纠错码拜占庭容错 Quorum 中错误检测机制*)

刘 钢 周敬利 秦磊华 陈小平
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 在大规模存储系统中,拜占庭存储节点的容错显得越来越重要。传统拜占庭 Quorum 通过复制可以容忍拜占庭失效,但是它们有两个主要缺点:低的存储空间利用率和静态 quorum 参数。我们提出纠错码拜占庭容错 Quorum(Erasure-code Byzantine Fault-tolerance Quorum, E-BFQ),E-BFQ 采用纠错码作为冗余策略,可以提供高可靠性,同时比复制占用更少存储空间。通过客户端读/写操作和管理器诊断操作,E-BFQ 可以检测拜占庭节点,动态调整系统规模和故障阈值。结果显示本文方法可以达到动态调整的目的。

关键词 故障检测,纠错码,拜占庭容错,Quorum

A Fault Detection Mechanism in Erasure-code Byzantine Fault-tolerance Quorum

LIU Gang ZHOU Jing-Li Qin Lei-Hua CHEN Xiao-Ping
(Computer Department of Huazhong University of Science and Technology, Wuhan 430074)

Abstract Fault-tolerance is increasingly significant for large-scale storage systems in which Byzantine failure of storage nodes may happen. Traditional Byzantine Quorum systems that tolerate Byzantine failures by using replication have two main limitations: low space-efficiency and static quorum variables. We propose an Erasure-code Byzantine Fault-tolerance Quorum that can provide high reliability with far lower storage overhead than replication by adopting erasure code as redundancy scheme. Through read/write operations of clients and diagnose operation of supervisor, our Quorum system can detect Byzantine nodes, and dynamically adjust system size and fault threshold. Simulation results show that our method improves performance for the Quorum with relatively small quorums.

Keywords Fault detection, Erasure code, Byzantine fault-tolerance, Quorum

随着分布式存储系统规模越来越大,存储节点失效也越来越频繁。现在存在两种较多失效:崩溃和拜占庭失效。当一个存储节点被认为是拜占庭节点时,它的行为可能任意偏移它的说明,表现在:拒绝响应请求、发送错误消息、存储错误信息[1]。

传统 Byzantine Quorum 协议[2~5]已经在拜占庭失效上做了深入的研究,但是目前拜占庭 Quorum 系统有两个主要缺陷。首先,拜占庭 Quorum 系统主要采用复制作为冗余策略。尽管复制提供了高可靠性和性能,存储空间开销也很大。其次,Quorum 系统的参数不能动态变化。拜占庭 Quorum 系统依赖一个故障阈值 b 限制系统中拜占庭节点的最大数目。为了容忍最坏情况,传统 Quorum 系统保守地选择一个较大数值作为故障阈值,导致系统中副本数目远大于实际需要。固定的 Quorum 规模也不能动态反映存储节点加入与退出 Quorum 系统的情况。

本文提出 E-BFQ 提供高可靠存储服务。E-BFQ 有 3 个主要优点。1、E-BFQ 使用纠错码作为冗余策略,存储空间开销远小于传统拜占庭 Quorum 系统。2、E-BFQ 可以动态增加和减少故障阈值,因此减少了容忍拜占庭失效所需存储节点数目。3、E-BFQ 运行时可以检测到拜占庭节点。实验结果表明 E-BFQ 中读操作所需参与的存储节点数目大大小于静态 Quorum 中读操作所需节点数目,提高了性能。

1 E-BFQ

1.1 E-BFQ 系统结构

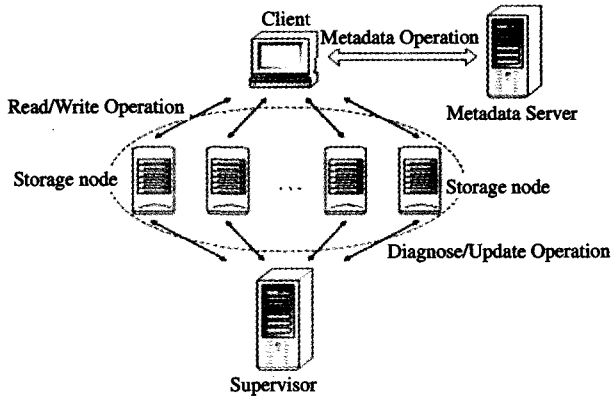

图 1 E-BFQ 结构

图 1 说明了 E-BFQ 的结构由 4 个部分组成:任意数量的客户、元数据服务器、管理服务器和存储节点集合。存储节点负责存储数据段,并提供给客户机访问接口。客户机通过基于 TCP 的 RPC 接口与存储节点通信,读取或者写入数据。当一个客户察觉到一个存储节点的非正常行为,它将立即告知管理服务器。元数据服务器维护对象的元数据。管理服务器负责监视存储节点并收集存储节点状态。当检测到拜占庭节点,管理服务器更新系统规模和故障阈值。

1.2 容错机制

在存储系统中客户机与存储节点都有可能发生拜占庭失效。但是存储系统的安全模型已经对客户机恶意攻击存储节

*)本文得到国家自然科学基金项目"基于冗余智能存储通道的简约容灾存储系统关键技术研究"(60373088)资助。刘 钢 博士生,研究方向为计算机网络存储;周敬利 教授,博导,研究方向为计算机多媒体网络通信,高性能网络接口和计算机网络存储;秦磊华 副教授,研究方向为计算机网络存储;陈小平 博士生,研究方向为计算机多媒体网络通信。

点的情形进行了深入分析，在此我们仅考虑存储节点可能发生拜占庭失效的情况。并且假设元数据服务器与管理服务器不会发生任何失效，一个正确的存储节点将对所有请求做出正确响应。

在 E-BFQ 中存储节点采用 *m-of-n* 纠错码[6~8]作为冗余策略。一个数据对象被划分为 m 个数据段，根据这 m 个数据段可以计算出 $n-m$ 个校验数据段。因此每个数据对象可以产生 n 个数据段，而每个存储节点存储其中一个数据段。这些数据段中任何 m 个数据段都可以译码出原始数据。

为了保证客户机所读取的数据段没有被篡改，需要检验这些数据段的完整性。在 E-BFQ 中根据每个数据段计算出一个交叉校验码。N 个数据段的交叉校验码聚合为$\{cc_1,\cdots,cc_n\}$并存储在每个存储节点中。当读取一个对象时，客户机首先通过选举读取$\{cc_1,\cdots,cc_n\}$。然后通过比较从某个存储节点中接受的数据 D 所计算出的 CC 于 D 在$\{cc_1,\cdots,cc_n\}$中所对应的交叉校验码 CC，客户机可以判断 D 是否正确。如果 CC 与 CC 相等，则 D 被证实有效。否则 D 已经被篡改。这个方法可以容忍拜占庭 Quorum 系统中最多 b 个错误$\{cc_1,\cdots,cc_n\}$。如果大于 $b+1$ 个存储节点返回最新时间戳与相同$\{cc_1,\cdots,cc_n\}$，可以确认已获得数据段正确的交叉校验码。

时间戳与$\{cc_1,\cdots,cc_n\}$等信息记录在对象属性中，对象属性将保存在存储有对象数据段的所有存储节点中。

1.3 E-BFQ 参数

为了保证每个读/写操作可以正确地进行，读/写 quorum 的参数需要满足一些条件。我们假设一个系统采用 *m-of-n* 纠错码作为冗余策略，系统中包括 N 个存储节点，其中最多有 b 个存储节点是拜占庭节点。在一个写操作中，客户将给 E-BFQ 中所有 N 个节点发送数据，写 quorum Q_w 满足

$$|Q_w|=N$$

E-BFQ 中需要有足够多的正确存储节点完成写操作。假设其中 N_C 个节点完成了写操作，那么在一个异步系统中可能有 $N-b$ 个节点响应一个写操作。N_C 个正确节点和 b 个拜占庭节点可能响应一个读操作。

$$N_C+b\leqslant N-b$$

$$N_C\leqslant N-2b$$

为了保证一个读操作可以获取最少 m 数据段，读操作和写操作至少要有 $m+b$ 个交叉存储节点。为了保证交叉校验码未被篡改，读操作和写操作至少要有 $2b+1$ 个交叉存储节点。我们定义

$$k=\max(m,b+1)$$

一个读 quorum Q_r 和一个写 quorum Q_w 必须交叉有至少 $k+b$ 个存储节点。

$$|Q_r\cap Q_w|\geqslant k+b$$

最坏的情况，读操作中包括了最多 $N-N_C$ 个没有存储正确数据段的存储节点，因为读请求最少需要访问 $k+b$ 个正确存储节点，所以读 quorum 满足

$$|Q_r|-(N-N_C)\geqslant k+b$$

$$|Q_r|\geqslant k+3b$$

这意味着一个读请求至少需要访问 $k+3b$ 个存储节点。

2 E-BFQ 中操作

E-BFQ 中读和写操作经过 3 个阶段完成。1)读取 Quorum 参数，这个阶段客户将得到 Quorum 规模与故障阈值。2)写操作中生成交叉校验码，读操作中交叉校验码证实有效。3)数据段从 quorum 中读出或写入到 quorum 中。

2.1 写操作

Quorum 参数定义为$\langle N, b\rangle$，N 代表 Quorum 规模，b 代表估计故障阈值。Quorum 规模是 E-BFQ 中存储节点数目，当新的存储节点加入或者拜占庭节点删除时，Quorum 将发生变化。故障阈值是介于$[b_{\min},b_{\max}]$的整数。E-BFQ 中的每个存储节点保持一份 Quorum 参数副本。在 GET_VARIALBES 函数中，客户向一个拥有 $3b_{\max}+1$ 个存储节点的 quorum 请求 Quorum 参数。如果 quorum 中至少 $2b_{\max}+1$ 个节点返回 Quorum 参数，客户选择有着最新时间戳并被 $b_{\max}+1$ 个节点返回的参数。为了决定数据的最新时间戳，可以向所有存储节点发送读时间戳的请求，然后收集 $N-b_{\max}$ 个响应，读取时间戳的值。

原始数据分块为 m 个数据段，这些数据段随后编码为 n 个数据段，并计算这些数据段的校验码。客户选择一个最新时间戳并把该时间戳与校验码写入到相关数据段的属性中。最后数据段与属性在函数 STORE 中发送到每个存储节点中。当客户收到至少 $N-b$ 个存储节点的响应，写操作结束。

```
WRITE(Data)
1) v←GET_VARIABLES()
2) if v=FALSE return FALSE
3) get the newest timestamp ts
4) encode Data into{D1,…,Dn}
5) generate attribute attr
6) STORE({D1,…,Dn},attr)

GET_VARIALBES():
1)   Randomly choose a quorum Qr s. t. |Qr|=3bmax+1
2) Read⟨V,TSV⟩from storage nodes in Qr
3) max←the returned greatest TSV
4) if⟨V,max⟩is returned by at least bmax+1 storage nodes then
5)   return⟨V, max⟩
6) else
7)   return FALSE

STORE({D1,…,Dn},attr)
1) for all storage nodes Nj in Quorum
2)   send⟨Di,attr⟩ to Nj, where Di∈{D1,…,Dn}
3) end for
4) if N−b storage nodes reply OK
5)   return TURE
6) else
7)   return FALSE
```

2.2 读操作

读操作中读取 Quorum 参数的过程与前面写操作一致。一旦 Quorum 参数被读入，客户通过 GET_NODESET 选择一组可以返回正确数据段的存储节点。

客户选择一个有 $k+3b$ 个存储节点的读 quorum 并向这些节点发送读请求。存储节点将响应一个数据段与属性，属性中包括时间戳与交叉校验码。我们使用 *attr* 代表属性，*attr. ts* 代表时间戳，*attr. cc* 代表交叉校验码。在所有响应中，客户首先找出拥有最新时间戳的正确校验码。如果超过 $k+b$ 个存储节点返回相同最新时间戳与相同校验码，客户考虑该时间戳与校验码可信。基于交叉校验码，从响应中得到的数据段可以被证实有效。如果证明之后存在 m 个正确数据段，客户可以通过解码恢复原始数据段，此时读操作完成。

当客户不能从存储节点中获得正确数据段，它把这些存储节点报告给管理服务器。管理服务器分析被报告存储节点的行为，判断他们是否失效。

```
READ()
1) v←GET_VARIABLES()
2) if v=FALSE then return FALSE
3)      ⟨NodeSet, FaultSet, timestamp⟩←GET_NODESET()
4) if ResultNULL then
5) choose m nodes from NodeSet and get {D1,…,Dm}
6)   decode{D1,…,Dm}
7)  send(⟨FaultSet, timestamp⟩) to supervisor

GET_NODESET()
1) Randomly choose a quorum Qr s. t. |Qr|=k+3b
2) Read attr from storage nodes in Qr
3) max←the returned greatest attr. ts
```

```
4) ResponseSet=Qr
5) if no more than k+b storage nodes in ResponseSet return max then
6)    return ⟨NULL,Qr,NULL⟩
7) remove the storage nodes from ResponseSet that don't return max
8) if no more than k storage nodes in ResponseSet return the same attr.cc then
9)    return ⟨NULL,Qr,NULL⟩
10) remove storage nodes in ResponseSet with different attr.cc
11) for all storage node Ni in ResponseSet
12)    read Data Segment Di from Ni
13)    if the hash value of Di is not corresponding to the value in attr.cc then
14)       remove Ni from ResponseSet
15) end for
16) if |ResponseSet|<m
17)    return ⟨NULL,(Qr−ResponseSet),attr.ts⟩
18) choose a NodeSet that contains m storage nodes in RsponseSet
19) return ⟨NodeSet,(Qr−ResponseSet),attr.ts⟩
```

2.3 诊断操作

因为一个 E-BFQ 中每个存储节点被包含写请求中，一旦存储节点在读请求中不能正确响应请求就有可能是拜占庭节点。此时，客户发送⟨*NodeSet*, *timestamp*⟩到管理服务器，其中 *NodeSet* 代表不能正确响应请求的节点集合，*timestamp* 代表数据最新时间戳。在一段时间内，如 200 秒，大于一定数目 N_{fault} 声明某个存储节点出错，该节点就被诊断为拜占庭节点并从 E-BFQ 中删除。

存在一个存储节点因为加入系统时间晚于最新时间戳而不能返回正确值的可能。为了避免这种情况下节点被诊断为拜占庭节点，管理服务器比较时间戳与节点加入系统时间。当节点加入系统时间早于时间戳，存储节点被怀疑失效数目加 1。

Quorum 参数根据诊断结果更新。当有新节点加入或拜占庭节点删除时，Quorum 规模随之改变。当 50s 内，系统中有 2 个或以上客户机报告存储节点，该存储节点被怀疑是拜占庭节点，管理服务器需要修正故障阈值 b。故障阈值的估计值等于一段时间内存在的拜占庭节点与被怀疑节点数目加 1 作为安全范围。

在 E-BFQ 中，只有管理服务器有权利设置 Quorum 参数，Quorum 参数也存在自身的时间戳。管理服务器更新 Quorum 参数的方法与 Quorum 系统中更新数据的方法相同。

DIAGNOSE(NodeSet,timestamp)

```
1) if timestamp=NULL then
2)    system error
3) else
4)    for each storage node Ni in NodeSet
5)       if timestamp is larger than the time that Ni joins then
6)          Ni.count++=1
7)    end for
8) for each storage node Ni in NodeSet
9)    if Ni.count>2 in 50 s then update b
10)   if Ni.count>M in 200 s then Ni is fault
11) end for
```

3 实验结果

我们假设一个存储系统拥有 32 个存储节点。该存储系统的冗余策略是 10-*of*-32，故障阈值范围是[1,5]。每个存储节点成为拜占庭节点是独立事件，概率为 0.05。读/写请求到达率为 0.25，其中读操作百分比为 70%。选择 $N_{fault}=4$，我们比较 E-BFQ 的读 quorum 与采用 10-*of*-32 静态 quorum。静态 quorum 的 Quorum 规模 N_s 与故障阈值 b_s 保持不变。这里 $N_s=32, b_s=5$。

图 2 显示 E-BFQ 读 quorum 规模随着检测到拜占庭节点而增大，拜占庭节点删除而减小。大多数时候 E-BFQ 读 quorum 规模时 13，大大低于静态 Quorum。因此 E-BFQ 可以请求更少存储节点，提高读性能。

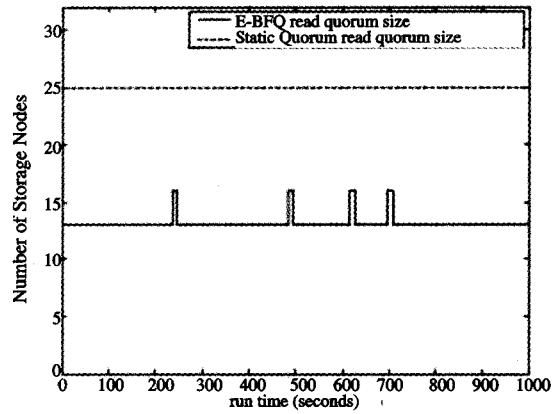

图 2　E-BFQ 与静态 Quorum 读 quorum 对比

选择合适 N_{fault} 值对于 E-BFQ 的效率非常重要。当 N_{fault} 选取过大时，管理服务器需要更多的时间来证实一个拜占庭节点。而 N_{fault} 选取过小时，网络或者 I/O 错误都有可能使一个正常节点被误认为使拜占庭节点。在此，我们通过比较不同 N_{fault} 与 b 情况下检验出一个拜占庭节点所需要的平均延时，来确认适当的 N_{fault} 值。

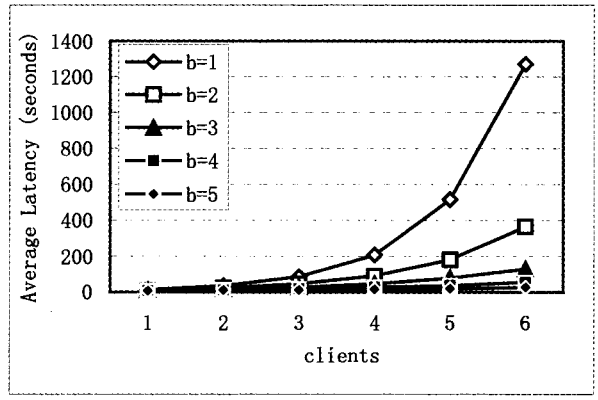

图 3　N_{fault} 声明一拜占庭节点的平均延时

图 3 描述了探测拜占庭节点的平均延时，b 越小，则读操作需要参与的存储节点越少，因此存储节点被包含在一个读操作中的机会也越少，而导致参测拜占庭节点需要消耗更多时间。从图 3 可以看出，当 N_{fault} 小于 6 时，对于不同的 b，平均延时都是可以接受的。

为了避免一个正常节点仅仅因为网络或者 I/O 错误被误认为是拜占庭节点，我们需要选取的 N_{fault} 值可以容忍此类错误。这意味着正常节点出现不能正确响应请求而被 N_{fault} 个客户机报告所需要的平均时间应该远大于探测出拜占庭节点所需要的平均时间。我们假设一正常节点可以正确响应客户请求的概率是 p_c，而因为网络和 I/O 故障而不能正确响应的概率是 $1-p_c$。表 1 描述了不同 p_c 情况下 N_{fault} 个客户报告一正常节点的平均时间。

表 1　正常节点发生 N_{fault} 次失效的平均时间

p_c	$N_{fault}=2$	$N_{fault}=3$	$N_{fault}=4$
0.90	49 s	487 s	4866 s
0.95	98 s	1946 s	38929 s
0.99	487 s	48661 s	4.8×10^6s

从表 1，我们可以观察到 4 个客户向管理服务器报告一个正常节点的平均时间远大于 4 个客户报告一个错误节点。

因此 $N_{fault}=4$ 被我们认为是探测错误节点合适的值。

总结 本文中，我们提出了 E-BFQ 容忍存储节点的拜占庭失效。相比传统拜占庭 Quorum 系统，E-BFQ 减少了提供正确数据的存储节点数目，利用纠错码提高了存储空间利用率，并且 E-BFQ 可探测故障节点并动态调整故障阈值，从而减少了读 Quorum 规模，提高了效率。

参 考 文 献

1 Kim J M, Manabe Y. A Byzantine Fault-Tolerant Mutual Exclusion Algorithm and its Application to Byzantine Fault-Tolerant Storage Systems. In: Proceedings in 25th IEEE Conference on Distributed Computing Systems Workshops, 2005

2 Malkhi D, Reiter M K. Byzantine quorum systems. Distributed Computing, 1998, 11(4): 203～213

3 Alvisi L, et al. Dynamic Byzantine Quorum Systems. In: Proceedings of Internationa conference on Dependable Systems and Networks, 2000

4 Goodson R G, et al. Efficient Byzantine-tolerant erasure-coded storage. In: 2004 International Conference on Dependable Systems and Networks, 2004

5 Kong L, et al. Agile Store: Experience with Quorum-based Data Replication Techniques for Adaptive Byzantine Fault Tolerance. In: IEEE Symposium on Reliable Distributed Systems, 2005

6 Cooley J A, et al. Software-based Erasure Codes for Scalable Distributed Storage. In: Proceedings of the 20 th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03), 2003

7 Zhang Z, Lian Q. Reperasure: Replication Protocol using Erasure-code in Peer-to-Peer Storage Network. 2002

8 Frolund S, et al. A Decentralized Algorithm for Erasure-Code Virtual Disks. in Dependable Systems and Networks, 2004

(上接第 37 页)

下面我们近一步探讨控制信道存在误帧情况下 MR-ARQ 的性能。假定每个确认帧仅能确认一个周期的发送数据。在 MR-ARQ 机制中，当数据 ω 个周期都未收到确认，则进入超时重传状态，发方自动重发该帧。那么当控制信道的误帧概率为 p_c，周期仍然为 T_{MR} 时，重发一个数据帧所需的时间为：

$$t_{RT}=\frac{T_{MR}}{\delta}(1-p_c)+\left(\frac{T_{MR}}{\delta}+T_{MR}\right)\cdot p_c(1-p_c)+\cdots+\left[\frac{T_{MR}}{\delta}+(\omega-1)\cdot T_{MR}\right]\cdot p_c^{\omega-1}(1-p_c)+\left[\frac{T_{MR}}{\delta}+(\omega-1)\cdot T_{MR}\right]\cdot p_c^{\omega}$$

$$=\frac{T_{MR}}{\delta}+T_{MR}(1-p_c)\sum_{i=1}^{\omega-2}i\cdot p_c^i+T_{MR}(\omega-1)p_c^{\omega-1} \quad (11)$$

结合式(4)式和(11)式，在考虑控制信道误帧的条件下，正确传送一个数据帧所需的平均时间为：

$$t_{AV}=t_f+(1-p_f)\sum_{i=1}^{\infty}ip_f^it_{RT}=t_f[1+(\Psi-1)p_f]/(1-p_f) \quad (12)$$

其中 $\Psi=\frac{t_{RT}}{t_f}$ (13)

此时归一化吞吐量的表达式为：

$$\rho_{CE}\leqslant t_f/t_{AV}\leqslant(1-p_f)/(1-p_f+\Psi\cdot p_f) \quad (14)$$

可见，控制信道的误帧对系统的吞吐率会产生很大的影响。实际上，我们采用的 MR-ARQ 机制，由于确认效率足够大，一定程度上减小了控制误帧对系统性能的影响。仿真结果表明，采用 MR-ARQ 机制的系统对控制信道误帧的适应性，优于采用传统的 ARQ 机制的系统。

综上，我们可以得出以下结论：在误码率较高、突发误帧较多的无线通信系统中，采用 MR-ARQ 机制要比传统的 ARQ 机制要好。

3 系统级仿真

根据以上分析，我们采用二状态 Markov 的仿真信道模型进行了系统级仿真。着重仿真了采用上行集中动态带宽分配和 MR_ARQ 的系统(优化后的系统)对 TCP 业务的支持性能。

仿真中，设用户数为 5，每个用户一个 TCP 连接，每个 TCP 连接的业务量仍是 10M 字节，则总业务量就是 50M 字节。对于 FTP 下载业务，仿真显示下行信道 10%误帧率，上行信道零误帧时，进行优化后的系统和未进行优化的系统，在 TCP 业务吞吐量的比较存在一定的差异，如图 2 所示。

图 2 中，优化后的系统平均吞吐率曲线比未进行优化的系统平均吞吐率有一定的提高。这可以由仿真数据计算出的平均吞吐率来说明，未优化系统的平均吞吐率是 27.2Mbps，进行优化后的系统平均吞吐率是 29.6Mbps，后者比前者增加了 8.82%。且由于出现了大量的多次重传，未进行系统性能优化的系统存在较大抖动，流量不明显。

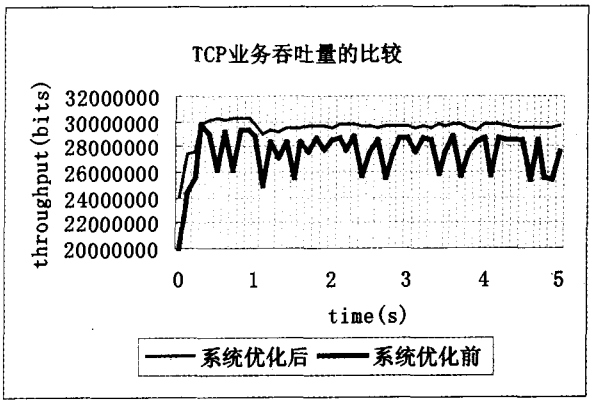

图 2 系统性能优化前后 TCP 业务吞吐量的对比图

此外，采用了上行集中动态带宽分配的系统对带宽资源的有效使用率达到了 82.3%以上，这相对于现有的 IEEE 802.11a 等固定宽带无线接入系统 60%左右的带宽利用率来讲，已经有了很大的提高。

结束语 本文所论述的上行集中动态带宽分配和 MR_ARQ 机制已经成功运用于 5.8GHz 固定无线接入系统的项目中，经过验证，改进后的宽带无线接入系统具有如下性能优点：

采用上行集中动态带宽分配的策略后提高了带宽利用率，减少了冲突造成的误帧。

采用多拒绝 ARQ 机制(MR_ARQ)，系统对控制信道误帧不敏感，尤其是对采用 TCP 提供的传输服务的业务造成的性能影响基本可以忽略。在 5%的控制信道误帧率和 10%的数据信道误帧率条件下，系统吞吐量仍然可以接近理论值。

以上结果表明，采用了上行集中动态带宽分配和多拒绝 ARQ 机制的宽带固定无线接入系统在系统容量上有了较大的提高。

参 考 文 献

1 Masamura T. Towards 4th generation mobile communications. In: CIC2002, Monte Carlo Resort, USA, 2002. USA: CSREA Press, 2002

2 Kumar V. Wireless communications beyond 3G. Alcatel Telecommunications Review, 1st Quarter 2001

3 徐昌彪. 无线应用中 TCP 拥塞裁决技术的研究和分析. 计算机工程, 2001, 27(4): 98～100

4 周天翔，蒋铃鸽，铁玲，等. 通过 ARQ 提高无线网络中的 TCP 吞吐量. 上海交通大学学报，2002，36(5)：633～636