一种基于端到端的 Ad Hoc 网络 TCP 拥塞控制改进算法

蒋道霞[1,2] 潘守伟[3] 周 曜[1] 刘凤玉[1]
(南京理工大学计算机科学与技术学院 南京 210094)[1]
(江苏财经职业技术学院计算机技术与艺术设计系 淮安 223003)[2]
(淮安工商局信息中心 淮安 223001)[3]

摘 要 提出了一种基于端到端的 Ad Hoc 网络 TCP 拥塞控制改进算法 IADTCP(Improvement AD hoc network TCP congestion control)。对现有 Ad Hoc 网络慢启动方案进行改进,以解决拥塞窗口增长不够平滑的问题;利用两连续数据包单向传输延迟差异 IDD 和短期吞吐量 STT 两个度量参数,联合判断网络拥塞状态;用丢包率 PLR 和包错序率 POR 判断信道错误、路由改变等网络状态;通过回送的 ACK 数据包携带网络状态信息,以便让发送端采取适当的控制措施。仿真结果表明,该方案是可行和有效的。

关键词 移动 Ad Hoc 网络,传输控制协议,拥塞控制,端到端测量
中图法分类号 TP393 **文献标识码** A

Improvement of TCP Congestion Control Algorithm for Mobile Ad Hoc Networks through Employment End-to-End Identification

JIANG Dao-xia[1,2] PAN Shou-wei[3] ZHOU Yao[1] LIU Feng-yu[1]
(Institute of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China)[1]
(Department of Computer Technology and Artistic Design, Jiangsu Vocational and Technical College of Finance & Economics, Huaian 223003, China)[2]
(Information Center, Huaian Administration for Industry & Commerce, Huaian 223001, China)[3]

Abstract This paper proposed an IADTCP (improvement ad hoc network tcp congestion control) algorithm, which improved TCP congestion control algorithm for mobile Ad Hoc networks through employing end-to-end identification. The IADTCP improved the slow-start strategy in traditional Ad Hoc network to resolve the problem that the congestion window grows un-smoothly, used two metrics IDD (inter delay difference) and STT (short term throughput) to perform multi-metric joint identification for congestion detection, used PLR (packet loss ratio) and POR (packet out-of-order delivery ratio) to identify the network states of CHANNEL ERR and ROUTE CHANGE. Then the sender side would take advisable measures in light of network state information carried by back ACK package. The simulation results have validated the feasibility and efficiency of the proposal.

Keywords Mobile Ad Hoc network, Ttransmission control protocol (TCP), Congestion control, End-to-end measurement

没有任何基础设施的移动 Ad Hoc 网络由若干独立节点组成,这些节点能充当源节点、目的节点、中间节点等角色,每个节点既是端节点,又是路由器。被有线网络广泛接受的传输控制协议(TCP),已成了移动 Ad Hoc 网络事实上的可靠数据传输标准。但在 Ad Hoc 网络中,无线信道的损耗、介质访问控制的低效率、路由协议的巨大开销、不同协议层之间的相互作用都会影响 TCP 性能[1]。传统 TCP 直接用于 Ad Hoc 网络,其性能不令人满意。

近年来,对移动 Ad Hoc 网络 TCP 性能的分析和改进成了研究者们关注的热点[2-11]。在 Ad Hoc 网络中要考虑由移动引起的路径中断[2,3,5]和 TCP 的公平性问题[7],基于窗口的 TCP 传输被改变为基于窗口和基于速率的混合传输[8,10]。一些研究者则关注用主动丢包来限制 TCP 的发送速率、最大窗口的极限[6]和较小窗口的增加步长[9,11]。

TCP 用慢启动和拥塞避免两个不同阶段探测动态网络的可能带宽。当接收到来自接收端的一个新的 ACK 时,若 TCP 处于慢启动阶段,发送端将当前窗口增加一个报文段大小;若处于拥塞避免阶段,则增加 $1/w$,此处 w 是当前窗口大小。这种机制适合具有较大带宽延迟乘积的有线网络。而在移动 Ad Hoc 网络中,带宽延迟乘积相对较小,过快的窗口增加策略常导致网络过载、丢包、路径重建、超时、重传等,所有这些都会减少 TCP 吞吐量。

到稿日期:2009-08-25 返修日期:2009-11-06 本文受某部委"十一五"重点项目,淮安市科技局项目(HAG07023)资助。
蒋道霞(1966-),女,博士生,副教授,主要研究方向为网络性能,E-mail:jiangdaoxia@sohu.com;刘凤玉(1943-),女,教授,博士生导师,主要研究方向为网络性能和信息安全。

本文对现有的端到端拥塞控制算法进行改进，对丢包、超时和收到的重复 ACK 等现象加以判别，以进行相应处理[12]。基于端到端的 TCP 改进方案无须中间节点的协作，能保持 TCP 协议端到端的语义，且容易在网络中实现，具有较强的适应性和背景兼容性。

1 基于端到端的拥塞控制改进方案 IADTCP

现有典型的基于端到端的 TCP 性能改进方案有固定重传超时(fixed RTO)[5]、TCP 错序检测与响应(TCP Detecting Out Of Order and Response)[13]、动态延迟 ACK (Dynamic delayed ACK)等。本文改进现有慢启动阶段拥塞窗口的增加步长，用多种度量参数交叉识别拥塞，用固定 RTO 处理路径中断等。针对拥塞、信道错误、路由改变、路径中断采用不同的方法进行传输控制，提出了基于端到端的 Ad Hoc 网络 TCP 拥塞控制改进方案 IADTCP (improvement ad hoc network tcp congestion control)。

1.1 改进 Ad Hoc 网络中的慢启动算法

TCP 试图充分利用带宽，但这容易导致网络进入拥塞，由于路由变化和难以预测的 MAC 层变化延迟等许多原因，有线网络中拥塞窗口和可达到的数据速率之间的关系在 Ad Hoc 网络中不再保持，为旧路由计算的拥塞窗口大小对于新路由来说可能太大，若发送者仍按旧的拥塞窗口全速发送数据，将很快导致拥塞。

拥塞/超载将引起缓冲区溢出并增加链路竞争，从而降低 TCP 性能，且 Ad Hoc 网络容量随流量或竞争节点的增加而降低，当拥塞发生时，传统 TCP 拥塞控制的慢启动易造成突发流量，慢启动后期 cwnd 增长过快，会很快导致网络拥塞并丢包。为了解决这些问题，对慢启动稍作修改[14]。

慢启动阶段的 cwnd 变化公式为：

$$cwnd=cwnd+(int)(2\times\log_2 N_{ACK}) \tag{1}$$

式中，N_{ACK} 是发送端累计接收到的 ACK 个数。

这一修改，可使慢启动后期的 cwnd 增长缓慢，过渡平滑，延缓拥塞发生。当数据量较少时，在拥塞发生前可完成数据发送，并能减少网络突发流量，减轻瓶颈路由器的瞬时负载。

1.2 拥塞度量参数

文献[15]用单向数据包传输时延的改变来判别网络是否拥塞，但用单一度量参数测量网络，会把不拥塞检测为拥塞的概率较高。一旦判断错误，不恰当地调用拥塞控制，将导致吞吐量严重下降[16]。本文参考文献[12,16,20]，用多个度量参数交叉确认、联合识别网络拥塞，提高网络状态识别准确度。文献[15,17]将数据包的单向传输时延(One-way Transit Time，OTT)定义为数据包从发送端到接收端所经历的时间。本文选用连续两个数据包的单向传输时延差异 IDD(Inter Dalay Defference)和短期吞吐量 STT(Short Term Throughput)两个参数联合判断网络拥塞。

1.2.1 包间延迟差异 IDD 及其测量方法

连续两个包之间的延迟差异 IDD，能在一定程度上反映前向路径的拥塞情况。每个数据包到达接收端时，接收端计算 IDD 值。文献[12]的仿真实验显示，一系列 IDD 样本值的增加反映了拥塞产生和节点队列加长，而随机信道错误和包发送速率等都不影响 IDD 值。但在 Ad Hoc 网络中，仍有一些因素，如节点移动导致的包错序影响 IDD 判断网络拥塞的准确性。

IDD 可通过发送和接收时的时间戳测得。发送端发送数据包时，给每个数据包头加上一个发送时间戳。当数据包到达接收端时，记录该数据包的接收时间戳。IDD 的计算方法如下：

$$IDD=(A^{i+1}-S^{i+1})-(A^i-S^i) \tag{2}$$

式中，A_i 是第 i 个包到达接收端的时间戳，S_i 是第 i 个包在发送端的发送时间戳。

1.2.2 短期吞吐量 STT 及其测量方法

STT 提供 T 时间段内的网络吞吐量。与 IDD 相比，STT 对一段时间内产生的错序包传输不敏感，因此 STT 对短暂的路由改变更具有鲁棒性，而路由改变在 Ad Hoc 网络中是经常发生的。但单独用 STT 作为网络拥塞的检测度量，易受突发信道错误、路径中断、源发送速率改变引起的噪音影响，所以理想的方案是将 IDD 和 STT 二者结合，联合识别网络拥塞。

STT 计算方法为：

$$STT=N_p(T)/T \tag{3}$$

式中，$N_p(T)$ 是 T 时间段内收到的数据包个数。

时间间隔 T 根据网络环境确定，其取值对算法性能的影响比较大。如果 T 取值比较小，对网络负荷状态的反应会比较灵敏，但是这将增加接收端的计算负担；如果 T 取值较大，对网络负荷的反应就不够灵敏。T 的取值公式为：

$$T=k\times RTT\times(1-PLR) \tag{4}$$

式中，RTT 是往返传输时间，PLR(Packet Loss Ratio)是丢包率，k 是一个常数。当 PLR 较大时，丢包较频繁，测量间隔要小，缩短 T；反之，延长 T。

1.2.3 丢包率 PLR 及其测量方法

在每个时间间隔 T 内，接收端计算丢包率 PLR。PLR 不仅能用来估计时间段 T，也用来检测信道错误。PLR 计算方法如下[18]：

$$PLR=\left|1-\frac{N_p(T)}{P_n-P_{n-1}}\right|\times 100\% \tag{5}$$

式中，T 是时间间隔，P_n 是 T 周期内收到的数据包的最大序号，P_{n-1} 是上一周期内收到的数据包的最大序号，T 时间段内实际收到的数据包的个数为 $N_p(T)$。

1.2.4 包错序传输率 POR 及其测量方法

如果在同一个 TCP 发送端，先发送的数据包到达接收端的时间比后发送的数据包到达的时间迟，就叫包错序(Packet Out-of-order delivery Ratio)。包错序是由路由改变引起的。在路由切换期间，可能有多个传输路径存在，沿着新路径的后发送的包，可能比沿旧路径的先发送的包先到接收端，引起包错序。

POR 计算方法为：

$$POR=\frac{N_o(T)}{P_n-P_{n-1}}\times 100\% \tag{6}$$

式中，T 是时间间隔，P_n 是 T 周期内收到的数据包的最大序号，P_{n-1} 是上一周期内收到的数据包的最大序号，$N_o(T)$ 是 T 时间段内错序包的个数。凡是发送时间戳较早的数据包，其接收时间戳迟于后发送包的，都是错序包。

接收端需维持一个数据结构，如表 1 所列，保存每一个已收到的数据包的序号、发送时间戳和接收时间戳、连续两包之间的 IDD、周期 T 的序号和长度以及 T 周期内计算的 STT，

PLR,POR 等信息。

表 1　接收端维持的数据结构

字段名	字段含义
Sequ_numr	接收包序号
S_timestamp	发送时间戳
R_timestamp	接收时间戳
OTT	单向传输时间
IDD	连续两个包之间的延迟差异
STT	短期吞吐量
PLR	丢包率
POR	包错序率
Tno	时间段序号
Tlen	时间段长度

1.3　拥塞识别与反馈

1.3.1　拥塞识别

当 IDD 较高、STT 较低时，网络拥塞；否则，网络不拥塞。

这里要为 IDD 和 STT 设置合适的门限值 high，low。当 IDD 高于 high 及 STT 低于 low 时，指示网络拥塞；其他情况下，网络一律不拥塞。high，low 值的设置非常重要，它们由具体的 Ad Hoc 环境决定。文献[12]实验显示，二者各取 30%，能取得较好的识别效果。

1.3.2　其它状态识别

在网络不拥塞的情况下，Ad Hoc 网络由节点移动导致的路由改变、路径中断和信道错误也会引起丢包，导致发送端接收到多个重复 ACK 或 RTO 超时。如果此时也调用拥塞控制算法，将严重影响 TCP 性能。

可用丢包率 PLR 和包错序率 POR 判断丢包具体原因。如果网络不拥塞，当 POR 较高且超过某个门限值时，认为路由改变引起了包错序；当 PLR 较高且超过某门限值时，则认为信道错误引起了随机丢包；当 STT 接近于 0 时，判断为路径中断。

1.3.3　接收端识别丢包原因实现方法

在正常接收到连续两数据包之后计算 IDD，每 T 周期计算一次 STT，PLR，POR 的样本值，并保存这些样本值。数据结构如表 1 所列。

每收到一个数据包，要向发送端发回一个 ACK 包。可在 ACK 包的 TCP 头中增加 3bits 网络状态反馈信息，如增加一个状态指示 SI(State Indication)选项。网络状态识别及指示如表 2 所列。

表 2　网络状态识别及指示表

网络状态	IDD 和 STT 取值	POR	PLR	SI 值
网络拥塞	(high,low)	任意	任意	001
信道错误	NOT(high,low)	任意	high	010
路由改变	NOT(high,low)	high	任意	011
路径中断	(任意,≈0)	任意	任意	100
网络正常	default			000

在 ACK 包 TCP 首部的原 6 个保留位中，用 3 位表示网络状态，分别表示网络正常(000)、网络拥塞(001)、信道错误(010)、路由改变(011)、路径中断(100)5 种状态。ACK 的 TCP 首部如图 1 所示。

当包到达接收端时，执行如下算法：

①保存数据包序号、发送及接收时间戳；

②计算样本值 T，IDD，STT，PLR，POR；

③为各参数估计 high/low，判断网络状态；

④生成 ACK 包，设置 ACK 包头状态指示 SI 选项。

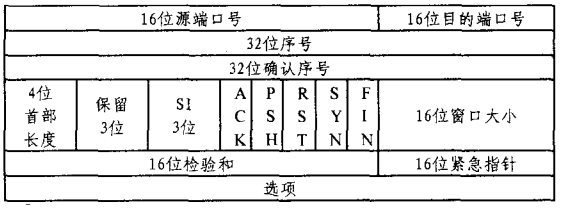

图 1　ACK 包的 TCP 首部

1.4　发送端的拥塞控制

当发送端收到 3 个重复的 ACK，或发生 RTO 超时，得知数据包丢失，便通过这些 ACK 中携带的接收端对网络状态判断的反馈信息，来判断网络是否发生了拥塞。若 ACK 包 TCP 首部的 SI 域为“001”，可知网络发生了拥塞，便启动拥塞控制算法：当收到第 3 个重复的 ACK 时，若 RTO 没有超时时，执行拥塞避免算法；若此时 RTO 超时，则将 cwnd 设置为 1 个报文段，调用慢启动拥塞控制算法。

1.5　发送端对非拥塞丢包采取的措施

发送端接收到 3 个重复 ACK 或 RTO 超时时，得知数据包丢失。但发现 ACK 中携带的网络状态指示 SI 不是“001”，可知丢包不是由网络拥塞引起，不需要调用拥塞控制算法。

1.5.1　信道错误引起的丢包处理

在 Ad Hoc 网络中多跳无线信道由于信道衰落会产生误码，突发的误码会破坏发送的包，导致 TCP 数据包丢失或 ACK 包丢失。在传统的 TCP 传输控制中，如果在 RTO 时间内没有收到 ACK，则认为发生了拥塞。TCP 发送者立即减少发送窗口为 1 个数据包，进入重传指数退避，并重传已丢失的包。断断续续的信道错误导致发送端拥塞窗口很小，从而降低吞吐量。但在 Ad Hoc 网络中，RTO 超时不一定是由拥塞引起的。在本方案中，如果接收端发生 RTO 超时，或收到 3 个重复 ACK，而从接收端收到的 ACK 包中的反馈信息得知，网络状态 SI 为“010”，认为丢包是由信道错误引起的随机丢包，不需要调用拥塞控制的慢启动，只需要重传丢失的数据包，并保持 cwnd，RTO 等变量不变。

1.5.2　路由改变引起的丢包处理

Ad Hoc 网络中节点的移动性经常会引起路由改变，由此引起的丢包也与拥塞无关。如果接收端发生 RTO 超时，或收到 3 个重复 ACK，而 ACK 包中的 SI 为“011”，可知丢包是由路由改变引起的，也不需要调用拥塞控制算法。此时，发送端也维持现有发送状态不变，重传丢失的数据包。

1.5.3　路径中断引起的丢包处理

如果接收端发生 RTO 超时，或收到 3 个重复 ACK，而 ACK 包中的 SI 为“100”，可知发生了路径中断。当路径中断时，发送端会收到多个重复 ACK，并会发生 RTO 连续超时现象。此时，用固定 RTO 策略[5]，发送端重传没有被确认的数据包，但不改变 RTO 的值，不进入指数退避，直到路由重新建立。

2　仿真与分析

2.1　传统 TCP 和改进 TCP 的慢启动拥塞窗口改变情况分析

传统 TCP 和 IADTCP 两协议的慢启动拥塞窗口(cwnd)随往返时间的改变如图 2 所示。从图中可见，传统 TCP 慢启动(slow start)拥塞窗口 cwnd 随往返时间个数的增加而增加，开始时每收到一个 ACK 增加一个报文段。因传输开始时，每个 RTT 内收到的 ACK 个数较少，所以 cwnd 增加缓

慢。而改进的慢启动(I-slow start)开始时，cwnd 增加比传统慢启动(slow start)阶段 cwnd 的增加稍快，可较充分利用带宽。

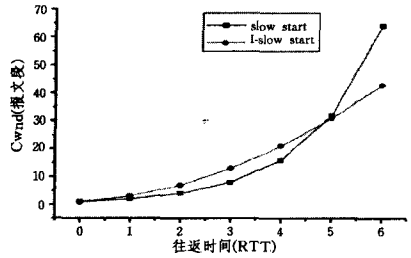

图 2　传统 TCP 和 IADTCP 慢启动算法 cwnd 变化比较

但到第 5 个 RTT 以后，传统慢启动的 cwnd 增加很快，远超过改进慢启动(I-slow start)的 cwnd 增加幅度，这易造成突发流量，从而导致网络拥塞，过早进入拥塞避免或再进入慢启动阶段，使发送端不得不降低发送速率，因而大大减少网络吞吐量。而此时 I-slow start 的 cwnd 增长较平缓，不会使网络产生突发数据流量，从而保证较高的网络吞吐量，平滑过渡到拥塞避免阶段。

2.2　IADTCP 性能仿真与分析

实验使用康内尔大学开发的 JiST/SWANS[19] 作为仿真平台。网络接口用 IEEE 802.11 DCF MAC 协议，路由协议用 AODV。仿真选用 RWP 移动模型，场景大小为 1500m×1000m 矩形区域。节点从一个随机位置开始，并随机选一个目的节点，节点最大移动速度为 20m/s，节点数 50 个，节点传输半径为 250m，带宽为 2Mbps，消息大小为 1460 字节，最大拥塞窗口和接收端通告窗口都为 8，传输层协议用 FTP。

为了验证本文提出的 IADTCP 的性能，实验首先对 NewReno TCP 和 IADTCP 协议在节点移动情况下的网络吞吐量性能进行仿真比较。在仿真场景中分别只用一个 TCP 流，实验结果如图 3 所示。

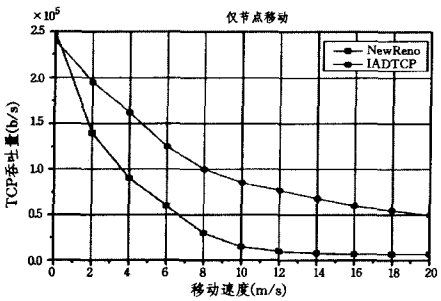

图 3　IADTCP 在节点移动场景下的性能仿真

从图 3 中可见，IADTCP 的吞吐量明显高于 NewReno TCP。当节点移动时，IADTCP 获得的吞吐量比 NewReno TCP 有 100%到 800%的提高。随着节点移动速率的增加，两种算法的吞吐量都呈下降趋势。但在相同速率下，IADTCP 的吞吐量明显高于 NewReno TCP，即 NewReno TCP 比 IADTCP 对节点移动更敏感。这是因为随着节点移动速率的增加，经常导致路径中断或路由改变，从而产生丢包或包错序，IADTCP 能正确识别这些非拥塞丢包原因，并采取适当的控制措施，维持现有传输速率和 RTO 值不变，重传丢失包或错序包，使网络吞吐量不会突然下降。而 NewReno TCP 把所有丢包一律判断为网络产生拥塞并调用拥塞控制算法，突然把发送速率减少一半(拥塞避免)或降为一个数据包(慢启动)，使网络吞吐量下降很快。

第二个实验验证移动性和信道错误对 IADTCP 协议吞吐量的影响。实验仍选用 NewReno TCP 和 IADTCP 进行比较。仿真结果如图 4 所示。实验中引入信道固定误码率为 5%，节点移动速率从 0 到 20m/s。

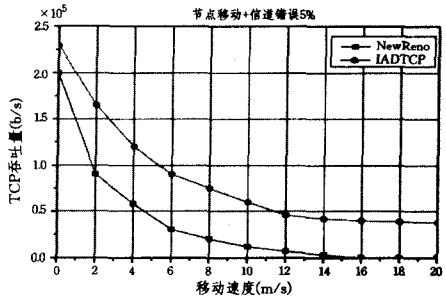

图 4　IADTCP 在节点移动和信道错误场景下的性能仿真

从图 4 中可见，与没有信道错误的第一个实验相比，两个算法获得的吞吐量都有所下降，但 IADTCP 的吞吐量性能仍明显优于 NewReno TCP。这是因为信道衰落产生误码，会破坏发送的包，导致 TCP 数据包丢失或 ACKs 丢失，发送端会收到重复 ACK 或 RTO 超时。IADTCP 能通过来自接收端的 ACK 包反馈信息正确识别这种非拥塞丢包，然后采取重传丢失包措施，并维持现有发送窗口和 RTO 值不变，这样对网络吞吐量的影响不会太大。而 NewReno TCP 则调用拥塞控制算法，立即减少发送窗口为 1 个包，进入重传指数退避，并重传丢失包，断断续续的信道错误，导致发送端拥塞窗口很小，从而降低网络吞吐量。

第三个实验则再引入周期性的拥塞。实验结果如图 5 所示。可见两种算法所得到的吞吐量与前两个实验结果相比都有大幅下降，说明网络拥塞对二者的性能都有严重影响。但 IADTCP 性能在这种情况下也优于 NewReno TCP。一是因为高误码率和网络拥塞丢包导致发送端产生较多的 RTO 超时事件，NewReno TCP 发送端把窗口减为 1 个包，进入重传指数退避，并重传丢失包；当接收到 3 个重复 ACK 且 RTO 没超时，发送端将拥塞窗口减少一半，从而使吞吐量下降。而 IADTCP 可以通过接收端 ACK 包中携带的网络状态反馈信息区分丢包是由误码产生还是由拥塞引起的。若为误码原因，则不改变现有发送状态，重传丢失包。只有当判断丢包的真正原因是网络拥塞时才调用拥塞控制算法，所以与 NewReno TCP 相比，取得的吞吐量较高。

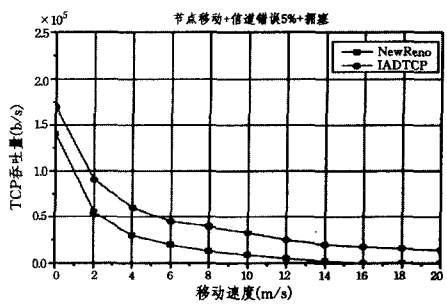

图 5　IADTCP 在节点移动、信道错误和拥塞场景下的性能仿真

结束语　通过端到端网络状态检测机制，来改善 Ad Hoc 网络中 TCP 协议的性能。新提出的 IADTCP 方案仍采用基于窗口的拥塞控制机制，对慢启动阶段 cwnd 的增加步长进行了改进，使之过渡平缓，防止网络出现突发流量；仅在发送端和接收端做了一些扩展，接收端除了正常的操作外，还要计

算接收到的数据包的 IDD,STT,PLR,POR 等参数样本值,根据 IDD 和 STT 值,判断网络是否发生拥塞;根据 PLR 和 POR 值判断非拥塞状态下的丢包原因;然后在回复的 ACK 数据包中用 3bits 携带网络状态信息,以便明确通知发送端,网络是否发生了拥塞。当发送端接收到 3 个重复的 ACK 或 RTO 超时时,得知数据包丢失,便通过这些 ACK 中携带的状态指示,采取适当的控制措施:如果是网络拥塞,便调用拥塞控制算法,否则只重传丢失的数据包;当发生路径中断时,发送端采用固定 RTO 重传策略。仿真试验验证了此方案是有效的。

本协议的优势在于它仅在发送端和接收端引入简单的计算和判断算法,没有利用过多的开销,不改变网络中其他各层协议,实现简单。但缺陷是本方案中门限值 high,low 的设置将影响它的性能。因此,设计合理、简单而有效的端到端 TCP 改进方案将是一个重要研究方向。

参 考 文 献

[1] Fu Z, Meng X, Lu S. How bad tcp can perform in mobile ad hoc networks[C]// Seventh IEEE Symposium on Computers and Communications(ISCC'02). July 2002:298-303

[2] Holland G, Vaidya N. Analysis of tcp performance over mobile ad hoc networks[C]// Proc. of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking. ACM New York, NY, USA, 1999:219-230

[3] Chandran K, Raghunathan S, Venkatesan S, et al. A feedback-based scheme for improving tcp performance in ad hoc wireless networks[C]// Proc. of the 18th International Conference on Distributed Computing Systems. Washington, DC, USA. IEEE Computer Society, 1998:472

[4] Chen K, Xue Y, Nahrstedt K. On setting tcp's congestion window limit in mobile ad hoc network[C]// IEEE International Conference on Communications, 2003(ICC '03). May 2003(2): 1080-1084

[5] Dyer T D, Boppana R V. A comparison of tcp performance over three routing protocols for mobile ad hoc networks[C]// Proceedings of the 2nd ACM International Symposium on Mobile Ad hoc Networking & Computing. ACM New York, NY, USA, 2001:56-66

[6] Fu Z, Zerfos P, Luo H, et al. The impact of multihop wireless channel on tcp throughput and loss[C]// INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, 30 March-3 April 2003 (3):1744-1753

[7] Xu K X, Gerla M, Qi L, et al. Enhancing tcp fairness in ad hoc wireless networks using neighborhood red[C]// Proc. of the 9th Annual International Conference on Mobile Computing and Networking. ACM New York, NY, USA, 2003:16-28

[8] ElRakabawy S M, Klemm A, Lindemann C. TCP with adaptive pacing for multihop wireless networks[C]// Proc. of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing. ACM New York, NY, USA, 2005:288-299

[9] Nahm K, Helmy A, Jay Kuo C C. Tcp over multihop 802.11 networks: issues and performance enhancement[C]// Proceedings of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing. ACM New York, NY, USA, 2005:277-287

[10] Zhai H Q, Chen X, Fang Y. Improving transport layer performance in multihop ad hoc networks by exploiting MAC layer information[J]. IEEE Transactions on Wireless Communications, 2007(6):1692-1701

[11] Ding L, Wang X, Xu Y, et al. Vegas-w: an enhanced tcp-vegas for wireless ad hoc networks[C]// Proc. IEEE Int. Conf. on Communications. Beijing, China, 2008:2383-2387

[12] Fu Z, Greenstein B, Meng X, et al. Design and implementation of a tcp friendly transport protocol for ad hoc wireless networks [C]// The 10th IEEE International Conference on Network Protocols. Paris, France, 2002:212-225

[13] Wang F, Zhang Y. Protocols: Improving tcp performance over mobile ad hoc networks with out-of-order detection and response[C]// Proc. of the 3rd ACM International Symposium on Mobile Ad hoc Networking & Computing. ACM New York, NY, USA, 2002:217-225

[14] Wang Q, Pu J, Liu W. Research of slow-start strategy of tcp congestion control[J]. Microelectronics & Computer, 2007, 24 (12):210-212

[15] Wu Q, Wu B, Zhang J. Improving tcp performance for mobile ad hoc networks through employing end-to-end identification[J]. Journal of Hefei University of Technology, 2008, 31(9): 1458-1461

[16] Mirhosseini S M, Torgheh F. Improvement of tcp performance in ad hoc networks using cross layer approach[C]// Proc. of the 2008 Third International Conference on Systems and Networks Communications. Washington, DC, USA: IEEE Computer Society, 2008:322-328

[17] Paxson V. Measurements and analysis of end-to-end Internet dynamics[D]. CA, USA: University of California at Berkeley Berkeley, 1998

[18] Niu Z, Xiang Y. End to end congestion control of udp stream for ad hoc networks[J]. Computer Engineering and Design, 2009, 30 (6):1301-1306

[19] JiST user guide and swans user guide [EB/OL]. http://jist.ece.conell.edu/docs.html, March 2004

[20] Fu Z, Meng X, Lu S. A transport protocol for supporting multimedia streaming in mobile ad hoc networks[J]. IEEE Journal on Selected Areas in Communications, 2003, 21(10):1615-1626

[21] Handley M, Floyd S, Padhye J, et al. Tcp friendly rate control (TFRC): Protocol Specification[S]. RFC 3448

(上接第 86 页)

[6] Henriksson D, Lu Y, Abdelzaher T. Improved prediction for web server delay control[C]// Processings of the 16th Euromicro Conference on Real-Time Systems. 2004:61-68

[7] Lu Y, Abdelzaher T, Lu C, et al. Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers[C]// Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium. 2003:208

[8] Barford P, Crovella M. Generating Representative Web Workloads for Network and Server Performance Evaluation[J]. ACM SIGMETRICS Performance Evaluation Review, 1998, 26(1): 151-160

[9] Abdelzaher T F, Bhatti N. Web Server QoS Management by Adaptive Content Delivery[C]// Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on. 1999:216-225

[10] Wei J, Xu C Z. eQoS: Provisioning of Client-Perceived End-to-End QoS Guarantees in Web Servers[J]. IEEE Transactions on Computers, 2006

[11] Wei J, Xu C Z. A Self-tuning Fuzzy Control Approach for End-to-End QoS Guarantees in Web Servers[C]// Quality of Service. IWQoS 2005 vol. 3552, 2005:123-135

[12] Diao Y, Hellerstein J L, Parekh S. Using fuzzy control to maximize profits in service level management-IBM[J]. Journal of Research and Development, 2002