软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(7):1849−1868 [doi: 10.3724/SP.J.1001.2012.04097]

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

TypeSampler:一种基于 gossip 的类型采样方法*

郑 重+, 王意洁, 马行空, 杨永滔

(国防科学技术大学 计算机学院 并行与分布处理国家重点实验室,湖南 长沙 410073)

TypeSampler: A Type Sampling Approach Based on Gossip

ZHENG Zhong+, WANG Yi-Jie, MA Xing-Kong, YANG Yong-Tao

(National Key Laboratory for Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: zhengzhong@nudt.edu.cn

Zheng Z, Wang YJ, Ma XK, Yang YT. TypeSampler: A type sampling approach based on gossip. *Journal of Software*, 2012,23(7):1849−1868 (in Chinese). http://www.jos.org.cn/1000-9825/4097.htm

Abstract: In many P2P applications, nodes can be classified into different types according to their interests or resources, and the routing for the nodes with a specified type over overlay networks is the key for data distribution and query in these applications. Unstructured overlays have a low maintenance cost and high robustness, but fail to ensure the routing efficiency. This paper proposes a gossip-based type sampling approach—TypeSampler, which samples the nodes of different types with the same probability (called type sampling). The type sampling ensures the lower bound probability of finding a neighbor node with a specified type at any node, and thus ensures the routing efficiency over the unstructured overlay. For type sampling, TypeSampler first implements the proportion estimation of types through peer sampling and anti-entropy aggregation based on gossip. Next, TypeSampler maintains a type sampling table at each node, periodically, based on the estimated proportion values. Theoretical analysis and experimental results reveal that TypeSampler can not only achieve precise proportion estimation and approximately uniform random type sampling, but can also work well even in the dynamic network environment. Moreover, TypeSampler can support more efficient routing and has better scalability compared to the existing approaches.

Key words: type sampling; proportion estimation; routing; unstructured overlay network; P2P

摘 要: 在很多 P2P 应用中,节点可以根据其兴趣或资源划分为不同的类型,而以特定类型节点为目标的基于覆盖网的路由也就成为实现数据分发及查询的关键.非结构化覆盖网具有维护开销低、鲁棒性高的优点,却也因此难以保证路由效率.提出了一种基于 gossip 的类型采样方法——TypeSampler,它以等概率采样不同类型的节点(称为类型采样),以此保证在任意节点发现特定类型邻居节点的概率下界,进而保证非结构化覆盖网中的路由效率.为了实现类型采样,TypeSampler 首先通过基于 gossip 的节点采样及反熵聚集估计各类型节点的比例,然后,TypeSampler 再根据这些比例估计值周期性地维护每个节点的类型采样表.理论分析与实验结果表明,TypeSampler 能够实现精确

* 基金项目: 国家自然科学基金(60873215); 国家重点基础研究发展计划(973)(2011CB302601); 湖南省自然科学杰出青年基金(S2010J5050); 高等学校博士学科点专项科研基金(200899980003)

收稿时间: 2011-03-05; 修改时间: 2011-05-18; 定稿时间: 2011-07-21

的类型比例估计以及近似均匀随机的类型采样,并能适应动态的网络环境.而且相对于已有的方法,TypeSampler 能够支持更高效的路由,且具有更好的可扩展性.

关键词: 类型采样;比例估计;路由;非结构化覆盖网;P2P

中图法分类号: TP393 **文献标识码:** A

随着 Internet 的迅速发展,各种 P2P 应用日益流行.P2P 覆盖网是由 P2P 应用中节点组成的逻辑网络,是 P2P 应用的基础设施.在 P2P 覆盖网中,每个节点地位对等,同时充当着服务器与客户的角色.P2P 覆盖网可分为两类:非结构化覆盖网与结构化覆盖网.非结构化覆盖没有固定、严格的拓扑结构,因而维护开销较低且具有较强的鲁棒性,典型的系统有 Gnutella[1],Freenet[2]等.结构化覆盖网的拓扑由确定性的算法严格控制,因而能够支持节点及资源的高效定位,但维护开销较大,鲁棒性较差,典型的系统有 Chord[3],CAN[4]等.

非结构化覆盖网因其维护开销小、鲁棒性强而常常被选作面向动态网络环境的各种 P2P 应用的基础架构.在这些应用中,节点的兴趣以及存储的资源往往可以按照主题或者类型分类,而节点也可以根据这些主题或类型划分为不同的子集.在本文中,我们将主题与类型统称为类型.于是,数据分发及数据查询的问题也就转化为如何将数据/查询快速路由至属于特定类型的节点问题.然而,非结构覆盖网本身的拓扑过于简单,难以支持高效的路由.虽然很多研究通过引入超级节点[5,6]、局部索引[7−10]、语义簇聚[11−13]等方法以改善路由性能,但是非结构化覆盖网固有的随机特性还是使得路由效率难以得到保证.

为了支持非结构化覆盖网中的高效路由,我们提出了一种基于 gossip 的类型采样方法——TypeSampler. TypeSampler 通过对系统中不同类型的节点进行均匀随机采样(称为类型采样)来保证路由的平均效率.在 TypeSampler 中,每个节点首先通过 gossip 对系统中的节点进行采样以估计各类型节点的比例,然后再通过基于 gossip 的反熵聚集快速精化这些初步的估计值.TypeSampler 利用这些类型比例估计值为每个节点维护一个类型采样表(type sampling table,简称 TST),尽可能地使不同类型的节点以等概率出现在该表中.节点通过节点采样建立起的邻居关系实际上构成了一个非结构化覆盖网,而每个节点的类型采样表则可以作为路由表来引导数据或查询在这个非结构化覆盖网上路由.值得一提的是:一方面,TypeSampler 的比例估计精度随着系统节点规模的增大而有所提高;另一方面,TypeSampler 的存储与通信开销随着单个节点所属类型数的增加而增长缓慢,因而 TypeSampler 的类型采样以及对高效路由的支持具有良好的可扩展性.

TypeSampler 中的类型可以根据需求而赋予不同的语义,因而 TypeSampler 能够作为基础服务而整合到多种基于非结构化覆盖网的 P2P 应用中.如果将类型定义为拥有特定副本,那么副本复制算法[14]便可以利用 TypeSampler 所获取的各类型节点比例估计值来优化其复制策略;如果将类型定义为拥有特定资源,那么对于根据节点资源相似性簇聚节点的资源共享系统[12],TypeSampler 可以集成到其底层架构中以保证针对各种类型资源的查询路由效率;如果将类型定义为具有特定兴趣,那么对于根据节点兴趣簇聚节点的发布/订阅系统[15], TypeSampler 同样可以集成到其底层架构中,以引导各种事件迅速路由至兴趣匹配的节点簇.由于文献[12,15]中的系统均采用双层拓扑结构,且维护其底层拓扑的节点采样服务也正是 TypeSampler 实现比例估计及类型采样所需要的,因此,TypeSampler 可以很容易地集成到这些现有系统中.

1 相关工作

1.1 非结构化覆盖网上的路由

为了提高非结构化覆盖网的路由性能,已有很多方法被提了出来.根据其实现方式,这些方法大致可以分为 4 类:基于超级节点的方法[5,6]、基于局部索引的方法[7−10]、基于语义簇聚的方法[11−13]、基于类型采样的方法[16].

基于超级节点的方法利用节点异构性,在覆盖网中引入超级节点.超级节点负责维护普通节点提供的数据索引,主要的数据查询路由在超级节点间完成.由于超级节点存放了大量数据索引且超级节点的数目也相对较少,因而路由跳步数以及通信开销也得到了限制.然而,超级节点的存在影响了整个系统的鲁棒性,而且对于节

点能力普遍有限的情况也不适用.

基于局部索引的方法通过在每个节点上维护部分索引信息来引导路由.这些索引可以是数据索引[7,8];也可以是节点兴趣索引[9,10].节点在转发数据或查询时,根据自己维护的局部索引来选择合适的邻居作为下一跳,直到遇到目标节点.由局部索引引导的路由的效率高于单纯的泛洪及随机行走,但是数据以及兴趣在覆盖网中的随机散布,使得路由效率无法得到保证.

基于语义簇聚的方法通过在维护覆盖网的过程中簇聚语义相似的节点来提高路由效率.一方面,节点对邻居语义的了解可以引导路由;另一方面,由于节点与自己的邻居语义相近,因此一旦数据或查询命中一个目标节点,那么它们也很容易由此出发找到其他目标节点.然而与基于局部索引的方法类似,基于语义簇聚的方法难以保证路由效率,特别是以稀有兴趣或资源为目标的路由可能需要较大的时间延迟与通信开销才能成功.

基于类型采样的方法通过对系统中不同类型的节点进行均匀随机采样来保证平均路由的效率.TERA[16]基于节点规模估计实现类型采样.然而,对每种类型节点的规模进行估计,需要为每种类型的节点分别维护单独的覆盖网.所以,当TERA在每个节点所属类型数较多时存储与通信开销较大,可扩展性较差.而且,由于TERA仅利用类型采样来引导路由而没有实现及利用对邻居节点类型的感知,因此其路由性能也仍有较大的改进空间.

相对于 TERA,TypeSample 基于类型比例估计实现类型采样,不需要为每个类型维护单独的覆盖网,因而节点的存储及通信开销相对较少,具有更好的可扩展性.而且,由于 TypeSamplert 利用邻居节点及类型采样表中的节点类型信息,因此其对路由性能的保证也显著优于 TERA.

1.2 Gossip

Gossip 是一种基本的通信模式,意指系统中的节点间重复地、概率地交换信息.Gossip 不仅可以用来实现数据分发,而且还可以用来实现节点采样、拓扑维护、聚集计算、资源管理等多种功能[17].

基于 gossip 的节点采样方法已有不少,其基本思想都是视图交换[18−20].在这些方法中,每个节点维护一个由若干节点条目组成的视图,节点周期性地与视图中的某个节点交换彼此的视图,并利用接收到的视图条目来更新自己的视图.这些方法不仅能够保证视图成为对系统中节点的均匀随机采样,而且也通过视图维护了一个动态更新的、鲁棒的非结构化覆盖网.文献[20]提出了基于 gossip 的节点采样方法的共同框架,TypeSampler 采用其作为底层的节点采样服务,从而为比例估计以及类型采样提供输入.

基于 gossip 的反熵聚集方法以及相关的理论分析也有很多[21−23],在这些方法中,每个节点周期性地与某个随机节点交换各自保存的数值,并利用接收到的数值按照一定的规则更新自己原有的数值.这些方法可以用来进行平均值、总和及系统节点规模等多种聚集计算.另外,有的基于反熵聚集的方法可以实现对属性值分布的估计[24,25].尽管分布估计与比例估计类似,然而这些方法针对的是连续数值分布的估计,因而难以直接应用于离散的比例估计.实际上,就我们所知,目前尚没有专门的比例估计方法.TypeSampler 在进行比例估计时需要利用反熵聚集来获取各节点估计值的平均值,因而采用了文献[22]中的基本思想,而文献[22]中的相关理论结果也为对 TypeSampler 进行理论分析提供了基础.不过,文献[22]中的节点在反熵聚集时只维护单一的数值,而 TypeSampler 中的每个节点则针对多个类型的比例估计值进行反熵聚集;并且由于节点随机选择其负责反熵聚集的类型,因此,节点对在每次聚集交互时能够同时更新多个共同类型的比例估计值,从而实现快速收敛.

2 TypeSampler 的基本思想

TypeSampler 基于 gossip 实现比例估计,进而根据比例估计实现类型采样.比例估计是指估计系统中各种类型的节点数占节点总数的比例,而类型采样是指从系统中均匀随机地采样不同类型的节点.比例估计为类型采样提供输入,类型采样则用于保证路由效率.有的应用可能只需要系统中某类节点所占的比例值,而不需要对所有类型进行均匀随机采样.TypeSampler 可以直接满足这类应用的需求,并可以在不影响比例估计的同时关闭类型采样功能,以节约存储和通信开销.

TypeSampler 进行比例估计的基本思想是,利用大数定律,即根据均匀随机采样得到的节点样本中的各类型节点的比例估计整个系统中各类型的比例.只要样本规模足够大,TypeSampler 就能以较大的概率得到各类型实

际比例值的合适的近似值.TypeSampler 采用已有的基于视图交换的节点采样方法[20]来获取节点样本,但是由于视图大小有限导致节点每次获取的节点样本规模有限,因而直接采用基于视图交换的节点采样方法来获取大量节点采样会带来较大的时间延迟和通信开销.所以,为了以较小的通信开销快速提高类型比例估计值的精度,TypeSampler 还利用基于 gossip 的反熵聚集来使节点间快速共享各自单独采样的结果.尽管每个节点单独获取的样本规模不大,但将它们累加起来也能够达到相当的数量,足以保证类型比例估计值的精度.

为了实现类型采样,TypeSampler 在每个节点上维护一个类型采样表.每个节点周期性地与视图中的某个随机节点交换类型采样请求(type sampling request,简称 TSR),其中包含节点所属某类型(称为采样类型)、该类型比例估计值、节点标识、节点所属所有类型等信息,而节点收到 TSR 后则基于一定的接受概率用该 TSR 来更新自己的类型采样表.为了实现对系统中所有类型的均匀随机采样,TSR 接受概率由采样类型的比例估计值以及发出 TSR 节点的类型数等信息来决定.通过选择合适的 TSR 接收概率,TypeSampler 不仅能够保证对类型的均匀随机采样,而且还能保证属于某个特定类型的不同节点能够以同样的概率在类型采样表中代表该类型,前者有助于保证路由性能,而后者则有助于保证同一类型节点间的负载平衡.

3 TypeSampler 描述

TypeSampler 主要包括两种算法:比例估计算法和采样更新算法.比例估计算法以节点采样服务维护的视图为输入来估计各类型比例,而采样更新算法则利用比例估计算法提供的各类型比例估计值来维护各节点的类型采样表,从而实现类型采样.

3.1 比例估计算法

快速计算精确的比例估计值的关键在于在较短时间内获取大量的节点样本.每个节点从自己的视图中获取的节点样本数只能随着时间呈线性增长,但是,如果多个节点同时进行采样并将结果集中起来,那么就可以起到倍增的效果.节点采样可以用基于周期性视图交换的方法[20]实现,但比例估计算法还需要被采样节点的所属类型信息.因此,在为 TypeSampler 实现节点采样时,需要将节点的类型信息与节点标识一起加入到节点的局部视图中,而这些视图所维护的邻居关系实际上构成了一个非结构化覆盖网.在本文中,我们假定已存在这样的节点采样服务为比例估计算法提供输入,而不描述其实现细节.

为使针对任意类型比例的估计都能在短时间内获取大规模的节点样本,每个节点不仅负责估计自己所属类型的比例,而且还负责估计若干随机选择的类型的比例值.如此,任一类型都会由相当数量的节点负责估计比例,而这些节点共同收集的大规模节点样本也就能保证该类型比例估计值的精度.在本文中,若某节点负责估计某类型的比例,则称该节点关注该类型.在 TypeSampler 中,每个节点的关注类型的具体选择方式如下:每个节点关注自己所属的各类型;假设系统中所有可能类型有 R 个,它们根据标识大小组成一个环,每个节点首先利用哈希函数将自己的节点标识映射到这个环中的某一类型,然后将以该类型开始的环上的 rR($r\in(0,1]$,称为关注率)个类型作为除本节点所属类型之外的关注类型.每个节点的关注率均为 r,因此每个节点的关注类型数大于等于 rR.因为节点采样服务提供的节点条目中含有该节点的标识及所属类型信息,所以其他节点无需与之通信便可以利用条目中的这些信息计算出该节点所关注的类型.

图 1 显示了 TypeSampler 的比例估计算法的伪代码.比例估计算法在每个节点上周期性地执行,每次执行又包括两个阶段:采样阶段、聚集阶段.针对这两个阶段,比例估计算法分别保存两套比例估计值,即 *initial_proportions* 和 *proportions*.两者都由节点关注的类型对应的条目组成,*initial_proportions* 条目的 *weight* 值表示样本个数,而 *proportions* 则作为正式的比例估计值提供给类型采样或其他应用使用.为保证节点采样的质量,比例估计算法执行的周期间隔应大于等于节点采样服务的执行周期间隔.在采样阶段,每个节点根据自己的节点采样更新自己关注类型的比例估计值(步骤 1~步骤 6).在聚集阶段,每个节点与随机选择的节点交换彼此共同关注的类型对应的比例估计值,并用平均值分别更新自己的估计,即进行反熵聚集(步骤 11~步骤 19).每隔 T 个周期,算法用采样阶段维护的 *initial_proportions* 中的比例估计值更新聚集阶段维护的 *proportions* 中的对应估计值,然后再将 *initial_proportions* 清零(步骤 7~步骤 10).也就是说,采样阶段与聚集阶段都以 T 为周期循环执行,且每

个采样阶段的最终值即为下一个聚集阶段的初值.采样阶段与聚集阶段循环执行的目的在于适应动态的网络环境,以使系统的最新状态能够及时反映到比例估计值中.在本文中,称采样阶段与聚集阶段的循环周期 T 为估计周期.随着在聚集阶段不断交换并平均估计值,每个节点维护的类型比例估计值将逐步收敛,而收敛值实际上等于利用关注该类型的各节点的节点采样集合之并计算出的比例估计值,具体分析见第 4.1 节.

```
比例估计算法
cycle:当前周期
T:采样阶段及聚集阶段的周期数,即估计周期
semantic_view:由元组⟨node,types⟩组成的节点采样集合,由节点采样服务更新
concerned_types:节点关注的类型集合
initial_proportions:由元组⟨type,value,weight⟩组成的初步比例估计值集合,根据节点自己的节点采样维护
proportions:由元组⟨type,value⟩组成的比例估计值集合
1  for each t∈concerned_types
2    frequency←t 在 semantic_view 中的出现次数
3    e←initial_proportions 中 type 值为 t 的条目
4    e.value←(e.value*e.weight+frequency)/(e.weight+|semantic_view|)
5    e.weight←e.weight+|semantic_view|
6  end for
7  if (cycle%T==0) then
8    用 initial_proportions 中条目的 value 值更新 proportions 中 type 相同的条目的 value 值
9    将 initial_proportions 中所有条目的 value 及 weight 清零
10 end if
11 q←getRandomItem(semantic_view)
12 common_types←q.node 所关注的类型集合与 concerned_types 的交集
13 send_buffer←common_types 对应的 proportions 中的条目集合
14 sendTo(q.node,send_buffer)
15 recvFrom(q.node,recv_buffer)
16 for each v∈recv_buffer
17   u←proportions 中 type 为 v.type 的条目
18   u.value←(u.value+v.value)/2
19 end for
20 cycle←cycle+1
```

Fig.1 Proportion estimation algorithm

图 1 比例估计算法

为了实现最大限度地聚集各节点单独采样的结果,比例估计算法的执行在各节点应是同步的,即各节点应保持同样的 *cycle* 值.在动态的网络环境中,新加入节点首先需要加入节点采样服务,因而它可以将自己的初始化视图中节点的 *cycle* 值的最大者作为自己的 *cycle* 值;已加入的节点则可以在进行比例估计值的聚集交互过程中附加上自己的 *cycle* 值,然后以两者中的较大者更新自己的 *cycle* 值,这实际上也是基于反熵 gossip 的同步方式,而基于反熵 gossip 的通信能够在节点总数为 N 的系统内以 $O(\log N)$个周期实现节点间的同步[26].

3.2 采样更新算法

TypeSampler 的采样更新算法在节点采样服务以及比例估计算法的基础上维护每个节点的 TST,从而实现类型采样.简而言之,节点周期性地与节点采样服务提供的一个随机节点交换各自的TSR,然后,两节点根据对方TSR 的内容决定是否用该 TSR 更新自己的 TST.在本文中,如果节点决定以收到的某 TSR 更新自己的 TST,则称该节点接受了该 TSR.TSR 是一个四元组,即⟨采样类型,采样类型的比例估计值,节点标识,节点所属类型集合⟩,其中,采样类型是从节点所属类型集合中随机选择的,它表示节点作为该类型的代表以供采样;而节点标识则由节点的 IP 地址及端口组成.TST 则是由三元组⟨采样类型,节点标识,节点所属类型集合⟩组成的集合,每个节点的 TST 大小都相等.类型采样的目的实际上就是使各种类型能够以同样的概率出现在 TST 的采样类型字段,而这时,采样类型及对应的节点标识便可以用来引导路由.

如果每个节点只有一个所属类型,那么由于节点采样服务提供的是对节点的均匀随机采样,因此某种类型的比例越大,节点收到以该类型作为采样类型的 TSR 的概率也就越大.在这种情况下,为了实现对类型的均匀随机采样,收到TSR 的节点应当以正比于该TSR 中比例估计值倒数的概率接受该TSR.但在实际应用中,每个节点

往往同时属于多种类型,其 TSR 中的采样类型需要从多个类型中随机选择,因而节点所属类型数目的差异会影响类型采样的均匀性.所以,针对这种一般的情况,收到 TSR 的节点在计算 TSR 接受概率时必须考虑发出 TSR 的节点的类型数目,即接受概率不仅要正比于 TSR 中的比例估计值倒数,同时还要正比于发出 TSR 的节点所属类型数.

图 2 显示了 TypeSampler 的采样更新算法的伪代码.采样更新算法也在每个节点上周期性地执行.TSR 中的采样类型从节点所属类型中随机选择,其对应的比例估计值则从比例估计算法提供的 *proportions* 中获取.收到 TSR 的节点分 3 种情况处理 TSR:首先检查自己的 TST 中是否已有 TSR 的采样类型对应的条目,若存在,则根据概率 $P_1=|TSR.types|/K_{max}$ 用 TSR 的内容改写原条目;然后检查自己的 TST 是否还有空位,若有,则根据 TSR 的内容创建一个新的 TST 条目;最后,根据概率 $P_2=(|TSR.types|/K_{max})\times(P_{min}/TSR.proportion)$ 用 TSR 中的内容改写 TST 中的任一条目.只要比例估计值足够精确,P_1 和 P_2 不仅能够保证各种类型以同样概率出现在 TST 的采样类型字段,而且还能保证属于某个类型的不同节点的标识能够以同样概率出现在系统中所有该类型对应的 TST 条目中,具体分析见第 4.2 节.

```
采样更新算法
Kmax:系统中单个节点所属类型数目的最大值
Pmin:系统中各类型比例的最小值
s:节点类型采样表的大小上限
self_id:本节点的标识
self_types:本节点所属类型集合
semantic_view:由元组⟨node,types⟩组成的节点采样集合
proportions:由元组⟨type,value⟩组成的比例估计值集合
TST:由元组⟨type,node,types⟩组成的集合,类型采样表
TSR:元组⟨type,proportion,node,types⟩,类型采样请求
1  t←getRandomType(types)
2  p←proportions.getValueByType(t)
3  TSR←⟨t,p,self_id,self_types⟩
4  q←getRandomItem(semantic_view)
5  sendTo(q.node,TSR)
6  recvFrom(q.node,TSR)
7  if (TST.containType(TSR.type) && nextRandomFloat()<|TSR.types|/Kmax) then
8     用⟨TSR.type,TSR.node,TSR.types⟩替换 TST 中 type 与 TSR.type 相同的条目
9  else if (|TST|<s) then
10    TST←TST∪{⟨TSR.type,TSR.node,TSR.types⟩}
11 else if (nextRandomFloat(·)<(|TSR.types|/Kmax)×(Pmin/TSR.proportion)) then
12    用⟨TSR.type,TSR.node,TSR.types⟩替换 TST 中随机选择的某一条目
13 end if
```

Fig.2 Sample update algorithm

图 2 采样更新算法

算法中,K_{max} 与 P_{min} 的作用只是保证 P_1 和 P_2 小于等于 1.节点可以通过 gossip 通信同步它们的值,其实现可以整合到比例估计算法的聚集阶段的交互中.但在拥有足够的全局知识的情况下,K_{max} 与 P_{min} 也可以用合适的常数替代.由于 TST 中包含对不同类型的均匀采样,以不同类型为目标的数据/查询能够以同样的概率在 TST 中发现以该目标类型为采样类型的条目,因此即便目标是稀有类型,TypeSampler 也能够保证路由的基本性能.由于节点采样服务的视图 *semantic_view* 以及 TST 还含有节点所属所有类型的信息,所以,TypeSampler 还可以利用这些信息进一步提高数据/查询在任意节点命中目标类型的概率.也就是说,当消息 *m* 路由到某节点时,该节点可以在 *semantic_view* 以及 TST 中查找是否有节点属于消息 *m* 的目标类型,如果有,则将消息 *m* 转发给该节点;否则,将消息 *m* 转发给 *semantic_view* 中的随机节点以继续这一过程.通过 *semantic_view* 以及 TST 的引导,基于随机行走的路由所需平均跳步数可以大为降低,路由效率的具体分析见第 4.3 节.

3.3 动态适应性

TypeSampler 没有专门的节点失效处理机制,当节点需要退出时只需直接退出,其他节点直接将其作为失效

节点看待;TypeSampler 也没有专门的机制处理节点所属类型的变化.TypeSampler 主要利用节点采样服务来处理节点的失效以及类型的变化.一方面,节点采样服务利用节点条目中附加的年龄信息,在周期性的视图交换中自动剔除失效的节点;另一方面,当节点所属类型发生变化时,它同样通过节点采样服务传播这一变化.即,在与其他节点交换视图时更新自己的类型信息,以通知系统中的其他节点.节点采样服务维护的视图是比例估计算法的输入,而比例估计算法输出的各类型比例估计值又是采样更新算法的输入.因此,节点失效以及类型变化都能通过节点采样服务的视图更新反映到 TypeSampler 的比例估计及类型采样中.

当新节点加入系统时,首先需要加入节点采样服务,即,新节点从已在系统中的某节点发起多个随机行走,然后用各随机行走终止节点初始化自己的视图,并将代表自己的条目加入到这些终止节点的视图中.在这个过程中,新节点用初始视图中各节点的 *cycle* 值的最大者设置自己的 *cycle* 值.一旦新节点加入了节点采样服务,无需特定的初始化过程,比例估计算法及采样更新算法便可开始执行.

4 理论分析

在本节,我们从理论上分析 TypeSampler 的比例估计算法、采样更新算法的性能以及它们所保证的路由效率,并比较分析 TypeSampler 与 TERA 的存储及通信开销.对于比例估计算法,我们关注的性能包括收敛性与精确性.对于采样更新算法,我们关注的性能包括类型采样的均匀随机性与对同类型节点采样的均匀随机性,前者是指各类型是否在 TST 的采样类型字段等概率出现,后者是指同类型的不同节点是否在 TST 中以该类型作为采样类型的条目中等概率出现.

4.1 比例估计算法

比例估计算法的采样阶段可以视作是在进行一系列独立的伯努利实验,即节点针对其关注的某类型 i,检查每个被采样节点 Y 可能属于或不属于它.令

$$X=\begin{cases}1, & \text{若}Y\text{属于类型}i\\ 0, & \text{其他}\end{cases}.$$

如果节点 Y 是从系统中均匀随机选取的,那么 X=1 的概率恰好就是属于类型 i 的节点在系统中所占的比例 p,即 $\Pr(X=1)=p$.每个节点在执行比例估计算法时,实际上就是通过采样节点以针对自己关注的每个类型进行实验.

引理 1. 令 *item* 是 *initial_proportions* 中关于某一类型的条目,X_j 为针对该类型的第 j 次实验的 X 值,c 为节点采样服务的视图大小,T 为估计周期长度,m_k 为第 k 轮结束时的 *item.weight* 的值,Z_k 为第 k 轮结束时 *item.value* 的值,其中,$1\leqslant k\leqslant T$,则有 $m_k=kc$, $Z_k=\dfrac{\sum_{j=1}^{m_k}X_j}{m_k}$.

证明:由比例估计算法步骤 5 易得 $m_k=kc$.当 k=1 时,有 $Z_1=\dfrac{\sum_{j=1}^{m_1}X_j}{m_1}$.假设 $k=l(1\leqslant l<T)$时有 $Z_l=\dfrac{\sum_{j=1}^{m_l}X_j}{m_l}$, 由比例估计算法步骤 4 可得:

$$Z_{l+1}=\frac{Z_l\times m_l+\dfrac{\sum_{j=m_l+1}^{m_l+c}X_j}{c}\times c}{m_l+c}=\frac{\dfrac{\sum_{j=1}^{m_l}X_j}{m_l}\times m_l+\dfrac{\sum_{j=m_l+1}^{m_l+c}X_j}{c}\times c}{m_l+c}=\frac{\sum_{j=1}^{m_{l+1}}X_j}{m_{l+1}}.$$

于是得到 $Z_k=\dfrac{\sum_{j=1}^{m_k}X_j}{m_k}$, $1\leqslant k\leqslant T$.引理得证. □

根据引理 1,有 $Z_T=\frac{\sum_{j=1}^{cT}X_j}{cT}$,即,在估计周期结束时,*initial_proportions* 中的比例估计值是基于 *cT* 个节点采样计算得到的.

引理 2. 令 *c* 为节点采样服务的视图大小,*M* 为系统中关注某一类型的节点数,X_{jk} 为其中第 *j* 个节点针对该类型的第 *k* 次实验的 *X* 值,Z_j 为其中第 *j* 个节点的 *initial_proportions* 在估计周期结束时关于该类型的比例估计值,则有 $\frac{\sum_j Z_j}{M}=\frac{\sum_{jk}X_{jk}}{McT}$.

证明:由引理 1 可知 $Z_j=\frac{\sum_{k=1}^{cT}X_{jk}}{cT}$,则显然有 $\frac{\sum_j Z_j}{M}=\frac{\sum_{jk}X_{jk}}{McT}$ 成立.引理得证. □

引理 2 表明,关注同一类型的 *M* 个节点的 *initial_proportions* 中关于该类型的比例估计值的平均值,可以看作是基于 *McT* 个节点采样计算的.若系统节点规模为 *N*,类型总数为 *R*,每个节点除自己所属类型外还随机选择 *rR* 个类型关注,即关注率为 *r*,则平均关注某个类型的节点数至少为 *Nr*.因此,当关注同一类型的节点共享它们的节点采样之后,平均而言,它们对该类型的比例估计便可以基于至少 *NrcT* 个节点采样计算,而更多的采样也就意味着更高的精度.

比例估计算法的聚集阶段实际上是试图使得关注某类型的所有节点获取它们各自比例估计值的平均值,根据引理 2,也就是说,获取在它们所有节点采样基础上估计的比例值.现在的问题是,比例估计算法中需要多少个周期才能使这些节点的估计值收敛于平均值,为此我们有以下结果.

引理 3[22]. 若每个节点每个周期从系统中随机选择一个节点交换彼此的聚集数值,并用两者的平均值更新自己的聚集数值,令 $E(\sigma_k^2)$ 为第 *k* 轮次时所有节点聚集数值的样本方差的期望,则有 $E(\sigma_{k+1}^2)\approx\frac{1}{2\sqrt{\mathrm{e}}}E(\sigma_k^2)$.

定理 1. 令 *c* 为节点采样服务的视图大小,*T* 为估计周期长度,*r* 为关注率,*p* 为类型 *i* 的实际比例值,*M* 为关注类型 *i* 的节点数,$E(\sigma_k^2)$ 为估计周期中第 *k* 轮次关注类型 *i* 的节点的 *proportions* 中对类型 *i* 的比例估计值的样本方差的期望,其中,$1\leqslant k\leqslant T$,则有 $E(\sigma_T^2)\approx\left(\frac{1}{2\sqrt{\mathrm{e}}}\right)^{TM/N}\frac{1}{cT}p(1-p)$.

证明:设 Z_{j1} 为节点 *j* 的聚集阶段的关于类型 *i* 的比例估计初值,根据比例估计算法以及引理 1 可知,Z_{j1} 是上一个采样阶段的结果,即 $Z_{j1}=\frac{\sum_{k=1}^{cT}X_{jk}}{cT}$.$\sum_{k=1}^{cT}X_{jk}$ 实际上是参数为 *cT* 和 *p* 的二项随机变量,其方差为 *cTp*(1–*p*),因此,聚集阶段节点关于类型 *i* 的比例估计初值的样本方差的均值为 $\frac{1}{cT}p(1-p)$, 即 $E(\sigma_1^2)=\frac{1}{cT}p(1-p)$.

当节点与随机选择的节点交换 *proportions* 中的条目时,对方有 *M*/*N* 的概率也关注类型 *i*.因此在 *T* 个周期中,每个节点平均发起和接收 *TM*/*N* 次关于类型 *i* 比例估计值的聚集交互.

根据引理 3,可得 $E(\sigma_T^2)\approx\left(\frac{1}{2\sqrt{\mathrm{e}}}\right)^{TM/N}\frac{1}{cT}p(1-p)$.定理得证. □

当各节点关于某类型的比例估计值样本方差趋于 0 时,则它们的比例估计值也就趋于其平均值.由引理 2 可知,此时,它们的估计值近似等价于基于所有这些节点的采样计算出的比例估计值.因此,为了充分共享关注同一类型的节点各自获取的节点样本,估计周期的长度必须能够保证各节点的比例估计值样本方差最终近似为 0,而定理 1 恰恰可以用来估算必要的估计周期大小.假设类型总数 *R* 为 100,每个节点所属的类型数在 5~15 之间均匀随机选取,而节点所属每个类型按照参数为 1.0 的 Zipf 分布随机选择,下面分两种情况对定理 1 进行定量分析.

表 1 显示了其他参数固定而系统节点规模 *N* 不同时,在聚集阶段各类型比例估计值样本方差的期望的平均

值随视图交换周期增加而变化的情况.其中,关注率 r=0.1,节点采样服务的视图大小 c=20,估计周期 T=100,而表中各行则对应不同系统节点规模 N.由于 $\frac{1}{cT}p(1-p)$ 都相等,因此各种 N 对应的方差期望的均值在聚集阶段的初值都相等;由于关注率 r 以及类型分布相同,因此各种 N 对应的方差期望的均值在聚集阶段的变化也大致相同.由表 1 可知,80 个周期后,方差期望的均值均减为 10^{-8} 量级,这意味着各节点关于各类型的比例估计值与相应均值的偏差约为 10^{-4} 量级,而由于类型比例的最小值为 10^{-2} 量级,因而此时各类型的比例估计值与相应均值的相对偏差最大约为 10^{-2} 量级,近似地,可以认为此时聚集过程已收敛.以上分析表明,对于前述参数设置,估计周期 T=100 大致能够满足聚集收敛的要求.

Table 1 Average of the expectation of the proportion estimation sample deviation (variable N)

表 1 比例估计值样本方差的期望的平均值(N 变化)

r=0.1, c=20, T=100	*Cycle*										
	0	10	20	30	40	50	60	70	80	90	100
N=1000	3.6e–05	1.5e–05	6.1e–06	2.8e–06	1.3e–06	6.1e–07	3.1e–07	1.5e–07	7.9e–08	3.9e–08	2.0e–08
N=2000	3.6e–05	1.5e–05	6.4e–06	2.9e–06	1.3e–06	6.3e–07	3.0e–07	1.5e–07	6.9e–08	3.6e–08	1.8e–08
N=3000	3.6e–05	1.5e–05	6.2e–06	2.8e–06	1.3e–06	6.3e–07	3.0e–07	1.5e–07	7.4e–08	3.8e–08	1.9e–08
N=4000	3.6e–05	1.5e–05	6.1e–06	2.8e–06	1.3e–06	6.4e–07	3.0e–07	1.5e–07	7.5e–08	3.9e–08	2.0e–08

表 2 显示了系统节点规模 N 固定而其他参数不同时,在聚集阶段各类型比例估计值样本方差的期望的平均值随视图交换周期增加而发生变化的情况.其中,系统节点规模 N=1000,而表中各行则对应 r,c,T 的不同组合.由于 $\frac{1}{cT}p(1-p)$ 决定方差期望的均值在聚集阶段的初值,因此 cT 越大,则初值越小;而关注率 r 影响 M 的大小,进而影响方差期望减小的速度,因而,当 r=0.2,方差期望的均值在 50 个周期时即降至 10^{-8} 量级.特别地,对于 T=50 的情况,50 个周期后方差期望的均值为 10^{-6} 量级.由于类型比例的最小值为 10^{-2} 量级,因而此时类型的比例估计值与相应均值的相对偏差最大约为 10^{-1} 量级,其较差的收敛效果实际上源自其较大的初值和较短的收敛时间.以上分析表明,对于前述参数设置,估计周期 T=100 大致能够满足聚集收敛的要求.

Table 2 Average of the expectation of the proportion estimation sample deviation (invariable N)

表 2 比例估计值样本方差的期望的平均值(N 不变)

N=1000	*Cycle*										
	0	10	20	30	40	50	60	70	80	90	100
r=0.1, c=20, T=50	7.2e–05	3.0e–05	1.2e–05	5.6e–06	2.6e–06	1.2e–06					
r=0.1, c=20, T=100	3.6e–05	1.5e–05	6.1e–06	2.8e–06	1.3e–06	6.1e–07	3.1e–07	1.5e–07	7.9e–08	3.9e–08	2.0e–08
r=0.2, c=20, T=100	3.6e–05	9.7e–06	2.5e–06	7.0e–07	1.9e–07	5.7e–08	1.7e–08	5.3e–09	1.6e–09	5.1e–10	1.5e–10
r=0.1, c=80, T=100	8.9e–06	3.8e–06	1.5e–06	7.0e–07	3.3e–07	1.5e–07	7.6e–08	3.8e–08	2.0e–08	9.7e–09	4.9e–09

实际上,由于每个节点除了随机选择 rR(R 为类型总数)个类型关注之外,还关注自己所属的类型,因此有 $E(M)\geqslant Nr$.所以,在使用定理 1 时也可用 $\left(\frac{1}{2\sqrt{\mathrm{e}}}\right)^{Tr}\frac{1}{cT}p(1-p)$ 代替 $\left(\frac{1}{2\sqrt{\mathrm{e}}}\right)^{TM/N}\frac{1}{cT}p(1-p)$ 以进行近似计算.现在的问题是,当估计周期长度足以保证关注同一类型的节点充分共享它们的节点样本时,关于该类型的比例估计值的精度究竟如何.为此,我们有以下结果.

定理 2. 令 N 为系统节点规模,c 为节点采样服务的视图大小,r 为关注率,M 为关注类型 i 的节点数,p 为类型 i 的实际比例值.若在估计周期 T 的时间长度内,关注类型 i 的节点的 *proportions* 中相应的比例估计值收敛于

$\hat{p}$,则有 $\Pr(|\hat{p}-p|\geqslant\delta p)\leqslant 2e^{-McTp\delta^2/3}$.

证明:由引理 2 可知, $\hat{p}=\dfrac{\sum_{jk}X_{jk}}{McT}$,应用切尔诺夫界,则有:

$$\Pr\left(\left|\sum_{jk}X_{jk}-E\left(\sum_{jk}X_{jk}\right)\right|\geqslant\delta E\left(\sum_{jk}X_{jk}\right)\right)\leqslant 2e^{-E\left(\sum_{jk}X_{jk}\right)\delta^2/3},$$

$$\Pr\left(\left|\sum_{jk}X_{jk}-McTp\right|\geqslant\delta McTp\right)\leqslant 2e^{-McTp\delta^2/3},$$

$$\Pr\left(\left|\frac{\sum_{jk}X_{jk}}{McT}-p\right|\geqslant\delta p\right)\leqslant 2e^{-McTp\delta^2/3},$$

$$\Pr(|\hat{p}-p|\geqslant\delta p)\leqslant 2e^{-McTp\delta^2/3}.$$

定理得证. □

假设类型总数 R 为 100,每个节点所属的类型数在 5~15 之间均匀随机地选取,而节点所属每个类型按照参数为 1.0 的 Zipf 分布随机选择,下面分两种情况对定理 2 进行定量分析.

表 3 显示了其他参数固定而系统节点规模 N 不同时,聚集过程开始时以及收敛后的各类型比例估计值的相对误差上界的平均值,每种类型对应上界的置信水平均为 0.99.由于 c,T 相同,不同 N 情况下单个节点在采样阶段收集的节点样本数均相同,因此,各种 N 情况下的聚集开始时相对误差上界的平均值大致相同.之所以略有差别,是因为这里的相对误差上界平均值是根据各种类型的估计值个数加权计算,而不同 N 情况下各类型估计值个数的分布又不完全相同.由于关注率 r 一定,那么,N 越大,则关注同一类型的节点越多,聚集收敛后比例估计值对应的节点样本数也就越大,因此,随着 N 的增大,聚集收敛后相对误差上界的平均值也从 2.5%下降到 1.3%.

Table 3 Average of the relative error upper bound of the proportion estimation (variable N)

表 3 比例估计值的相对误差上界的平均值(N 变化)

r=0.1, c=20, T=100	N			
	1 000	2 000	3 000	4 000
聚集开始时相对误差上界的平均值(%)	31.8	32.0	31.7	31.9
聚集收敛后相对误差上界的平均值(%)	2.5	1.8	1.4	1.3

表 4 显示了系统节点规模 N 固定而其他参数不同时,聚集过程开始时以及收敛后的各类型比例估计值的相对误差上界的平均值,每种类型对应上界的置信水平均为 0.99.由于 cT 越大,则单个节点在采样阶段收集的节点样本数越大,因此,对应的聚集开始时相对误差上界的平均值越小;由于 rcT 越大,则聚集收敛后比例估计值对应的节点样本数越大,因此,对应的聚集收敛后相对误差上界的平均值也就越小.

Table 4 Average of the relative error upper bound of the proportion estimation (invariable N)

表 4 比例估计值的相对误差上界的平均值(N 不变)

N=1000	r=0.1, c=20, T=50	r=0.1, c=20, T=100	r=0.2, c=20, T=100	r=0.1, c=80, T=100
聚集开始时相对误差上界的平均值(%)	4.9	3.8	3.7	3.9
聚集收敛后相对误差上界的平均值(%)	3.5	.5	.2	.2

由表 3 及表 4 可知,关注同一类型的节点通过聚集过程使得收集的节点样本数实质上从 cT 增长到 McT,于是,相应的比例估计值的相对误差也就会大幅降低.由于 $E(M)\geqslant Nr$,因此可用 $NrcT$ 作为 McT 的近似值.根据定理 2,我们可有如下近似的推论:当聚集过程收敛时,$NrcT$ 越大,则比例估计值越精确.于是,我们结合定理 1 可知,T 越大就越能够保证聚集过程收敛,而且最终的比例估计值也越精确.但是由于聚集阶段的初值使用的是上一个采样阶段获取的数据,同时,聚集过程本身并没有引入新的节点采样,T 过大,则会导致比例估计值不能及时反映系统的动态变化,因此,T 的数值需要根据具体的应用环境权衡选择.另一方面,近似而言,如果聚集过程收敛且其他

参数一定,那么系统节点规模越大,比例估计值也越精确,这反映出比例估计算法具有良好的可扩展性.

4.2 采样更新算法

类型采样的均匀随机性是保证路由性能的基础,而对同类型节点采样的均匀随机性,则有助于保证同类型节点间的负载平衡.关于这两种均匀随机性,我们有以下结果.

引理 4. 若比例估计算法提供的类型比例估计值等于实际值,那么采样更新算法能够保证以任何不在当前 TST 采样类型字段的类型作为采样类型的 TSR 被接受的概率相等.

证明:设任意类型 i 的实际比例值为 p.由于交换 TSR 的节点对是从系统中随机选择的,因此对收到 TSR 的节点 B 来说,发送该 TSR 的节点 A 属于类型 i 的概率为 p.若节点 A 属于类型 i 且所属类型数为 k,则类型 i 被选择作为其 TSR 采样类型的概率为 $1/k$.根据采样更新算法,在类型 i 不在节点 B 的 TST 采样类型字段的情况下,节点 B 接受该 TSR 的概率为$(k/K_{\max})\times(P_{\min}/\hat{p})$,其中,$K_{\max}$ 与 $P_{\min}$ 为常数, $\hat{p}$ 为节点 A 关于类型 i 的比例估计值.因此,节点 B 将类型 i 加入自己的 TST 的概率 $P_{accept}=p\times\frac{1}{k}\times\frac{k}{K_{\max}}\times\frac{P_{\min}}{\hat{p}}$.当 $\hat{p}=p$ 时,有 $P_{accept}=\frac{P_{\min}}{K_{\max}}$.

引理得证. □

定理 3. 若比例估计算法提供的类型比例估计值等于实际值,那么采样更新算法能够保证无论任意节点的 TST 的初始状态如何,任意类型在该 TST 采样类型字段出现的概率均随算法执行周期的增加而收敛于 s/R,其中,s 为每个节点 TST 的大小,R 为类型总数.

证明:设 $p_i^{(k)}$ 为类型 i 在任意节点处理第 k 个 TSR 后出现于其 TST 采样类型字段的概率, $p_i^{(0)}$ 为初始概率.由引理 4 可知,在每次处理 TSR 的过程中,若类型 i 已在 TST 的采样类型字段,则它被某类型 j 取代的概率为 $(1-p'_j)\frac{P_{\min}}{K_{\max}}\frac{1}{s}$,其中, p'_j 表示类型 j 在类型 i 已在 TST 的采样类型字段条件下也存在于 TST 的采样类型字段的条件概率;若类型 i 不在 TST 的采样类型字段,则它被更新进 TST 的概率为 $\frac{P_{\min}}{K_{\max}}$.于是有,

$$p_i^{(k+1)}=p_i^{(k)}\left[1-\frac{P_{\min}}{K_{\max}}\frac{1}{s}\sum_{j\neq i}(1-p'_j)\right]+(1-p_i^{(k)})\frac{P_{\min}}{K_{\max}}.$$

因为 $\sum_{j\neq i}p'_j=s-1$, 所以有 $p_i^{(k+1)}=p_i^{(k)}\left[1-\frac{P_{\min}}{K_{\max}}\frac{1}{s}(R-s)\right]+(1-p_i^{(k)})\frac{P_{\min}}{K_{\max}}=\left(1-\frac{P_{\min}}{K_{\max}}\frac{R}{s}\right)p_i^{(k)}+\frac{P_{\min}}{K_{\max}}$.

令 $\alpha=\frac{P_{\min}}{K_{\max}},\beta=1-\frac{P_{\min}}{K_{\max}}\frac{R}{s}$,则有 $p_i^{(k+1)}=\beta p_i^{(k)}+\alpha$,于是易得 $p_i^{(k)}=\beta^k p_i^{(0)}+\frac{\alpha(1-\beta^k)}{1-\beta}$.

因为 $P_{\min}R\leqslant K_{\max}$,所以 $0\leqslant\beta<1$,故有 $\lim_{k\to\infty}p_i^{(k)}=\frac{\alpha}{1-\beta}=\frac{s}{R}$.在采样更新算法的每个周期内,每个节点平均处理两个 TSR,即 k 平均增加 2,因此定理得证. □

假设类型总数 R 为 100,每个节点所属的类型数在 5~15 之间均匀随机地选取,而节点所属每个类型按照参数为 1.0 的 Zipf 分布随机选择,系统节点规模 N=1000,关注率 r=0.1,节点采样服务的视图大小 c=20,估计周期 T=100,TST 表大小 s=10,下面对定理 3 进行定量分析.

我们根据定理 3 证明中的 $p_i^{(k+1)}=\left(1-\frac{P_{\min}}{K_{\max}}\frac{R}{s}\right)p_i^{(k)}+\frac{P_{\min}}{K_{\max}}$ 来考察类型采样的收敛情况.节点的 TST 一开始为空,然后随着采样更新算法的执行而逐步被填充,而刚被填满时各类型在其采样类型字段中出现的概率约等于各类型实际比例,若设此时的 k=0,则最流行类型与最稀有类型分别对应于最大的 $p^{(0)}$和最小的 $p^{(0)}$.图 3 显示了最流行类型与最稀有类型对应的 $p^{(k)}$随 k 的变化情况.由图 3 可知,两种类型对应 $p^{(k)}$都随着 k 的增长而迅速收敛,且均当 k 为 500 左右时大致收敛于 0.1,即 s/R.由定理 3 的证明可知,在采样更新算法的每个周期中,k 平均增加 2,因此最流行类型与最稀有类型的类型采样收敛时间大约为 250 个周期.考虑到它们的收敛速度比其他类型要慢,因此整个系统的类型采样收敛时间最多大约为 250 个周期.

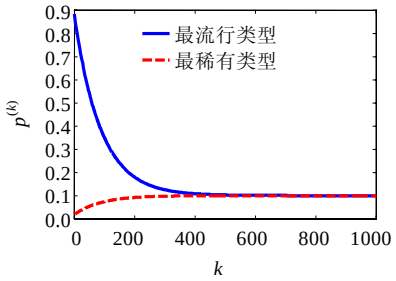


Fig.3 Convergence speed of the type sampling

图 3 类型采样的收敛速度

定理 4. 若比例估计算法提供的类型比例估计值等于实际值,那么采样更新算法能够保证在执行足够多周期后,属于任意类型的任意节点以该类型作为采样类型更新 TST 的概率相等.

证明:设系统节点规模为 N,任意类型 i 的实际比例值为 p,节点 A 属于类型 i 且所属类型数为 k,其关于类型 i 的比例估计值为 $\hat{p}$.当节点 B 接收 TSR 时,该 TSR 来自节点 A 的概率为 $1/N$,而节点 A 选择类型 i 作为采样类型的概率为 $1/k$,此时存在两种可能情况:(i) B 的 TST 中已存在以类型 i 作为采样类型的条目;(ii) B 的 TST 中不存在以类型 i 作为采样类型的条目.当定理 3 成立时,则在经过足够多个周期后,情况(i)发生的概率为 s/R,情况(ii)发生的概率为 $1-s/R$.

下面分别讨论这两种情况:

- 对于情况(i),根据采样更新算法,节点 A 被用来更新 TST 中对应条目的概率为 $k/K_{\max}$,因此,属于类型 i 的任意节点 A 替换 TST 中采样类型 i 的条目的概率为 $P_{substitute}=\frac{1}{N}\times\frac{1}{k}\times\frac{s}{R}\times\frac{k}{K_{\max}}=\frac{s}{NRK_{\max}}$;
- 对于情况(ii),根据采样更新算法,节点 A 被加入 TST 的概率为 $(k/K_{\max})\times(P_{\min}/\hat{p})$,因此,属于类型 i 的任意节点 A 与类型 i 一起加入 TST 的概率为 $P_{add}=\frac{1}{N}\times\frac{1}{k}\times\left(1-\frac{s}{R}\right)\times\frac{k}{K_{\max}}\times\frac{P_{\min}}{\hat{p}}=\left(1-\frac{s}{R}\right)\frac{P_{\min}}{NK_{\max}\hat{p}}$,当 $\hat{p}=p$ 时,有 $P_{add}=(1-\frac{s}{R})\frac{P_{\min}}{NK_{\max}p}$.

综上可得,属于任意类型 i 的任意节点 A 以类型 i 为采样类型更新 TST 的概率为

$$P_{accept}=P_{substitute}+P_{add}=\frac{sp+(R-s)P_{\min}}{NRK_{\max}p}.$$

对于属于类型 i 的任意节点来说,该概率都相等.定理得证. □

4.3 路由效率

定理 3 与定理 4 实际上保证了随着算法的执行,每个节点的 TST 能够成为对不同类型的均匀随机采样,而其中关于特定类型的条目能够成为对属于该类型的节点的均匀随机采样.而这正是 TypeSampler 为路由性能提供保证的基础.关于 TypeSampler 的路由效率,我们有以下结果.

定理 5. 令 c 为节点采样服务的视图大小,R 为类型总数,s 为每个节点的 TST 的大小,p 为类型 i 的实际比例值.若消息 m 以属于类型 i 的节点为目标利用节点采样服务在系统中随机行走,则当 TST 实现对类型的均匀随机采样时,消息 m 在任意节点发现属于类型 i 的节点的概率大于等于 $1-(1-p)^c\left(1-\frac{s}{R}\right)$,命中目标时路由跳数的期望小于等于 $1\Big/\left(1-(1-p)^c\left(1-\frac{s}{R}\right)\right)$.

证明:当 TST 实现对类型的均匀随机采样时,由于其条目中还含有节点除采样类型外其他所属类型的信息,因此,TST 中含有属于类型 i 的节点的概率大于等于 s/R.而 *semantic_view* 是对节点的均匀随机采样,因此,其中

任一节点属于类型 i 的概率为 p,而数据消息 m 同样可以利用 *semantic_view* 条目中的节点类型信息来检验有无目标节点.所以,m 无法从 TST 及 *semantic_view* 中发现目标节点的概率小于等于 $(1-p)^c\left(1-\frac{s}{R}\right)$,即消息 m 在任意节点发现属于类型 i 的节点的概率大于等于 $1-(1-p)^c\left(1-\frac{s}{R}\right)$,因而路由成功时的期望跳数小于等于 $1\Big/\left(1-(1-p)^c\left(1-\frac{s}{R}\right)\right)$.定理得证. □

由定理 5 可知,TST 的类型采样实际上保证以任意类型为目标的消息在每跳的命中概率至少为 s/R,而其命中时的路由跳数的期望至多为 R/s.

4.4 存储与通信开销

TypeSampler 的存储开销主要包括 *semantic_view*,TST,*initial_proportions*,*proportions*.在这些数据结构单个条目大小固定的情况下,它们所含的条目数决定了单个节点的存储开销.*semantic_view* 与 TST 中的条目数根据统一的参数设置,而 *initial_proportions*,*proportions* 中的条目数由每个节点关注的类型数决定.令 R 为类型总数,r 为关注率,k 为节点平均所属类型数,则每个节点至少关注 rR 个类型,最多关注 $k+rR$ 个类型,平均关注 $(1-r)k+rR$ 个类型.令 c 为 *semantic_view* 的条目数,s 为 TST 的条目数,E_{sv} 为 *semantic_view* 的条目大小,E_{TST} 为 TST 的条目大小,E_{ip} 为 *initial_proportions* 的条目大小,E_p 为 *proportions* 的条目大小,则每个节点的平均存储开销为 $cE_{sv}+sE_{\mathrm{TST}}+[(1-r)k+rR](E_{ip}+E_p)$.

TypeSampler 的通信开销包括节点采样服务的视图交换、比例估计算法的聚集交互、采样更新算法的 TSR 交换.为了保证节点采样的质量,节点采样服务的视图交换的周期应小于等于比例估计算法的执行周期.在本文中,我们只考虑视图交换周期、比例估计算法执行周期、采样更新算法执行周期三者相等的情况,因而不再区分它们的名称.节点每次交换视图只交换各自视图的一半,而在执行比例估计算法时交换的 *proportions* 条目数必小于等于自己的关注类型数.在每个周期中,每个节点分别发起一次视图、比例估计值及 TSR 交换,而且还作为接收者平均各响应 1 次前述交换.令 E_{TSR} 为 TSR 的大小,则每个节点在一个周期内传输的数据量平均值不超过 $2cE_{sv}+4[(1-r)k+rR]E_p+4E_{\mathrm{TSR}}$.

TERA 除了需要基于节点采样服务为每个节点维护一个全局视图以外,还需要为其每个所属类型分别维护一个视图.其主要的存储开销就是这些视图以及它的类型采样表 APT.令 c' 为 TERA 中每个视图含有的条目数,s' 为其 APT 含有的条目数,k 为节点平均所属类型数,E_v 为视图的条目大小,E_{APT} 为 APT 的条目大小,则 TERA 中每个节点的平均存储开销为 $(k+1)c'E_v+s'E_{\mathrm{APT}}$.TERA 的主要通信开销则包括视图的交换开销以及订阅列表的发送.其视图的维护过程与 TypeSampler 类似,订阅列表则由每个节点每隔若干周期发送给若干随机选择的邻居,其作用类似于 TSR.但与 TSR 不同,订阅列表含有多个条目,且条目数等于发送节点所属类型数.若订阅列表取与 TSR 同样的发送频率,则每个节点每周期需发送两次.令 E_{sub} 为订阅列表的条目大小,则 TERA 中的每个节点在每个周期内传输的数据量平均值为 $2(k+1)c'E_v+4kE_{sub}$.

由于不包含节点所属所有类型的信息,因此,E_v,E_{APT},E_{sub} 与 E_{sv},E_{TST},E_{TSR} 相比较小,但类型信息可以利用位向量压缩存储,因此实际上差距有限.而且由上述分析可知,相对于 TypeSampler,TERA 的存储及通信开销与节点所属类型数 k 密切相关,因此就 k 而言,TypeSampler 比 TERA 具有更好的可扩展性.

5 实验评价

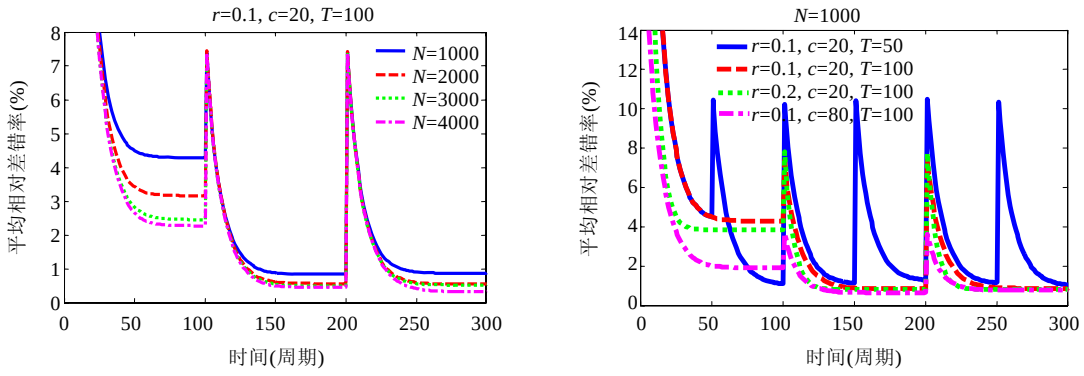
我们用 P2P 协议模拟器 PeerSim[27]实现 TypeSampler,并设置不同的场景进行性能评价,具体的实验内容包括比例估计、类型采样、路由性能、存储与通信开销.TERA 是已有的典型的类型采样方法,它基于节点规模估计实现类型采样.我们也用 PeerSim 实现了 TERA,并在评价 TypeSampler 的类型采样效果、路由性能以及存储与通信开销时与 TERA 进行比较.在实验中,TypeSampler 比例估计算法、采样更新算法以及节点采样服务的执行周期间隔设置为相同长度;TERA 所使用的节点采样服务的视图更新周期与 TypeSampler 的相同,其视图大小

c'与 TypeSampler 的视图大小 c 相同,其类型采样表 APT 的大小 s'也与 TypeSampler 的 TST 大小 s 相同,TERA 中每个节点在每个周期中向两个随机节点发送自己的订阅列表;该频率与 TypeSampler 中 TSR 的发送频率相同;类型总数 R 设为 100,每个节点所属类型数在 5~15 之间均匀随机取值,每个节点的所属类型均根据 Zipf 分布随机生成,即每个特定类型被选中的概率正比于 $i^{-\alpha}$,其中,i 为类型序号,α为 Zipf 参数,实验中取值为 1.0.

5.1 比例估计

在 TypeSampler 中,比例估计是进行类型采样的基础,因此,我们首先验证比例估计算法的性能.为了衡量类型比例估计值的精确性,我们定义平均相对差错率(mean relative error,简称 MRE),即 $MRE=\frac{1}{I}\sum_{i=1}^{I}\left|\frac{\hat{p}_i-p_i}{p_i}\right|$,其中,$\hat{p}_i$ 表示估计值,p_i 表示实际值,I 表示估计值的总个数.

图 4 显示了在静态网络环境中,不同参数设置对比例估计平均相对差错率的影响,其中,N 为系统节点规模,r 为关注率,c 为节点采样服务视图大小,T 为估计周期长度.由图 4 可知,各种情况下,比例估计的平均相对差错率都在迅速下降后产生了规则的波动,而波动的周期正好与比例估计算法的估计周期 T 相对应.平均相对差错率的波峰对应于估计周期开始每个节点根据自己的节点采样计算出的估计值,而波峰之后的下降则对应于估计值聚集的过程.因此,需要比例估计服务的应用可以根据当前周期以及 T 的大小来判断此时获取的比例估计值的精确度.由图 4 还可以看到:$NrcT$ 的值越大,图中平均相对差错率在估计周期中的最小值也越低,这与第 4.1 节的理论推论相一致;平均相对差错率在聚集阶段的收敛速度与第 4.1 节表 1 及表 2 中的理论聚集收敛速度相比要快,而聚集阶段初始时及收敛后的平均相对差错率也均小于第 4.1 节表 3 及表 4 中的理论上界.这表明,定理 2 的界比较宽松,也因此,具体的某种类型的相对差错率可以根据 $\delta\approx\sqrt{-3\ln(\beta/2)/(NrcT\hat{p})}$ 来估算,其中,$(1-\beta)$为置信水平,$\hat{p}$ 为该类型的比例估计值.当 N=1000,r=0.1,c=20,T=100 时,图中平均相对差错率在每个估计周期中均能迅速收敛到 1%以下,而其峰值也不超过 8%.第 5.2 节的实验结果将表明,这个精度足以保证实现较高均匀随机性的类型采样.值得注意的是,图 4(a)显示出,当其他参数固定,N 越大,则平均相对差错率的最小值也越低,这表明比例估计算法具有良好的可扩展性.



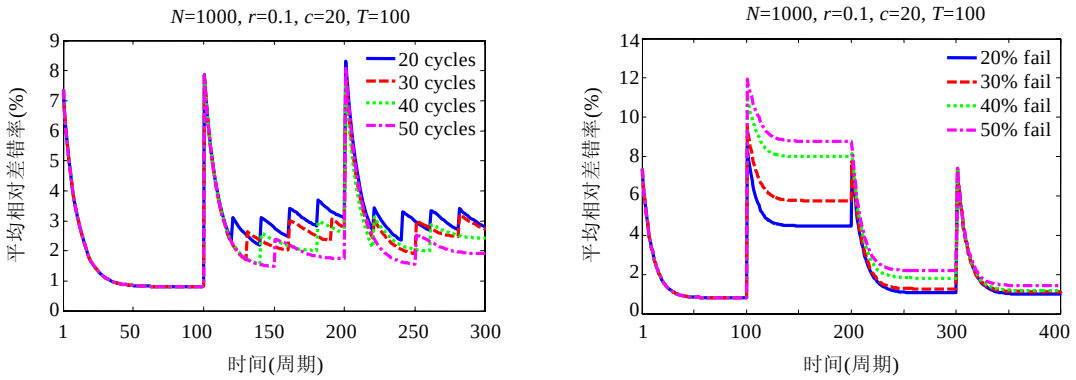
(a) N 变化 (b) N 不变

Fig.4 Proportion estimation in the static network environment

图 4 静态网络环境中的比例估计

图 5 显示了在动态网络环境中,不同参数设置对比例估计平均相对差错率的影响,其中,N=1000,r=0.1,c=20,T=100.图 5(a)显示了从第 100 个周期开始,每隔 20,30,40,50 个周期随机选择的 1%的在线节点被新节点替换情况下,比例估计的平均相对差错率的变化.由图 5 可知,随着节点被周期性地替换,平均相对差错率在估计周期内也发生了峰值不超过 4%的对应的周期性波动,而替换周期间隔越大,平均相对差错率也越小,这表明比例估计算法能够在节点波动的环境中持续工作.图 5(b)显示了在第 100 个周期随机选择的 20%,30%,40%,50%的节点瞬

时失效情况下,比例估计的平均相对差错率的变化.由图 5 可知,无论瞬时失效的节点数目为多少,经过两个估计周期的调整,平均相对差错率即恢复到静态环境中的水平,这表明比例估计算法具有很强的鲁棒性.值得注意的是,图 5(b)中,稳定后平均相对差错率的收敛值略高于失效前的相应值,其原因就在于节点失效后 $NrcT$ 变小了,而这与第 4.1 节的理论推论也是一致的.



(a) 节点替换 (b) 节点失效

Fig.5 Proportion estimation in the dynamic network environment

图 5 动态网络环境中的比例估计

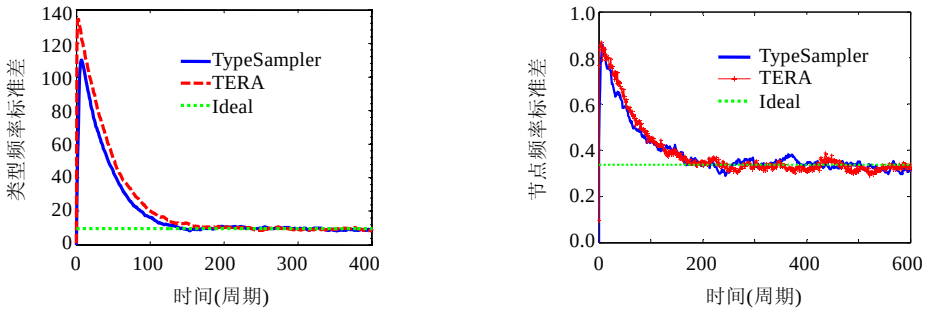
综上可知,TypeSampler 的比例估计算法能够实现较高精度的类型比例估计,而且能够适应节点波动及大量节点瞬时失效的网络动态变化.

5.2 类型采样

TypeSampler 的采样更新算法不仅要实现对类型的均匀随机采样,而且还要实现对同类型节点的均匀随机采样.为验证这两种采样的均匀随机性,我们考察不同类型在类型采样表采样类型字段出现的频率(以下简称类型频率),以及属于同一类型的不同节点在该类型对应的类型采样表条目中出现的频率(以下简称节点频率).在实验中,N=1000,r=0.1,c=20,T=100,s=10,其中,s 为每个节点的类型采样表的大小.

图 6(a)显示了类型频率标准差随时间变化的情况,其中,理想值对应着对类型均匀随机采样情况下的标准差.由图 6 可知,TypeSampler 及 TERA 的标准差都在大约 200 个周期后收敛到理想值附近.而第 4.2 节的图 3 显示,从 TST 填满开始,理论上大约最多需要 250 个周期类型采样即能收敛,如果再加上填满 TST 所需的 8 个左右的周期(对应于图 6(a)中 TypeSampler 的类型频率标准差从 0 增长到峰值的时间长度),那么理论上的收敛速度最多约为 258 个周期.显然,这个理论的界相对于实际情况而言比较宽松.图 6(b)显示了属于类型 1 的节点在类型采样表相应条目中出现频率的标准差随时间变化的情况,其中,理想值对应于在类型 1 条目数固定为 100 时对属于类型 1 的节点均匀随机采样情况下的标准差.由图 6 可知,TypeSampler 及 TERA 的标准差都在大约 200 个周期后收敛到理想值附近.图 6 表明,与 TERA 类似,TypeSampler 的类型频率及节点频率的标准差逐步收敛于均匀随机采样的标准差,但我们还需确定是否存在特定类型及节点的频率长期高于或低于平均值的情况.

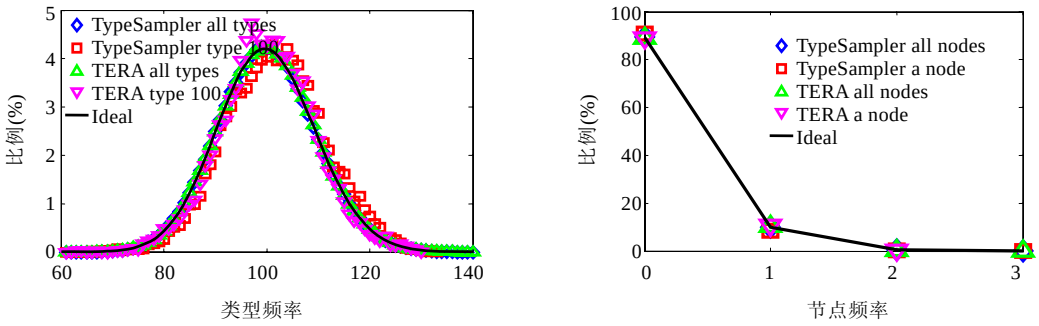
图 7(a)显示了所有类型及类型 100 在一段时间内的频率分布.由图 7 可知,TypeSampler 及 TERA 的所有类型的频率分布与均匀随机采样情况下的理想分布几乎完全重合,而两者的类型 100 的频率分布也都近似于理想分布.考虑到类型 100 的实际比例较低,这表明即使是稀有类型,其在类型采样表中的出现频率也会在平均值周围波动变化.图 7(b)显示了属于类型 1 的所有节点及某特定节点在一段时间内的频率分布情况.由图 7 可知,TypeSampler 及 TERA 的两种分布都与均匀随机采样情况下的理想分布几乎完全重合,这表明,属于某个类型的节点在该类型对应的类型采样表条目中出现的频率也是在平均值周围波动变化的.



(a) 类型频率的标准差 (b) 节点频率的标准差

Fig.6 Standard deviation of type frequency and node frequency

图 6 类型及节点在类型采样表中出现频率的标准差

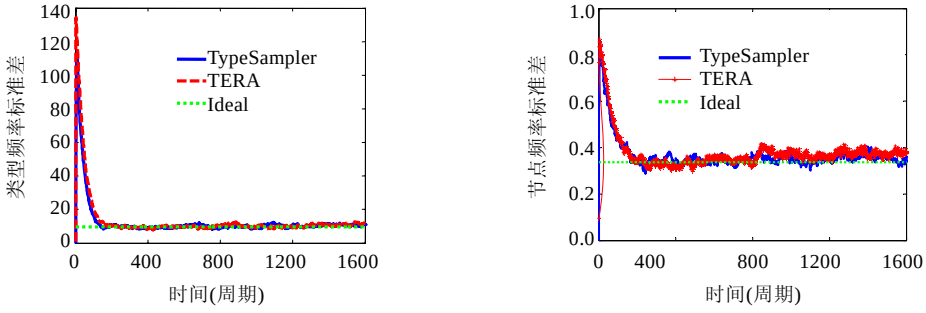


(a) 类型频率的分布 (b) 节点频率的分布

Fig.7 Distribution of type frequency and node frequency

图 7 类型及节点在类型采样表中出现频率的分布

图 8 显示了从第 800 个周期开始,每隔 20 个周期随机选择的 1%的节点被替换的情况下,类型及节点频率标准差的变化.由图 8 可知,TypeSampler 及 TERA 的类型和节点频率标准差在节点波动条件下均略有上升,这主要是因为节点波动会影响比例估计及规模估计的精确度,从而影响类型采样的效果.此外,图 8 还表明,尽管存在着节点波动的不利影响,TypeSampler 仍能保证与 TERA 类似的类型采样性能.

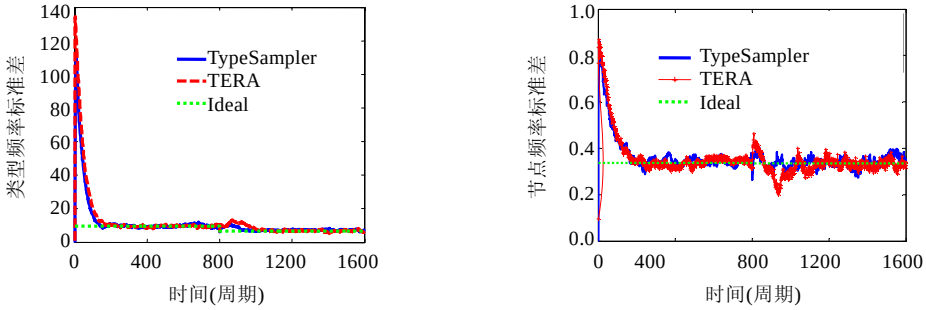


(a) 类型频率的标准差 (b) 节点频率的标准差

Fig.8 Standard deviation of type and node frequency with node substitution

图 8 节点替换情况下类型及节点频率的标准差

图 9 显示了在第 800 个周期随机选择的 50%的节点瞬时失效情况下,类型及节点频率标准差的变化.由图 9 可知,TypeSampler 及 TERA 的类型和节点频率标准差在节点失效时刻均略有起伏,但又都很快收敛于理想值.实际上,由于 TypeSampler 及 TERA 分别基于比例估计及规模估计实现类型采样,因此只要 TypeSampler 的比例估计算法以及 TERA 所依赖的规模估计算法能够适应大量节点瞬时失效的情况,且类型采样表能够得到周期性更新,那么其类型采样的性能也自然能够从大量节点瞬时失效所导致的波动中得以恢复.



(a) 类型频率的标准差 (b) 节点频率的标准差

Fig.9 Standard deviation of type and node frequency with instantaneous node failure

图 9 节点瞬时失效情况下类型及节点频率的标准差

综上可知,TypeSampler 能够保证类型采样以及对同一类型节点采样的近似均匀随机性,而且还能够适应动态的网络环境.值得注意的是,在第 4.2 节的理论分析中,采样更新算法对均匀随机性的保证需以精确的类型比例估计值为前提,但根据本节的实验参数设置,TypsSampler 的比例估计算法的平均相对差错率实际上在 8%与 0.8%之间波动,这表明,即使类型比例估计值存在一定程度的误差,TypeSampler 的采样更新算法仍然能够保证相当的类型采样性能.

5.3 路由效率

我们用路由的平均跳数来衡量路由效率,并通过考察以不同类型节点为目标的路由的平均跳数来检验类型采样对数据路由的支持.TypeSampler 与 TERA 均利用节点采样服务实现基于随机行走的路由,而在路由的每一跳,它们从类型采样表的采样类型及对应节点中检索目标节点以直接转发路由消息.除此之外,TypeSampler 还可利用 *semantic_view* 及 TST 中含有的节点所属类型信息来检索目标节点.而 TERA 的节点采样服务视图以及 APT 中没有这些信息,因而也就无法利用它们引导路由.除了 TypeSampler 及 TERA,我们还考察了不利用任何引导信息而完全基于节点采样服务的随机行走(simple random walk,简称 SRW)的路由效率.在实验中,N=1000,r=0.1,c=20,T=100,s=10.

表 5 显示了在静态网络环境中,TypeSampler,TERA,SRW 以不同类型为目标时的平均路由跳数,其中的 Theoretical 行对应的是根据第 4.3 节定理 5 计算出的 TypeSampler 的路由跳数期望的上界.由表 5 可知:TypeSampler 的平均路由跳数始终小于理论上界,这与定理 5 相一致;TERA 及 SRW 的平均路由跳数则始终大于 TypeSampler 的值.SRW 的平均路由跳数最大,乃是因为其没有利用任何引导信息;而 TERA 的平均路由跳数要高于 TypeSampler,则是因为其利用的路由引导信息少于 TypeSampler.由于节点所属类型是根据 Zpif 分布随机选择的,因此类型序号越大意味着其对应的节点比例越低.所以,随着目标类型序号的增大,SRW 的平均路由跳数增长明显,而由于利用了均匀随机的类型采样,TypeSampler 及 TERA 的平均路由跳数都增长缓慢.

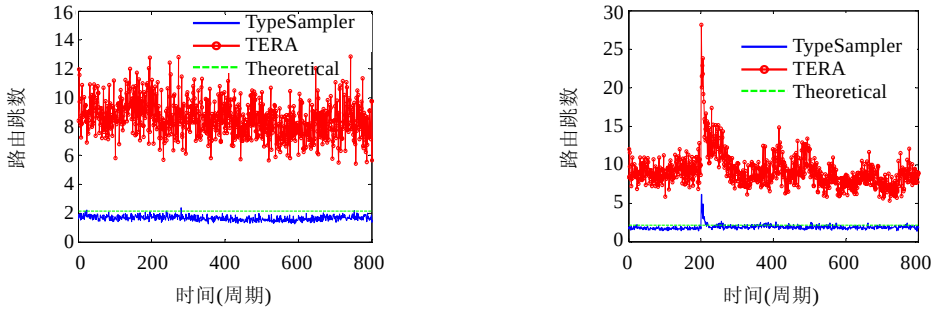
图 10 显示了动态网络环境对路由性能的影响,其中,路由的目标为类型 100;每个周期产生 50 个路由消息;每个数据点对应一个周期内的平均路由跳数,当节点选择的路由下一跳失效时,它将重新选择转发节点并将消息的路由跳数加 1.由于 SRW 的路由跳步数相对较大,为清晰起见,图 10 中没有将其标出.图 10(a)显示了从第 200 个周期开始,每隔 20 个周期随机选择的 1%的节点被替换情况下路由跳数的变化.由图 10 可知,节点替换对

TypeSampler 及 TERA 的路由性能均没有显著的影响.图 10(b)显示了在第 200 个周期随机选择的 50%的节点瞬时失效情况下路由跳数的变化.由图 10 可知,在经历短暂的波动后,TypeSampler 及 TERA 的路由性能都得到了迅速恢复.TypeSampler 及 TERA 的路由性能对动态网络环境的适应,实际上源自节点采样服务的自修复能力以及两者对类型采样表的周期性更新.

Table 5 Average number of routing hops in the static network environment

表 5 静态网络环境中的平均路由跳数

	Type 20	Type 40	Type 60	Type 80	Type 100
Theo tical	1.1	1.3	1.5	1.7	2.1
TypeSampler	1.0	1.1	1.2	1.4	1.8
TE A	4.5	6.2	6.8	7.2	8.7
SI N	7.5	13.8	20.1	25.6	38.3



(a) 节点替换 (b) 节点失效

Fig.10 Average number of routing hops in the dynamic network environment

图 10 动态网络环境中的平均路由跳数

综上可知,相对于 TERA 及 SRW,TypeSampler 能够支持更为高效的路由,而且能够适应节点波动及大量节点瞬时失效的网络动态变化.

5.4 存储与通信开销

我们用单个节点平均存储的数据量来衡量方法的存储开销,用单个节点在单个周期内平均传输的数据量来衡量方法的通信开销.当 N=1000,r=0.1,c=20,T=100,s=10 时,TypeSampler 中每个节点的 *initial_proportions* 及 *proportions* 的条目数平均为 19.1,而每个周期平均传输的 *proportions* 条目数为 20.6.如果以 4 个字节存储 IP 地址,2 个字节存储端口号,4 个字节存储条目年龄,长度为 100 的位向量存储节点所属类型信息,4 个字节存储类型序号,4 个字节存储比例估计值,4 个字节存储比例估计权值,则有 E_{sv}=22.5B,E_{TST}=22.5B,E_{ip}=12B,E_p=8B,E_{TSR}=26.5B,E_v=10B,E_{APT}=10B,E_{sub}=10B.于是,根据第 4.4 节的理论分析及前述每个节点的 *initial_proportions*, *proportions* 的条目数以及每个周期传输的 *proportions* 条目数等实验数据,有以下结果.

表 6 列出了在前述参数设置条件下的 TypeSampler 及 TERA 的存储与通信开销.由表 6 可知,TERA 的存储及通信开销均数倍于 TypeSampler.该结果对应于每个节点平均属于 10 个类型的情况,因此,结合第 4.4 节的分析可知,如果单个节点平均所属类型数进一步增加,那么 TypeSampler 在存储及通信开销上的优势将更加明显.

Table 6 Storage and communication cost

表 6 存储与通信开销

	存储开销(B)	通信开销(B)
TypeSampler	1 057	1 170.8
ERA	2 300	4 800

综上可知,针对单个节点所属类型数增长的情况,TypeSampler 具有比 TERA 更好的可扩展性.

6 结束语

本文提出了一种基于 gossip 的类型采样方法——TypeSampler.在 TypeSampler 中,节点通过 gossip 获取节点样本,并通过基于反熵的聚集过程共享各自单独采样的结果,以计算各类型比例估计值;利用类型比例估计值,节点周期性地更新自己的类型采样表,以使得不同类型的节点能够在其中等概率出现,以及属于同一类型的不同节点能够在该类型对应的条目中等概率出现.基于对类型的均匀随机采样,TypeSampler 能够保证以各种类型节点为目标的路由的平均效率.

理论分析与实验结果表明,即使在动态的网络环境中,TypeSampler 也能够实现精确的类型比例估计以及近似均匀随机的类型采样;而且相对于已有的方法,TypeSampler 能够支持更高效的路由,并具有更好的可扩展性.

References:

[1] Gnutella. http://rfc-gnutella.sourceforge.net/index.html

[2] Clarke I, Sandberg O, Wiley B, Hong TW. Freenet: A distributed anonymous information storage and retrieval system. In: Proc. of the Int'l Workshop on Design Privacy Enhancing Technologies. New York: Springer-Verlag, 2001. 46–66. [doi: 10.1007/3-540-44702-4_4]

[3] Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Balakrishnan H. Chord: A scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Trans. on Networking, 2003,11(1):17−32. [doi: 10.1109/TNET.2002.808407]

[4] Ratnasamy S, Francis P, Handley M, Karp R, Schenker S. A scalable content-addressable network. In: Cruz R, Varghese G, eds. Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2001). New York: ACM Press, 2001. 161−172. [doi: 10.1145/964723.383072]

[5] KaZaA. http://www.kazaa.com

[6] eMule. http://www.emule.org

[7] Yang B, Garcia-Molina H. Improving search in peer-to-peer networks. In: Rodrigues LET, Chen WE, eds. Proc. of the 22nd Int'l Conf. on Distributed Computing Systems (ICDCS 2002). Washington: IEEE Computer Society, 2002. 5−14. [doi: 10.1109/ICDCS.2002.1022237]

[8] Crespo A, Garcia-Molina H. Routing indices for peer-to-peer systems. In: Rodrigues LET, Chen WE, eds. Proc. of the 22nd Int'l Conf. on Distributed Computing Systems (ICDCS 2002). Washington: IEEE Computer Society, 2002. 23–32. [doi: 10.1109/ICDCS.2002.1022239]

[9] Costa P, Picco GP. Semi-Probabilistic content-based publish-subscribe. In: Lai TH, ed. Proc. of the 25th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS 2005). Washington: IEEE Computer Society, 2005. 575−585. [doi: 10.1109/ICDCS.2005.73]

[10] Wong B, Guha S. Quasar: A probabilistic publish-subscribe system for social networks. In: Iamnitchi A, Saroiu S, eds. Proc. of the 7th Int'l Conf. on Peer-to- Peer Systems (IPTPS 2008). Berkeley: USENIX Association, 2008. 2–2. http://dl.acm.org/citation.cfm?id=1855641.1855643&coll=DL&dl=GUIDE&CFID=90551956&CFTOKEN=83243560

[11] Chand R, Felber P. Semantic peer-to-peer overlays for publish/subscribe networks. In: Cunha JC, Medeiros PD, eds. Proc. of the Euro-Par 2005 Parallel Processing (Euro-Par 2005). Berlin: Springer-Verlag, 2005. 1194−1204. [doi: 10.1007/11549468_130]

[12] Voulgaris S, van Steen M. Epidemic-Style management of semantic overlays for content-based searching. In: Cunha JC, Medeiros PD, eds. Proc. of the Euro-Par 2005 Parallel Processing (Euro-Par 2005). Berlin: Springer-Verlag, 2005. 1143−1152. [doi: 10.1007/11549468_125]

[13] Zheng Z, Wang Y. SemanticCast: Content-Based data distribution over self-organizing semantic overlay networks. In: Liao X, Jin H, Zheng R, Zou D, eds. Proc. of the 11th Int'l Conf. on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2010). Washington: IEEE Computer Society, 2010. 42−49. [doi: 10.1109/PDCAT.2010.68]

[14] Cohen E, Shenker S. Replication strategies in unstructured peer-to-peer networks. In: Mathis M, Steenkiste P, eds. Proc. of the 2002 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2002). New York: ACM Press, 2002. 177−190. [doi: 10.1145/964725.633043]

[15] Chockler G, Melamed R, Tock Y, Vitenberg R. SpiderCast: A scalable interest-aware overlay for topic-based pub/sub communication. In: Jacobsen H, ed. Proc. of the 2007 Inaugural Int'l Conf. on Distributed Event-Based Systems (DEBS 2007). New York: ACM Press, 2007. 14−25. [doi: 10.1145/1266894.1266899]

[16] Baldoni R, Beraldi R, Quema V, Querzoni L, Tucci-Piergiovanni S. TERA: Topic-Based event routing for peer-to-peer architectures. In: Jacobsen H, ed. Proc. of the 2007 Inaugural Int'l Conf. on Distributed Event-Based Systems (DEBS 2007). New York: ACM Press, 2007. 2−13. [doi: 10.1145/1266894.1266898]

[17] Kermarrec A, van Steen M. Gossiping in distributed systems. ACM SIGOPS Operating Systems Review, 2007,41(5):2−7. [doi: 10.1145/1317379.1317381]

[18] Jelasity M, Kowalczyk W, van Steen M. Newscast computing. Technical Report, IR-CS-006, Amsterdam: Department of Computer Science, Vrije Universiteit, 2003.

[19] Voulgaris S, Gavidia D, van Steen M. CYCLON: Inexpensive membership management for unstructured P2P overlays. Journal of Network and Systems Management, 2005,13(2):197−217. [doi: 10.1007/s10922-005-4441-x]

[20] Jelasity M, Voulgaris S, Guerraoui R, Kermarrec A, van Steen M. Gossip-Based peer sampling. ACM Trans. on Computer Systems (TOCS), 2007,25(3): Article No. 8. [doi: 10.1145/1275517.1275520]

[21] Kempe D, Dobra A, Gehrke J. Gossip-Based computation of aggregate information. In: Broder AZ, ed. Proc. of the 44th Annual IEEE Symp. on Foundations of Computer Science (FOCS 2003). Washington: IEEE Computer Society, 2003. 482−491. [doi: 10.1109/SFCS.2003.1238221]

[22] Jelasity M, Montresor A, Babaoglu O. Gossip-Based aggregation in large dynamic networks. ACM Trans. on Computer Systems, 2005,23(3):219−252. [doi: 10.1145/1082469.1082470]

[23] Boyd S, Ghosh A, Prabhakar B, Shah D. Randomized gossip algorithms. IEEE Trans. on Information Theory, 2006,52(6): 2508−2530. [doi: 10.1109/TIT.2006.874516]

[24] Sacha J, Napper J, Stratan C, Pierre G. Adam2: Reliable distribution estimation in decentralised environments. In: Cellary W, Zhang X, eds. Proc. of the 2010 Int'l Conf. on Distributed Computing Systems (ICDCS 2010). Washington: IEEE Computer Society, 2010. 697−707. [doi: 10.1109/ICDCS.2010.16]

[25] Haridasan M, van Renesse R. Gossip-Based distribution estimation in peer-to-peer networks. In: Iamnitchi A, Saroiu S, eds. Proc. of the 7th Int'l Workshop on Peer-to-Peer Systems (IPTPS 2008). Berkeley: USENIX Association, 2008. 13−13. http://dl.acm.org/citation.cfm?id=1855641.1855654&coll=DL&dl=GUIDE&CFID=90607975&CFTOKEN=94856818

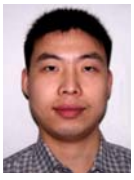
[26] Demers A, Greene D, Hauser C, Irish W, Larson J, Shenker S, Sturgis H, Swinehart D, Terry D. Epidemic algorithms for replicated database maintenance. In: Schneider FB, ed. Proc. of the 6th Annual ACM Symp. on Principles of Distributed Computing. New York: ACM Press, 1987. 1−12. [doi: 10.1145/41840.41841]

[27] PeerSim. http://peersim.sourceforge.net/

郑重(1982－),男,江苏扬州人,博士生,CCF 学生会员,主要研究领域为网络计算,数据分发技术.

马行空(1987－),男,博士生,CCF 学生会员,主要研究领域为网络计算,数据分发技术.

王意洁(1971－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,海量数据管理,虚拟化技术.



杨永滔(1981－),男,博士生,CCF 学生会员,主要研究领域为不确定数据管理,数据流查询.