通信网络中一种基于流的异步平均一致性协议

王得洋[1] 王从银[1] 庄 雷[1] 陈鸿昶[2]
(郑州大学信息工程学院 郑州 450001)[1]
(解放军信息工程大学国家数字交换系统工程技术研究中心 郑州 450002)[2]

摘 要 大规模的异步通信网络中,实时获取系统级网络平均值对于指导系统进行控制决策,比如资源选择、负载均衡等,具有重要的意义。基于此,重点研究了异步网络环境中的平均一致性问题,提出了一种基于流的异步平均一致性协议FBAA。FBAA协议适用于动态的异步通信网络系统,而且运行过程不需要全局协调。实验表明该协议能够以较快的速度收敛到平均值,且收敛时间与网络规模无关。进一步通过对实验数据的统计分析,得出收敛时间与相关参数的关系以及算法达到最优收敛时间的参数设置。

关键词 异步计算,平均一致性,流,收敛时间
中图法分类号 TP393 **文献标识码** A

Flow-based Asynchronous Averaging Consensus Protocol on Communication Network

WANG De-yang[1] WANG Cong-yin[1] ZHUANG Lei[1] CHEN Hong-chang[2]
(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)[1]
(Information Engineering University, National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China)[2]

Abstract In the large-scale asynchronous communication network, the real-timely computing of system-level average parameters is of great significance for guiding the systems to make control decisions, such as resource selection and load balancing. This paper concentrated on the problem of averaging consensus in the asynchronous network environment and proposed a flow-based asynchronous averaging consensus protocol FBAA. The FBAA protocol is applicable for the dynamic asynchronous communication network system, and doesn't require global coordination in its whole running process. The simulation results presented in this paper show that our protocol can converge to the average value more quickly and the convergence time is independent from the scale of network. Furthermore we derived the relationship between the convergence time and some other parameters through analyzing the experimental data, and the settings of parameters when the system achieves to optimal convergence time.

Keywords Asynchronous computing, Averaging consensus, Flow, Convergence time

1 引言

网络平均值是指某一系统参量相对于网络规模的平均数,在分布式网络系统中可以用来指示系统中某一资源状态属性的平均值,如果网络节点能实时获取这种动态的平均值,便能够很方便地解决很大一部分网络资源相关的问题,比如:IP网络中由于网络资源利用率在空间分布上的不均衡造成全网范围内的带宽使用率低[1],解决此类问题的关键是改变数据传输路径,最终仍然由节点做出相应的转发决策,因此,对于任一节点,若能有效地获取相关网络资源状态属性值,便能有效解决资源利用率低的问题;P2P的文件共享系统当中,要使文件资源均衡分布在系统节点,同时平衡各种资源的请求与供应量,简单有效的办法是直接获取相关资源的动态属性,然后做出相应的资源均衡决策,从而调节系统资源的供需平衡[2]。

理论上,计算分布式网络系统中某资源的全局平均值,可以采用洪泛消息或者构建结构化的覆盖网传播消息,然后收集消息最终计算出平均值,然而此类方法均需要预先请求,再收集结果,这样不但会产生大量的网络消息而且实时性较差,同时还需要一个全局的协调才能完成任务。使用异步迭代的方法计算网络的全局平均值无需全局协调,每个节点只同其邻居节点交换数据,即可实时地获取网络平均值,这种方法在很多领域已经有较深入的研究[3-7],尤其是在一些同步网络环境中,比如:在传感网络中用于计算某物理测量值的全局平均值,在多智能体编队控制中[5,6],借助于这种平均值使智能体涌现出一致的行为;也有很多学者将这种获取网络系统全局平均值方法重新设计,用于计算基于数据包交换的通信网络系统中[7,8]相关网络资源的平均值,以此为指导来均衡系统

到稿日期:2012-09-08 返修日期:2012-12-24 本文受国家重点基础研究发展计划(973计划)项目(2012CB315901),河南省教育厅自然科学基金项目(2010A520004),河南省科技厅攻关项目(122102210042)资助。

王得洋(1987-),男,硕士生,主要研究方向为计算机网络,E-mail:happy-wdy@qq.com;王从银(1977-),男,博士生,主要研究方向为计算机网络;庄 雷(1963-),女,博士,教授,博士生导师,主要研究方向为网络可重构;陈鸿昶(1964-),男,博士,教授,博士生导师,主要研究方向为计算机网络。

负载。本文所提出的基于流的异步平均一致性协 FBAA (Flow-based Asynchronous Averaging Consensus Protocol)也即是着眼于解决异步通信网络环境下的平均一致性问题，并进一步通过仿真实验对算法进行收敛分析。

对于异步平均一致性协议的设计，先前的研究一般都是采用推拉结合的方法，任意节点每一次仅能选择一个邻居节点进行数据迭代，而且对于节点的选择有的是基于概率框架[9]，有的是随机选择[6,7,10]，基于概率的节点选择方案需要从整体上考量每条链路发生数据迭代的概率，这依赖于网络拓扑，因此在异步动态的通信网络当中很难实施，而基于随机的选择方案具有盲目性，节点之间的数据交换没有方向，增加了收敛时间。

我们通过模拟液体流动行为设计一种新的基于推模式的异步平均一致性协议 FBAA，将数据流动抽像为流，任意节点可以同时对多个邻居节点发生数据流动，且节点发生数据交换的方式依据其状态变量从大到小的方向进行迭代，通过仿真实验验证了我们的算法能够以相对较快的速度收敛到正确的平均值，同时我们对实验数据进行对比分析得出算法收敛时间与网络规模、平均节点度、流动速度(算法发送流命令的频率)、算法平均步长等参数的关系。

本文第 2 节定义了通信网络中异步平均一致问题，以及提出了 6 条需要解决的关键问题；第 3 节详细介绍了基于流的异步平均一致性协议，同时对协议中部分结论做出了理论证明；第 4 节是仿真实验和算法收敛分析；最后给出结论和未来的工作。

2 定义和问题陈述

通信网络中异步平均一致问题：将网络抽象成图 $G(V,E)$，其中 $V=\{1,2,3\cdots,n\}$ 表示网络节点集合，$E=\{(i,j)\mid i,j\in V\}$ 表示节点之间的边集合，第 i 个节点维护着一个状态变量 z_i(z_i 一般用来指示一个资源状态属性，用某一实值表示)和一个随迭代次数增加而变化的本地变量 $x_i(t)$，并将 $x_i(0)$ 初始化为 z_i，每个节点通过周期性地和邻居节点进行本地变量的迭代更新，使得

$$\lim_{t\to\infty}x_i(t)=\frac{1}{n}\sum_{i=1}^{n}z_i \quad \forall i\in V \tag{1}$$

在基于离散时间的网络系统模型当中，假设在某一时刻，只有一对互为邻居的节点 i 和 j 进行数据交换，交换方式见式(2)：

$$\begin{cases}x_i(t+1)=x_i(t)+\gamma(x_j(t)-x_i(t))\\ x_j(t+1)=x_j(t)+\gamma(x_i(t)-x_j(t))\end{cases} \tag{2}$$

在本篇论文中，把式(2)中的 γ 叫做平均步长，且 $0<\gamma<1$。

文献[7]理论证明了，在网络 G 连通的情况下式(1)成立；文献[11]研究表明网络代数连通度是影响算法收敛快慢的重要因素。在此理论基础上，本文主要采用实验的方法定性和定量地对收敛时间进行统计分析，因此协议的设计不仅考虑到提高收敛时间，同时兼顾便于收敛分析。

我们将节点间的数据交换抽象成流函数，$f:V\times V\to R$，满足：

$$\begin{cases}f(i,j)=-f(j,i)\\ f(i,j)=0, & (i,j)\notin E\\ f(i,j)=\gamma(x_i-x_j)>0, & (i,j)\in E\end{cases} \tag{3}$$

下面使用更加简单的符号 f_{ij} 来表示 $f(i,j)$。因此节点间数据交换式(3)可改写为如下形式：

$$\begin{cases}x_i(t+1)=x_i(t)+f_{ij}(t)\\ x_j(t+1)=x_j(t)+f_{ji}(t)\end{cases} \tag{4}$$

通过分析式(3)易知基于节点对的数据交换实际上是饱和的，即节点 i 和节点 j 的本地变量在每一次数据交换后都向对方靠近 $\gamma|x_i-x_j|$ 长度，从整个网络来看，在任意一次迭代交换后 $\sum_{i=1}^{n}x_i(t)$ 始终等于 $\sum_{i=1}^{n}z_i$，因此我们定义系统聚集总量 M 来表示它：

$$M=\sum_{i=1}^{n}z_i \tag{5}$$

基于以上分析和定义，将待解决问题总结如下：

(1)对于任意节点 i，何时向其邻居节点(用 A_i 表示)发出数据交换。先前的工作基本上都是基于请求等待模式，这种模式的缺点是耗费时间，实现复杂。

(2)对于任意节点 i，如何选择邻居节点进行数据交换。有很多文章提出采用基于概率框架的算法，但其在动态异步的网络环境下表现得并不好，一般都采用随机选择的方式。

(3)假如对某一时刻节点 i 选择了节点 j 进行数据交换，那么具体交换多少，也即 γ 的取值为多少比较合适。先前文章对于 γ 未做讨论。

(4)节点动态退出，会使系统聚集总量 M 发生变化，造成 x_i 收敛到错误的值。一般的解决方法是增加变量 $\delta_{ij}=\sum_{t=0}^{\infty}f_{ij}(t)$，其表示由节点 i 向节点 j 发生交换的总量，当节点退出时，运行数据恢复机制。

(5)链路丢包，由于接收方未接收到数据包，而发送方运行了式(4)，导致系统聚集总量 M 发生变化，造成 x_i 收敛到错误的值，所以，一般需要数据包确认机制。

(6)实际应用中，状态变量 z_i 是随时间变化的，那么 $\lim_{t\to\infty}x_i(t)=\frac{1}{n}\sum_{i=1}^{n}z_i(t)$，因此对于能否使 $x_i(t)$ 及时收敛到正确的值是一个挑战。当然这与 z_i 变化的程度有密切关系，下文中将有较详细的说明。

3 基于流的异步平均一致性协议(FBAA)

3.1 协议概述

如果节点 i 向节点 j 发出数据流动时，必须满足的首要条件是：确定数据更新主从关系，也即确定哪一方是流的发动者，哪一方是流的接收者。在这里采用流总是从高处往底处流动的思想确定数据更新的主从关系，即若节点 i 向节点 j 发出流，则必须满足 $x_i>x_j$。由此分析，我们不妨将协议设计为两层：第一层用来通知邻居节点当前本地变量的值，用 $State(j,x_i)$ 命令表示，因此对于任意节点 i，都需要设定临时变量 $x_j'(j\in A_i)$ 保存其邻居节点的本地变量的值；第二层用来向邻居节点发送数据流，用 $Flow(j,f_{ij})$ 命令表示。

为加快收敛时间和消除阻塞，我们采用节点主动推数据的模式，即每一个节点都按某一频率不断向邻居节点发送 *State* 命令和 *Flow* 命令，因为 *Flow* 命令的发送依赖于对方节点当前本地变量值的大小，所以设定 *State* 的发送频率要高于 *Flow* 命令的发送频率。分别用 p_s 和 p_f 表示 *State* 命令和 *Flow* 命令的发送频率。*State* 线程和 *Flow* 线程的发送过程 *MP1* 和 *MP2* 具体描述如下。

*MP*1：每个节点以频率 p_s 向其所有邻居节点发送 *State* 命令，告之对方其当前本地变量值，接收方接到 *State* 命令后立即更新其相对应的本地变量。

*MP*2：每个节点以频率 p_f 向邻居节点发送 *Flow* 命令，具体的流分配算法见 A1，接收方接收到 *Flow* 命令后把命令里携带的数值累加在其当前本地变量上；具体发送过程如图 1 所示，其中 k_1, k_2, k_3 是节点 i 的邻居节点，且 $x_{k_1}' < x_{k_2}' < x_{k_3}'$。

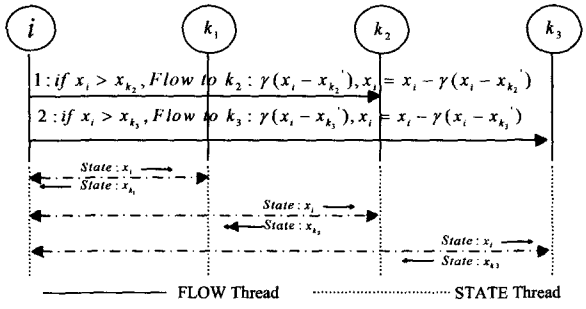

图 1　节点主动发送示意图

A1：节点 i 数据流分配算法

1. 初始化邻居节点集合 A_i；
2. 根据本地变量 $x_s'(s \in A_i)$ 从小到大对 A_i 进行排序；
3. For all $s \in A_i$ and $x_s' > x_i$ do
4. 　　Flow(s, f_{is})；
5. 　　$x_i -= \gamma |x_s' - x_i|$；
6. End For

3.2　节点动态加入和退出

由于网络节点的动态性可能会使算法在运行过程中发生错误收敛，因此协议必须有一定的节点加入和退出机制。

节点加入系统并连接一定个数的邻居节点后，即可运行协议，对系统协议收敛到正确的数值并不会造成影响，因此定义命令 *Join*，当节点初次加入系统时，向邻居节点发送此命令，并做一些初始化工作。

当节点退出时，如果没有补偿操作，就会使系统丢失一部分数据流量，假设在 t 时刻节点 i 离开，那么此时系统本地变量总和为 $(\sum_{k=1}^{n} x_k(t)) - x_i(t)$，即等于 $M - x_i(t)$，理想情况下，当节点 i 退出系统时系统状态变量总和应该为 $M - z_i$，因此如果没弥补措施，系统 M 的损失为：

$$L = x_i(t) - z_i \tag{6}$$

为避免因节点退出而引发数据流量的丢失，我们引入辅助变量 $\delta_{ij}(t) = \sum_{t=0}^{\infty} f_{ij}(t)$，对于任意节点 i，相对于其所有邻居节点都维护这样一个变量 $\delta_{ik}(t)$，记录总的数据流量，$\delta_{ik}(t) < 0$ 表示由节点 k 向节点 i 流入的数据量，$\delta_{ik}(t) > 0$ 表示由节点 i 向节点 k 流出的数据量，$\delta_{ik}(t) = 0$ 表示没有数据量流过或者是在整个过程数据流入与流出总量抵销。易得出结论：

$$\delta_{ik}(t) = -\delta_{ki}(t) \tag{7}$$

通过以上分析，我们不妨增加一种命令 *Quit*，任意节点 i 退出系统时，将向其所有邻居节点发送 *Quit* 命令，其邻居节点 k 收到 *Quit* 命令后则会按如下方式对本地变量进行弥补：

$$\begin{cases} x_k(t) = x_k(t) + \delta_{ki}(t) \\ \delta_{ki}(t) = 0 \end{cases} \tag{8}$$

命题 1　通过以上退出补偿方式，系统 M 损失将得以补偿。

证明：不失一般性，假设某一时刻节点 i 退出系统，并且向其所有邻居节点发送了退出命令，A_i 为邻居节点集合，那么当其所有邻居节点收到 *Quit* 命令后，运行式(7)，对系统聚集总量 M 总的补偿为：

$$C = \sum_{k \in A_i} \delta_{ki}(t) \tag{9}$$

由于 δ_{ik} 记录着所有流入和流出节点 i 的总量，易知：

$$z_i - \sum_{k \in A_i} \delta_{ik}(t) = x_i(t) \tag{10}$$

由式(6)－式(10)得，$L = C$，命题 1 得证。

关于节点退出的两点补充说明：(1)多个相邻的节点同时退出时依然不会影响系统 M，因为我们可以把多个节点看成是同一个节点，将多个节点的所有邻居节点看成是这个节点的邻居节点，结合上面的证明易知系统会得到正确的补偿；(2)节点由于断电或错误而发生硬关闭时，不会向其邻居节点发出 *Quit* 命令，这种情况的解决方法是让节点主动检测邻居节点是否在线，如果检测到邻居节点掉线就运行补偿程序。

3.3　链路丢包

在基于分组转发的通信网络中，发生链路丢包的现象较为普遍，特别是在网络拥塞的情况下，因此节点发出 *Flow* 命令的前提条件是其上一次 *Flow* 命令已经被确认，否则很可能会因为链路丢包使得系统聚集总量 M 丢失一定的数据量。我们采用稍带确认机制来解决这一问题，具体办法是通过在 *State* 命令中加上一个 *Flow_Ack* 字段，其表示对上一次收到对方的 *Flow* 命令预以确认；由于 *State* 命令会连续不断向接收方发送确认，因此在节点发送 *Flow* 命令后会不断接收到对方的确认，如果 *State* 确认命令在传输过程中偶尔再次丢包也不会给系统造成影响。

定义命令 *UnFlow*(j, f_{ij}) 表示对 *Flow*(j, f_{ij}) 命令的逆操作，π 为节点连续从某一邻居节点收到对非上次 *Flow* 命令确认的次数，如果 $\pi \geqslant p_f$，则认为上次发送的 *Flow* 数据包丢失。

通过以上的分析我们将数据流分配算法重新设计，如 A2 所示。

A2：节点 i 数据流分配算法

1. 初始化邻居节点集合 A_i；
2. 根据本地变量 $x_s'(s \in A_i)$ 从小到大对 A_i 进行排序；
3. For all $s \in A_i$ and $x_s' > x_i$ do
4. 　If(已收到节点 s 对上一次 Flow 命令的确认)
5. 　{
6. 　　Flow(s, f_{is})；
7. 　　$x_i -= \gamma |x_s' - x_i|$；
8. 　}
9. Else If($\pi \geqslant p_f$)
10. {
11. 　UnFlow(s, f_{is})；
12. }
13. End for

3.4　状态变量动态变化

以上我们的协议只适用于状态变量 z_i 不变的静态平均一致性问题，但在实际应用中，状态变量 z_i 可能是随时间变化的函数 $z_i(t)$，在这种情况下我们需要求的平均值 $\bar{x}$ 也是随时间变化的函数，即 $\bar{x}(t) = \frac{1}{n}\sum_{i=1}^{n} z_i(t)$，因此实时地求精确的 $\bar{x}(t)$ 将变得不可能，这种问题最终转化为对 $\bar{x}(t)$ 动态估计的问题，本文将不对此作过多的讨论。

定义 $\Delta z_i(t)$ 为时刻 t 状态变量 z_i 的增加量，为了保证 z_i 变化的情况下系统收敛到正确的值，我们不妨将 $\Delta z_i(t)$ 看成流量，然后向系统 M 注入即可，此时动态平均一致性问题又转化为静态平均一致性问题，当 z_i 发生变化时，节点应做如下的更新：

$$\begin{cases} x_i(t)=x_i(t)+\Delta z_i(t) \\ z_i=z_i+\Delta z_i(t) \end{cases} \tag{11}$$

相类数学符号说明如表 1 所列。

表 1 相关数学符号说明

符 号	描 述
G(V,E)	网络拓扑图(V 网络顶点集,E 网络边集)
$x_i(t)$	第 i 个节点在 t 时刻的本地资源状态值
$\bar{x}(t)$	t 时刻系统中某一资源的平均值
$z_i(t)$	第 i 个节点 t 时刻某一资源的当前总量值
f_{ij}	从节点 i 向节点 j 流动的数据量
γ	节点发出数据流动系数，取值 0～1 之间
θ	网络平均节点度
M	系统当前所有资源总量之和
$\delta_{ij}(t)$	记录截止到 t 时刻节点 i 向节点 j 流动数据量的总和
p_s	State 命令的发送频率
p_f	Flow 命令的发送频率
A_i	节点 i 的邻居节点集
x_i'	存储在节点 i 的邻居节点上，指示节点 i 的当前本地变量值
Flow(j,f_{ij})	节点 i 向其邻居节点 j 发送 f_{ij} 大小的流量
UnFlow(j,f_{ij})	撤销节点 i 向其邻居节点 j 发送的 f_{ij} 大小的流量
π	判断发送的流命令是否丢包的门限值
$\Delta z_i(t)$	时刻 t 节点 i 资源总量的变化量

4 仿真实验和收敛分析

4.1 实验参数设置

我们用 C++实现了一个基于离散事件仿真器，对协议进行仿真，实验环境是 VC6.0 平台，双核 CPU 和 4G 内存的服务器，具体的实验参数设置如表 2 所列。

表 2 模拟实验参数设置

参数	值域
链路延迟 d	服从 40～500 毫秒均匀分布
网络丢包概率 p_l	0.001
网络规模 n	0～2000
平均节点度 θ	2～(n−1)
平均步长 γ	0.0～1.0
Flow 发送频率 $p_f(s^{-1})$	1～+∞
State 发送频率 $p_s(s^{-1})$	$\mu p_f(\mu \geqslant 2)$

定义收敛误差 ε 来反映整体收敛程度：

$$\varepsilon=\left(\sum_{i=1}^{N}|x_i-\bar{x}|\right)/n \tag{12}$$

式中，$\bar{x}$ 表示正确的平均值，为了便于实验对比分析，需要设定算法收敛的判断：当 $\varepsilon \leqslant \bar{x}/1000$ 时，我们即认为算法已收敛。

定义每秒钟系统产生的消息数量 C_n 来衡量系统的运行过程中的网络开销：

$$C_n=\frac{\theta n}{2}(p_f+2p_s) \tag{13}$$

定义收敛时间 T 是关系于流动速度 p_f、平均步长 γ 和平均节点度 θ 的函数，即 $T(p_f,\gamma,\theta)$。

4.2 仿真实验结果与分析

4.2.1 协议正确性验证

我们创建一个有 2000 节点规模的随机网络，将 50%节点状态变量置为 1.0，50%节点状态变量置为 0.0，其它相关参数设置见图 2 标识，图 2 纵轴表示 2000 个节点状态变量 $x_i(t)$ 在异步平均过程中变化曲线的叠加，可以看出在第 5 到第 6 秒时所有的状态变量都收敛到正确的值 0.5。

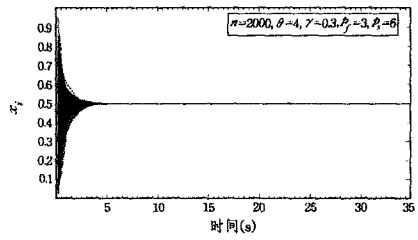

图 2 协议仿真实验结果

4.2.2 与 Mortada Mehyar 提出的算法做对比

我们将 Mortada Mehyar 在文献[7]设计的算法 A2 用 MMA2 标识，同时在本仿真器上实现了 MMA2 算法。通过对比实验结果发现我们的算法具有较快的收敛速度，如图 3 所示，其中参数 p_f 是 FBAA 算法的固有参数。同时经过测量系统每秒钟产生的消息数量发现，FBAA 的要略高一些，但是由于 FBAA 的消息是按某一频率主动发送，因此是稳定可控的。

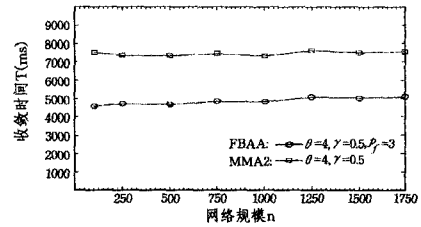

图 3 算法收敛时间对比

4.2.3 收敛时间与网络规模无关

通过大量的仿真测试发现在不同的参数设置下，收敛时间 T 不依赖于网络规模。如图 4 所示，在 3 种不同参数设置下，随着网络规模的增加，收敛时间 T 较为稳定，因此我们的协议适用于大规模的网络系统。

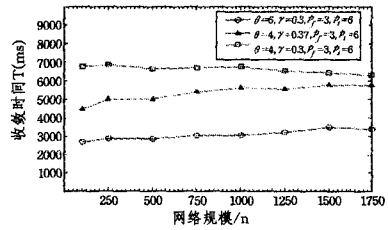

图 4 收敛时间与网络规模的关系

4.2.4 流动速度直接影响收敛时间的快慢

Flow 发送频率 p_f 和 *State* 发送频率 p_s 是间接刻画系统运行速度的重要参数，由于 *State* 命令只是保证状态变量在邻居节点上的同步更新，与算法迭代并无关系，因此我们主要讨论的是 *Flow* 发送频率与收敛时间的关系。理论上 p_f 越大，收敛时间会越小，但实际上 p_f 并不能无限增大：一方面，因为链路传输和节点的处理能力有限，导致 p_f 增大到一定的程度时并不能对降低收敛时间产生太大贡献；另一方面，从系统优化的角度考虑，由式(13)知，随着 p_f 的增大，网络开销 C_n 也会线性增加。因此合理地设定 p_f 对网络优化具有较重要的意义，如图 5 所示，随着 p_f 的增加，收敛时间 T 单调递减，但当 $p_f \to \infty$ 时，T 趋于稳定。

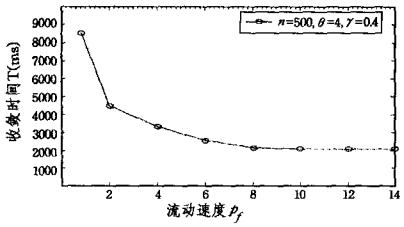

图 5　收敛时间与流动速度的关系

4.2.5　最优收敛时间

平均步长 γ 反映的是每次数据流动大小的比例常数，过大会引起网络中数据流震荡，延长收敛时间，过小使得收敛过程较平滑，但同样也会延长收敛时间；平均节点度反映的是整个网络的连通性，网络连通性越高，数据流就越快流动到达整个网络，那么收敛时间就越少（具体见文献[11]）。然而实际上在我们的算法当中，θ 并不直接影响收敛时间，而是受 γ 的影响，只有 γ 取值合适时，增加网络连通性才有意义，导致这种结果的主要原因是我们的数据流分配算法是针对所有邻居节点进行分配的。我们通过采用枚举参数的方法产生大量关于函数 $T(p_f=3,\gamma,\theta)$ 的数据，如图 6 所示。

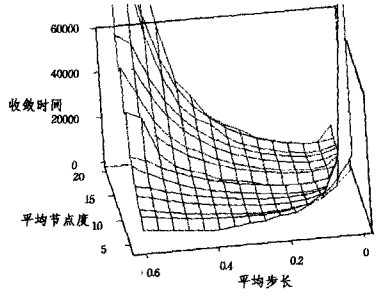

图 6　收敛时间与平均步长和平均节点度的关系

因为我们总是最关心最优收敛时间，所以下面着重分析如何获得最优收敛时间，以及在综合考虑网络开销的情况下又应该怎样取舍，从而获得较为满意的收敛效果。

从图 6 中可以发现：对于任意给定的平均节点度 θ，总是可以找到一个最优的平均步长 γ^*，使得收敛时间 T 最小。通过统计数据获得 γ^* 和 θ 的关系如图 7 所示。我们不妨采用指数曲线模型拟合 γ^*：

$$\gamma^* = ae^{\frac{b}{\theta}} \tag{14}$$

将式(14)进行化归，两边取对数得：

$$\ln\gamma^* = \ln a + \frac{b}{\theta} \tag{15}$$

令 $u=\ln\gamma^*$，$v=\frac{1}{\theta}$，$c=\ln a$ 得：

$$u = c + bv \tag{16}$$

由式(16)，我们将方程化归为线性方程，通过使用最小二乘法拟合直线方程最终得出：

$$\gamma^* = 0.5555e^{8.3917/\theta} \tag{17}$$

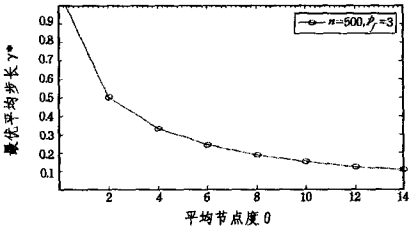

图 7　最优平均步长与平均节点度的关系

考虑 $\gamma=\gamma^*$ 时函数 $T(p_f=3,\gamma=\gamma^*,\theta)$ 的变化特征，通过统计数据获得 $T(p_f=3,\gamma=\gamma^*,\theta)$ 与 θ 的关系，如图 8 所示。分析发现此时 T 是关于 θ 单调递减函数，结合本节上述的论述可以得出最优收敛时间：

$$T^* = T(p_f=+\infty,\gamma=\gamma^*,\theta=n-1) \tag{18}$$

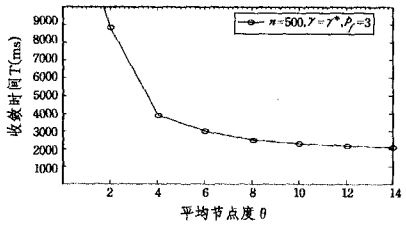

图 8　平均步长取最优时收敛时间与平均节点度的关系

由式(18)得出了达到最优收敛时间时各参数的设置，但是事实上从实际应用角度出发，我们必须兼顾考虑网络代价。由式(13)知网络代价是关于 p_f 和 θ 的线性增函数，因此在可以接受的收敛时间范围内，选择合适的 p_f、γ、θ 对系统优化具有重要的意义。

结束语　本文通过模拟液体流动行为设计了一套基于流的异步平均一致性协议，通过仿真实验验证了我们的算法能够以相对较快的速度收敛到正确的平均值，同时对于动态的异步通信网络环境协议具有较强的适应性，而且通过分析大量实验数据获得了算法收敛时间与相关参数的关系以及算法达到最优收敛时间时各参数的设置；但协议在获得快速收敛和适应高动态网络环境的优势的同时，对于带宽的要求也相对较高，因此本协议较适用于目前流行的 IP 通信网络以及构建在 IP 通信网络之上的分布式网络系统。

下一步工作是进一步挖掘基于流的异步平均算法的应用价值，比如用于解决目前互联网带宽利用率低的问题和基于流动特征的资源定位等。

参 考 文 献

[1] 卜佑军. IP 网多路径数据传输关键技术研究[D]. 郑州：解放军信息工程大学，2012

[2] Tang H-S, Chan S-H G, Li Hao-chao. Optimizing segment caching for peer-to-peer on-demand streaming[A]// Proceedings of the 2009 IEEE International Conference on Multimedia and Expo[C]. New York, NY, USA, 2009: 810-813

[3] Nedic A, Olshevsky A, Ozdaglar A. On Distributed Averaging Algorithms and Quantization Effects[J]. IEEE Transactions on Automatic Control, 2009, 54(11): 2506-2517

[4] Chen Xue-song, Yang Yi-min, Cai Shu-ting, et al. Modeling and Analysis of Multi-agent Coordination Using Nearest Neighbor Rules [C] // International Asia Conference on Informatics in Control, Automation and Robotics. 2009

[5] Olfati-saber R, Fax J A, Murray R M. Consensus and Cooperation in Networked Multi-Agent Systems[J]. Proceedings of the IEEE, 2007, 5(1): 215-233

[6] Bicocchi N, Mamei M, Zambonelli F. Handling dynamics in gossip-based aggregation schemes [C]// IEEE Symposium on Computers and Communications, ISCC 2009. July 2009: 380-385

[7] Mehyar M, Spanos D, Pongsajapan J. Asynchronous Distributed Averaging on Communication Networks[J]. IEEE/ACM Transactions on Networking, 2007, 15(3): 512-520

（下转第 76 页）

最后可以得到发送波束成形矩阵为

$$\hat{T}_k^{(b)}=((\beta_1\hat{H}_k^{(b)H}\hat{H}_k^{(b)}+\beta_2+\lambda_k I_{N_t})^{-1}\hat{H}_k^{(b)H} \quad (15)$$

式中，$\theta_1=(N_t-\frac{N_tD}{N_r}-\frac{N_tN_rD}{N_r(N_t-N_r)})$，$\theta_2=[C_k^{(b,1)}\quad C_k^{(b,2)}\quad\cdots\quad C_k^{(b,B)}]$，其中的列元素 $C_k^{(b1,b2)}$ 可以表示为 $C_k^{(b,b2)}=\sum_{j=1}^{K}\beta_{kj}^{(b,b2)}(A_j^{(b)H}\hat{H}_j^{(b)H}\hat{H}_j^{(b2)}A_j^{(b2)})$，$\lambda_k$ 由二分法搜索求得。

5 仿真结果

本文的仿真环境完全采用符合 LTE-A 的 Release10 的标准，在这里不再赘述。小区内基站和用户均配置 2 根天线，设符号周期 Ts 为 1μs，用户数 $K=2$，每一个小区簇有基站的数目为 $B=2$，调制方式为 QPSK，为了对比性能，本文在仿真中没有考虑信道编码。

图 1 仿真了所提 MMSE 预编码与忽略异步干扰下的 BD 波束成形算法随信噪比变化的最大互信息的对比。在反馈比特数 $Q=8$ 和 16 的情况下，从图可以看出，在完全 CSI 情况下的 MMSE 性能是最好的。但将忽略异步干扰情况下的有限反馈 MMSE 方案与本章所提的考虑异步干扰消除的情况的 MMSE 性能相比较，可以看出，本文所提的方案在一定程度上消除了由于异步干扰和信道时变所带来的干扰，提高了系统性能。同时可以发现，在考虑异步干扰的情况下，反馈比特数越大，系统的最大互信息也越大。这是由于越大的反馈比特数对信道的量化越精确。所以可以看出，在同时存在有限反馈和信道时变的特性下，本文所提出的方案在信道容量方面存在较大的优势，这也就保证了小区边缘用户在多媒体业务方面的容量支撑，通信的有效性大幅提高。

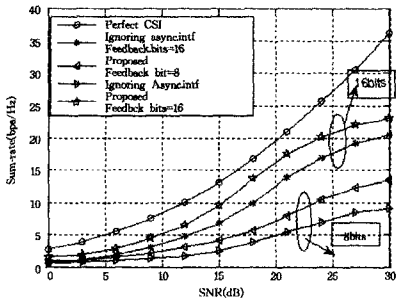

图 1 考虑异步干扰和反馈的最大互信息对比

图 2 显示了所提方案在误比特率 BER 下的性能。同样可以看出，完全 CSI 情况下的 BER 最小，但与忽略异步干扰情况下的 MMSE 相比，本文所提的消除异步干扰 BD 波束成形方案在 BER 方面体现了更好的性能，具有更小的 BER。可以看到在传统的 BD 算法中，其 BER 的性能是劣于 MMSE 的性能的，但是由于在预先消除了反馈误差和信道时变带来的干扰后，本文所提的方案相对于 MMSE 存在可观的优势，从而提升了系统的可靠性，为数据业务提供了很好的保证。

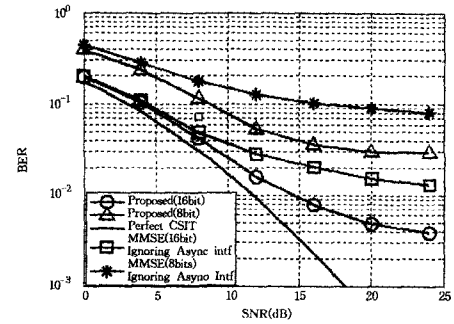

图 2 误比特性能对比

结束语 本文在考虑信道量化误差统计特性和信道时变特性的情况下，提出了一种计算复杂度较低的 BD 波束成形方法。此方法能够有效地抑制由于量化误差和信道时变特性所带来的干扰，提高多小区系统的传输速率，降低系统的误码率，有效地改善小区边缘用户的用户体验。

参 考 文 献

[1] Hand D, Yu W. Coordinated Beamforming for the Multi-cell Multi-antenna Wireless System[C]// Proc. CISS. Princeton, NJ, 2008

[2] Zhang H, Mehta N B, Molisch A F, et al. On the Fundamen-tally Asynchronous Nature of Interference in Cooperative Base Station Systems[C]// IEEE International Communication Conference. Glasgow, Scotland, June 2007

[3] Zhang Hong-yuan, Mehta N B, Molisch A F. Asynchronous Interference Mitigation in Cooperative Base Station Systems[J]. IEEE Transactions on Wireless Communications, 2008, 7(1): 155-165

[4] Azzam I H, Adve R S. Linear Precoding for Multiuser MIMO Systems with Multiple Base Stations[C]// IEEE International Conference on Communications(ICC '09). June 2009

[5] Ravindran N, Jindal N. Limited Feedback-based Block Diagonalization for the MIMO Broadcast Channel[J]. IEEE J. Sel. Areas Commun., 2008, 26(8): 1473-1482

[6] Park H, Park S-H, Lee I. Weighted Sum MSE Minimization under per-BS Power Constraint for Network MIMO Systems[J]. IEEE Communications Letters, 2011, 16(3): 360-363

[7] 赵伟，袁超伟，张金波. MIMO 广播有限反馈系统中联合预编码方法 [J]. 北京邮电大学学报，2012，35(2)：24-27

[8] Huang Yong-wei, Li Qiang, Ma Wing-Kin, et al. Robust Multicast Beamforming for Spectrum Sharing-Based Cognitive Radios [J]. IEEE Transactions on Signal Processing, 2012, 60(1): 527-533

[9] Zhang Lan, Liang Ying-chang, Xin Yan, et al. Robust Cognitive Beamforming with Partial Channel State Information [J]. IEEE Transactions on communications, 2009, 8(8): 4143-4153

[10] Lee K-J, Lee I. MMSE Based Block Diagonalization for Cognitive Radio MIMO Broadcast Channels [J]. IEEE Transactions on Wireless Communications, 2011, 10(10): 3139-3144

[11] 李应博，吕国成，张晓宁，等. 基于干扰控制的多小区分布式波束成形[J]. 重庆邮电大学学报：自然科学版，2012，24(5)：599-603

(上接第 48 页)

[8] Kumar R, Takai S. Inference-Based Ambiguity Management in Decentralized Decision-Making: Decentralized Diagnosis of Discrete-Event Systems [J]. IEEE Transactions on Automation Science and Engineering, 2009, 6(3): 479-491

[9] Stankovic S S, Ilic N, Stankovic M S, et al. Distributed Change Detection Based on a Consensus Algorithm [J]. IEEE Transactions on Signal Processing, 2011, 59(12): 5686-5697

[10] Mosk-Aoyama D, Shah D. Fast Distributed Algorithms for Computing Separable Functions [J]. IEEE Transactions on Information Theory, 2008, 54(7): 2997-3007

[11] Lin Xiao, Boyd S. Fast linear iterations for distributed aeraging [A]// Decision and Control, 2003. Proceedings. 42nd IEEE Conference on[C]. vol. 5, Dec. 2003: 4997-5002