Accurate Timeout Detection Despite Arbitrary Processing Delays

1. **关于论文**

该篇论文是由Sixiang Ma和Yang Wang（两人均在The Ohio State University）在2018年7月12日发表于2018 USENIX Annual Technical Conference(USENIX ATC’18)，属于“可用性”这一分类。

在计算机技术中，超时广泛用于故障检测[1, 2, 3, 4, 5]，常用发送“心跳包”来确定发送方是否仍在“存活”。即在规定的时间内，接收方如果没有按时收到发送方的“心跳包”信息，那么就会发送错误报告，判断发送方出了问题，从而解决错误。

然而超时检测并非那么精确，因为当超时现象发生的时候，我们很难确定到底是发送方故障了，还是“心跳包”在传输的路途中被耽搁了。

本着精益求精的学术研究态度，该篇论文在这一背景下，提出了一种新的思路，引入了一种新的机制：SafeTimer，并将SafeTimer和现有两种超时检测机制其一的“设置长超时间隔”进行了性能比较。

1. **关于SafeTimer**

SafeTimer是一种增强现有超时检测协议的机制，它可以容忍操作系统和应用程序中的长时间延迟：在心跳接收器端，SafeTimer会在报告故障之前检查是否有任何未决的心跳；在发件人方面，如果发件人无法及时发送心跳，SafeTimer会阻止发件人。并且经过作者证明，尽管操作系统和应用程序存在任意延迟，SafeTimer仍可以防止错误的故障报告。此属性允许现有协议放宽其时序假设，并使用更短的超时间隔来更快地检测故障。评估表明，SafeTimer的开销可以忽略不计，将SafeTimer应用于现有系统很容易。

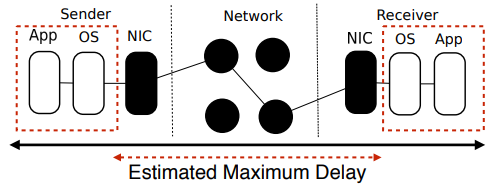
作者在github上进行说明，SafeTimer的基本思想是可用的，但是有关于如何构建和使用SafeTimer的文档还没有完成。

要了解SafeTimer，我们就应该先了解故障检测。

故障检测是用来保障系统的正确性的，在确保系统的正确性方面，现有两种机

制，一种是基于Paxos [6,7, 8].算法的共识，另一种是设置一个长超时间隔。比如Apache公司的Zookeeper、分布式锁服务Chubby、Google的Spanner数据库，就是使用基于Paxos算法的共识。Paxos算法是能够基于一大堆完全不可靠的网络条件下却能够可靠确定地实现共识一致性的算法。而Hadoop分布式文件系统（HDFS）、分布式存储系统Ceph、Hadoop资源管理器等依赖的技术是设置长超时间隔。设置长超时间隔可以最大容忍处理延迟，但是它的问题是有损系统的可用性，因为如果是发送方出来问题，我们不能及时做出处理、恢复。

SafeTimer就是一种在不影响系统的正确性前提下缩短超时间隔的一种机制。把在信息传输过程中，只能提供接收到数据包信息的部分称为“黑盒”，还能提供数据包被挂起或丢弃信息的部分称为“白盒”，比如操作系统和应用程序部分。而SafeTimer就是由于在操作系统和应用程序上的长延迟（比如磁盘I/O能达到10秒、JVM的垃圾回收能够达到26秒、一些应用程序（如HDFS）的特殊延迟（如心跳信息发送前的目录删除）甚至可以达到几分钟[9]）以及“白盒”的特征给解决问题创造了更好的解决方法这两点，进而产生的。



上面我们讲到，在心跳接收器端，SafeTimer会在报告故障之前检查是否有任何未决的心跳。这个过程的实现分为三个步骤：

（1）重定向心跳信息和障碍包信息到STQueue

（2）发送障碍包信息到一个特定的RX Queue里

（3）强制障碍包通过NIC

如果能够成功接收到障碍包，则说明在此之前发送的心跳包肯定没有被挂起。如果接收方没有收到心跳包，那只能说明要么发送方没有发送，要么心跳包被丢弃了。此时需要获取当时的包丢弃数，如果包丢弃数为0，那么就能说明接收方没有问题。

在发件人方面，如果发件人无法及时发送心跳，SafeTimer会阻止发件人。设置一个时间戳，表示在之前，发件人可以发送心跳，每当成功发送心跳，就会更新值。如果当前时间大于，则就会阻止发件人。

1. **在虚拟机中的应用**

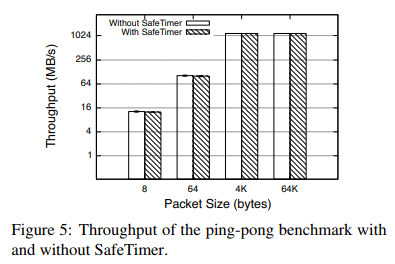
为了在虚拟机体系结构中最大化SafeTimer的好处，我们可以在主机OS或管理程序中实现SafeTimer，并使用超调用或远程过程调用向应用程序提供相关函数。

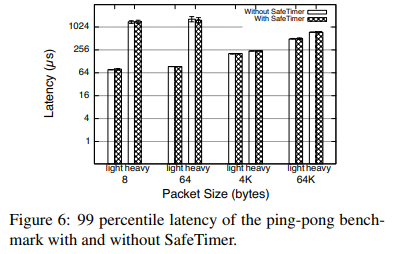
1. **试验台设置**

作者在CloudLab上进行的所有实验。每台机器配备了两个英特尔Xeon E5-2630八核CPU、128GB内存、1.2TB的SAS HDD和双端口英特尔X520-DA2 10GB的NIC。所有的机器都连接到Cisco Nexus C3172PQS交换机，所有的机器都安装的是Linux 4.4.0。

1. **开销**

SafeTimer的开销很小，对于4KB和64KB的数据包，开销小于1%；对于8B和64B的数据包，SafeTimer可以增加0.7%到2.7%的潜伏期，减少1.6%到2.4%的吞吐量。这样低的开销是合理的，因为SafeTimer的附加工作（即在发送方比较，在接收方读取目的端口）与操作系统必须对每个数据包执行的其他工作（比如中断处理、内存拷贝）相比是很小的。





1. **相关工作**

表明了许多分布式系统中的金典问题，比如共识，可以用一个精确完整的故障检测器[10]来解决。在实践中，超时被广泛用于故障检测，这一准确性取决于它们的时序假设。

在同步系统下，比如HDFS，SafeTimer可以增强同步系统，在没有要求任何时序假设的条件下容忍OS和应用程序中的异常事件。

在异步系统下，比如Spanner，与同步复制系统相比，Paxos在复制和消息的数量方面代价更高。异步系统不需要为了正确性的精确故障检测，但是有一个恢复失败的代价，SafeTimer可以通过减少错误失败报告的数量来有助于减少这种不必要的恢复。

一些系统在不使用超时的情况下实现了故障检测，比如Falcon [11]及其后续作品[12, 13]。在路由器中安装探针监视服务器并且在服务器的不同层安装探针以监视上层。这种方法本质上把整个通信通道转换成一个“白盒”，因此，它需要对路由层的侵入性修改，如果路由器超出了用户的控制权，这使其部署具有挑战性并且有时是不可能的。为了解决这些问题，Falcon使用超时作为备份。

实时Linux及其他实时Linux框架，比如RTAI和Xenomai可以保证重要任务或者中断在规定期限前被调度，然而，这不足以完成我们的目标，因为长延迟不仅仅是由于调度不合时宜造成的，也是因为重要任务偶尔被繁重任务阻止的事实造成。实时调度可以寻址前一个问题，而不是后一个。

1. **总结**

本篇论文表明我们不需要在超时间隔中包含本地处理延迟的最大值。因为本地处理的“白盒”性质，尽管有任意的处理延迟，我们可以构建这一部分的有效、准确故障检测，我们的SafeTimer原型允许我们在不牺牲系统精确性的条件下，使用一个更短的超时间隔来改进系统的可用性。

1. **参考文献**

**[1]** Apache HBASE. <http://hbase.apache.org/>.

**[2]** Mike Burrows. The Chubby lock service for loosely-coupled distributed systems. In OSDI,

2006.

**[3]** Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike

Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In OSDI, 2006.

**[4]** Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. In SOSP,

2003.

**[5]** Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, and Benjamin Reed. ZooKeeper: Wait-

free Coordination for Internet-scale Systems. In USENIX ATC, 2010.

**[6]** Leslie Lamport. The Part-time Parliament. ACM Transactions on Computer Systems,

16(2):133– 169, May 1998.

**[7]** Leslie Lamport. Paxos Made Simple. ACM SIGACT News (Distributed Computing Column),

32(4):51–58, December 2001.

**[8]** B. Oki and B. Liskov. Viewstamped replication: A general primary copy method to support

highlyavailable distributed systems. In Proc. 7th PODC, 1988.

**[9]** Yang Wang, Manos Kapritsos, Lorenzo Alvisi, and Mike Dahlin. Exalt: Empowering

Researchers to Evaluate Large-Scale Storage Systems. In NSDI, 2014.

**[10]** Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable

distributed systems. Journal of the ACM, 43(2):225–267, 1996.

**[11]** Joshua B. Leners, Hao Wu, Wei-Lun Hung, Marcos K. Aguilera, and Michael Walfish.

Detecting Failures in Distributed Systems with the Falcon Spy Network. In SOSP, 2011.

**[12]** Joshua B. Leners, Trinabh Gupta, Marcos K. Aguilera, and Michael Walfish. Improving

availability in distributed systems with failure informers. In 10th USENIX Symposium on

Networked Systems Design and Implementation (NSDI 13), pages 427– 441, Lombard,

IL, 2013. USENIX.

**[13]** Joshua B. Leners, Trinabh Gupta, Marcos K. Aguilera, and Michael Walfish. Taming

uncertainty in distributed systems with help from the network. In Proceedings of the

Tenth European Conference on Computer Systems, EuroSys ’15, pages 9:1–9:16, New

York, NY, USA, 2015. ACM.