式网络的环境，程序中实现了10000个高并发线程模拟10000个节点。实验结果表明，通过改变传播信息的概率，能够有效地降低Gossip算法的网络开销。Gossip算法的性能一般由算法的收敛速度和网络负载两个方面来衡量。算法的收敛速度是指算法将信息传播到网络上所有节点所需的周期数。网络负载又称为信息量，在本文中指信息传播时所发送的UDP数据包个数。本文中所有数据都是程序运行30次取的平均值。

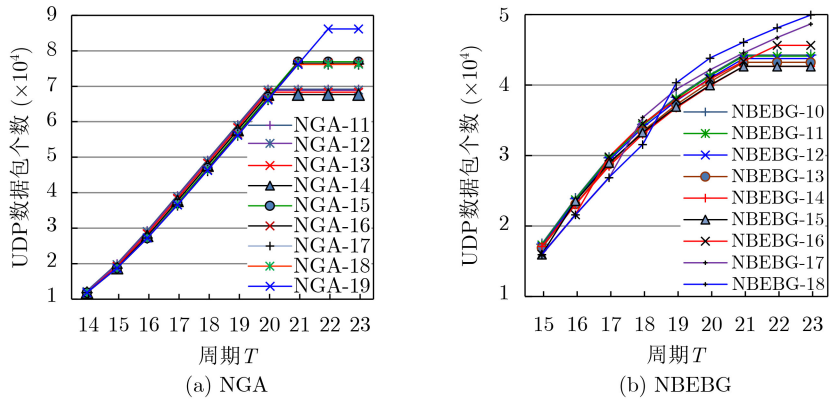
图4为“Pull”取不同值时，PBEBG与PGA总信息量随时间变化的示意图，即累计发送的UDP数据包个数随着轮次(周期)的变化图。由图可见，边缘节点开始向网络中其他节点发送请求消息的起始轮次对算法的信息量是有影响的；“Pull”取值为12时，PGA的信息量最少，“Pull”为14时，PBEBG的信息量最少。网络中有$10^4$个节点时，PBEBG比引入Pull的GA可减少约34%的网络负载。

图5为“Push”取不同值时，NBEBG与NGA总信息量随时间变化的示意图。由图可见，节点向前一个邻居节点发送信息的具体轮次对算法的信息量是有影响的；“Push”取值为14时，NGA的信




(a) PGA　　(b) PBEBG

图 4 不同“Pull”值时，算法PGA和PBEBG随时间变化的总信息量




(a) NGA　　(b) NBEBG

图 5 不同“Push”值时，算法NGA和NBEBG随时间变化的总信息量

息量最少，“Push”为15时，NBEBG的信息量最少。网络中有$10^4$个节点时，NBEBG比引入向邻居节点Push的GA减少了约37%的网络负载。

图6是6种算法将信息传遍整个网络所需的周期数。实验表明，除了BEBG，其余Gossip算法基本都能在30个周期以内将信息传遍网络，GA需要24个周期，但24周期时BEBG只能将信息传到网络中97.5%的节点，图中的28是虚值，代表很大的值。而加入“Pull”方式或者加入“向邻居Push”的4个算法，NGA-14(参数Push =14)，PGA-12(参数Pull=12)，NBEBG-15(参数Push =15)，PBEBG-14(参数Pull=14)，只需要19到21个周期就能将信息传遍网络，收敛速度相差不大，都有效解决了边缘节点问题。

图7为6种算法总信息量对比图。GA(24个周期)，NGA-14，PGA-12这3种算法的总信息量依次递减。而包含指数退避的算法中，BEBG(24个周期)，NBEBG-15，PBEBG-14这3种算法的总信息量依次递增。BEBG(24个周期)总信息量最少，但并不能将信息传遍整个网络，存在边缘节点问题；网络有$10^4$个节点时它比GA(24个周期)减少了约61%的网络负载。

实验结果显示，PBEBG与NBEBG均改进了BEBG存在边缘节点的不足，同时具有BEBG的网络负载低的优点。PBEBG与NBEBG的收敛速度和产生的网络负载相差无几，两算法应用条件不同，各有各的优势，PBEBG不需要知道自己的前一个节点是谁，而NBEBG不需要其他节点的响应。实际应用中，可以根据实际情况选择具体算法，比如基于DHT(分布式哈希表)的对等网络中的信息传播可以使用NBEBG，因为每个节点知道自己的前继，而对于信息更新周期比较长，节点之间基本同步，一轮时长可以估计的应用场景可以使用PBEBG。

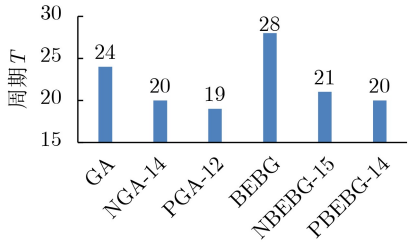



图 6 6种算法将信息传遍整个网络所需周期对比

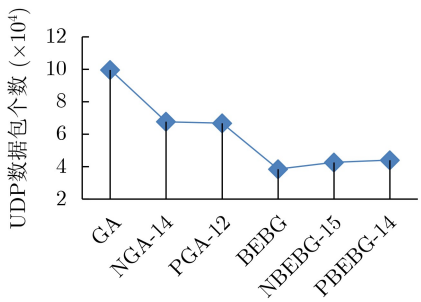



图 7 6种算法总信息量对比

## 5 结束语

为减少Gossip算法的网络负载，本文提出了一个将二进制指数退避算法与经典Gossip算法相结合的改进算法BEBG，它使用指数退避的信息传播策略，使一个节点收到同一更新信息的次数越多，继续传播更新信息的概率越低。仿真实验表明，该算法能够有效地减少网络负载，网络包含$10^4$个节点时BEBG比经典Gossip算法降低了约61%的网络负载。为解决BEBG的边缘节点问题，对BEBG算法作改进，提出了引入Pull的BEBG改进算法PBEBG，在若干轮次后让未收到更新信息的节点主动询问其他节点，网络包含$10^4$个节点时，PBEBG比引入Pull的经典Gossip算法减少约34%的网络负载。为解决边缘节点问题，对BEBG另作改进，提出了一种向邻居节点Push的BEBG改进算法NBEBG，除了常规Push，还让收到更新信息的节点向其前一个邻居节点发送更新信息以达到加快收敛速度的目的，网络包含$10^4$个节点时，NBEBG比引入向邻居节点Push的经典Gossip算法减少了约37%的网络负载。

## 参 考 文 献

[1] DEMERS A, GREENE D, HOUSER C, *et al*. Epidemic algorithms for replicated database maintenance[J]. *ACM SIGOPS Operating Systems Review*, 1988, 22(1): 8–32. doi: 10.1145/43921.43922.

[2] GUPTA I, VAN RENESSE R, and BIRMAN K P. Scalable fault-tolerant aggregation in large process groups[C]. 2001 International Conference on Dependable Systems and Networks, Gothenburg, Sweden, 2001: 433–442. doi: 10.1109/DSN.2001.941427.

[3] JELASITY M and BABAOGLU O. T-Man: Gossip-based overlay topology management[C]. The 3rd International Workshop on Engineering Self-Organising Applications, Utrecht, The Netherlands, 2005: 1–15. doi: 10.1007/11734697_1.

[4] PROVENSI L L, SINGH A, ELIASSEN F, *et al*. Maelstream: Self-organizing media streaming for many-to-many interaction[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2018, 29(6): 1342–1356. doi: 10.1109/TPDS.2018.2791599.

[5] LIU Mengdi, XU Guangquan, ZHANG Jun, *et al*. Roundtable gossip algorithm: A novel sparse trust mining

method for large-scale recommendation systems[C]. The 18th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP 2018, Guangzhou, China, 2018: 495–510. doi: 10.1007/978-3-030-05063-4_37.

[6] LAVINIA A, DOBRE C, POP F, *et al*. A failure detection system for large scale distributed systems[J]. *International Journal of Distributed Systems and Technologies*, 2011, 2(3): 64–87. doi: 10.1109/CISIS.2010.29.

[7] VAN RENESSE R, BIRMAN K P, and VOGELS W. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining[J]. *ACM Transactions on Computer Systems*, 2003, 21(2): 164–206. doi: 10.1145/762483.762485.

[8] PIETRO R D and MICHIARDI P. Gossip-based aggregate computation: Computing faster with non address-oblivious schemes[C]. The 27th ACM Symposium on Principles of Distributed Computing, Toronto, Canada, 2008: 442. doi: 10.1145/1400751.1400837.

[9] WANG Gang, WANG Zhiyue, and WU Jie. A local average broadcast gossip algorithm for fast global consensus over graphs[J]. *Journal of Parallel and Distributed Computing*, 2017, 109: 301–309. doi: 10.1016/j.jpdc.2017.05.008.

[10] 张仕将, 柴晶, 陈泽华, 等. 基于Gossip协议的拜占庭共识算法[J]. 计算机科学, 2018, 45(2): 20–24. doi: 10.11896/j.issn.1002-137X.2018.02.004.
ZHANG Shijiang, CHAI Jing, CHEN Zehua, *et al*. Byzantine consensus algorithm based on gossip protocol[J]. *Computer Science*, 2018, 45(2): 20–24. doi: 10.11896/j.issn.1002-137X.2018.02.004.

[11] WANG Huiwei, LIAO Xiaofeng, WANG Zidong, *et al*. Distributed parameter estimation in unreliable sensor networks via broadcast gossip algorithms[J]. *Neural Networks*, 2016, 73: 1–9. doi: 10.1016/j.neunet.2015.09.008.

[12] AZIMI R and SAJEDI H. Peer sampling gossip-based distributed clustering algorithm for unstructured P2P networks[J]. *Neural Computing and Applications*, 2018, 29(2): 593–612. doi: 10.1007/s00521-017-3119-0.

[13] AZIMI R and SAJEDI H. A decentralized gossip based approach for data clustering in peer-to-peer networks[J]. *Journal of Parallel and Distributed Computing*, 2018, 119: 64–80. doi: 10.1016/j.jpdc.2018.03.009.

[14] FEMMINELLA M, FRANCESCANGELI R, REALI G, *et al*. Gossip-based signaling dissemination extension for next steps in signaling[C]. 2012 IEEE Network Operations and Management Symposium, Maui, United States, 2012: 1022–1028. doi: 10.1109/NOMS.2012.6212024.

[15] 崔闻. 基于Gossip的无线传感器网络定位算法研究[D]. [硕士论文], 哈尔滨工业大学, 2014.
CUI Wen. Research on gossip-based localization algorithms in wireless sensor networks[D]. [Master dissertation], Harbin Institute of Technology, 2014.

[16] LIU Yan, NIU Di, and KHABBAZIAN M. Cooper: Expedite batch data dissemination in computer clusters with coded gossips[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2017, 28(8): 2204–2217. doi: 10.1109/TPDS.2017.2654242.

[17] KERMARREC A M and VAN STEEN M. Gossiping in distributed systems[J]. *ACM SIGOPS Operating Systems Review*, 2007, 41(5): 2–7. doi: 10.1145/1317379.1317381.

[18] ESPOSITO C, CASTIGLIONE A, PALMIERI F, *et al*. Improving the gossiping effectiveness with distributed strategic learning[J]. *Future Generation Computer Systems*, 2017, 71: 221–233. doi: 10.1016/j.future.2016.11.006.

[19] HOEFLER T, BARAK A, SHILOH A, *et al*. Corrected gossip algorithms for fast reliable broadcast on unreliable systems[C]. 2017 IEEE International Parallel and Distributed Processing Symposium, Orlando, United States, 2017: 357–366. doi: 10.1109/IPDPS.2017.36.

[20] LI Bo, WU Junfeng, QI Hongsheng, *et al*. Boolean gossip networks[J]. *IEEE/ACM Transactions on Networking*, 2018, 26(1): 118–130. doi: 10.1109/TNET.2017.2763964.

[21] FERTIN G, PETERS J G, RAABE L, *et al*. Odd gossiping[J]. *Discrete Applied Mathematics*, 2017, 216: 550–561. doi: 10.1016/j.dam.2016.01.034.

[22] 田振兴, 代杰. 基于改进Gossip协议的数据同步设计[J]. 指挥信息系统与技术, 2017, 8(5): 99–103. doi: 10.15908/j.cnki.cist.2017.05.017.
TIAN Zhenxing and DAI Jie. Data synchronization design based on improved gossip protocol[J]. *Command Information System and Technology*, 2017, 8(5): 99–103. doi: 10.15908/j.cnki.cist.2017.05.017.

[23] HAEUPLER B, PANDURANGAN G, PELEG D, *et al*. Discovery through gossip[C]. The 24th ACM Symposium on Parallelism in Algorithms and Architectures, Pittsburgh, United States, 2012: 140–149. doi: 10.1145/2312005.2312031.

成卫青：女，1972年生，教授，研究方向为网络测量、分布式系统和模式识别.

张　蕾：女，1994年生，硕士，研究方向为分布式系统.

责任编辑：马秀强