[文章编号] 1007－7405(2024)03－0271－11 DOI:10.19715/j.jmuzr.2024.03.10

一种新的RUDP协议设计及其Golang实现

张 赓[1]，刘年生[1,2]

（1. 集美大学计算机工程学院，福建 厦门 361021；
2. 集美大学数字福建大数据建模与智能计算研究所，福建 厦门 361021）

[摘要] 采用先进的Golang网络编程语言，设计和实现了一种新的RUDP（reliable user datagram protocol）传输协议，部署在实际的远距离Internet网络中，对新RUDP协议进行实验验证和性能测试分析。实验结果表明：新RUDP传输协议是可行的，实现了确认、序列号、重传、拥塞控制、滑动窗口、差错检测等可靠通信机制，保证了传输数据的可靠交付；在相同的实验条件下，新RUDP的丢包率与TCP（transmission control protocol）的相同为0%；新RUDP的端到端时延和时延抖动总体上都比TCP的要小，在一些常见的通信链路中，新RUDP的端到端时延、时延抖动与TCP的之间存在显著性差异，新RUDP的带宽占有率比UDP（user datagram protocol）的显著性低；从实测结果来看，新RUDP比TCP、UDP更适合应用于这类低时延高可靠性的网络新应用。最后，对新RUDP的协议参数进行了部分优化，为这类网络新应用提供性能更好的传输服务质量。

[关键词] 可靠用户数据报协议；服务质量；网络性能评价指标；可靠通信机制；Golang语言程序设计
[中图分类号] TP 393

Protocol Analysis of New Reliable User Datagram Protocol and Its Implementation in Golang

ZHANG Geng[1], LIU Niansheng[1,2]

(1. College of Computer Engineering, Jimei University, Xiamen 361021, China;
2. Institute of Big Data Modeling and Intelligent Computing for Digital Fujian, Jimei University, Xiamen 361021, China)

Abstract: The Golang network programming language is used to design and implement a new RUDP (reliable user datagram protocol) transport protocol, which is deployed in the real long-distance Internet. The experimental verification and performance test analysis of the proposed new RUDP protocol are carried out. The experimental results show that the proposed new RUDP transmission protocol is feasible, and the reliable communication mechanisms such as acknowledgment, sequence number, retransmission, congestion control, sliding window and error detection are implemented to ensure the reliable delivery of transmission data. Under the same experimental conditions, the packet loss rate of new RUDP is 0%, the same as that of TCP(transmission control protocol). In general, the end-to-end delay and delay jitter of new RUDP are lower than those of TCP.

[收稿日期] 2023－06－25
[基金项目] 国家自然科学基金项目“针对在共公共数据平台分享隐写数据进行隐蔽通信的隐写分析方法研究”（U1936114）；国家自然科学基金项目“基于概率程序推断的少样本学习理论与应用研究”（62006096）；福建省自然科学基金项目“基于图卷积网络的图像鲁棒零水印关键技术研究”（2021J01857）；福建省自然科学基金项目“MIMO链路签名及其安全机制研究”（2017J01761）；集美大学国家基金培育计划项目“深度神经网络的反直觉对抗攻击与防御机制研究”（ZP2020044）
[作者简介] 通信作者：刘年生（1967—），博士，教授，主要研究方向为人工智能与网络通信、信息安全。E-mail：nsliu@jmu.edu.cn

There are significant differences between new RUDP and TCP in some common communication links. At the same time, the bandwidth occupancy of new RUDP is significantly lower than that of UDP (user datagram protocol). The throughput of new RUDP is bigger that of TCP, and closer to that of UDP. Therefore, new RUDP is more suitable than TCP and UDP for this kind of new network applications with low latency and high reliability. Thirdly, the protocol parameters of new RUDP are partially optimized to provide better transmission service quality for these new network applications.

Keywords: RUDP; QoS (quality of service); performance evaluation index of the Internet; reliable communication mechanism; programming in Golang

0 引言

在 20 世纪 90 年代，Internet 应用迅速扩大到包含在线多媒体传输等新领域，原有的 TCP 和 UDP 协议已经不能满足这些新应用的 QoS 要求，一些适合新应用的网络传输协议被提出[1]。1999 年，Bova 等[2]在 UDP 协议基础上加入了确认、序列号、检错和重传等可靠性传输机制，支持数据报的分段与重组；2001 年，Stewart 等[3]在 TCP 协议基础上提出一种流控制的传输协议 SCTP（stream control transmission protocol），只保留了 TCP 协议的部分传输可靠性机制，如选择性确认机制、心跳保活机制和重传机制等，精简了拥塞控制机制，以提高 TCP/IP 网络传输吞吐量，降低时延，支持多目的 IP 地址传输等。随后，Le 等[4]对 RUDP 协议进行了改进，只保留了序列号、重传和差错控制机制，用于多移动智能体通信，在 6 个移动节点无线局域网中进行了实验验证；Herrero[5]从 RTP（real time protocol）角度提出了一种流媒体的可靠传输协议，应用于无线实时通信，Visual ProtoStack 仿真显示该协议有效地降低分组丢失；2018 年，Huh[6]将 RUDP 协议用于网络游戏，解决网络传输长时延问题，其 OPNET 仿真结果显示，相比 TCP 协议，RUDP 协议可以大幅度降低网络传输的端到端时延；2019 年，Chen 等[7]对 RUDP 协议进行了改进，只保留面向连接、确认、序列号和超时重传等可靠性通信机制，用于嵌入式实时通信系统，在 ZYNQ 7 开发版上验证了所改进协议的可行性，测试和分析了其传输速率、丢包率和重传率等性能指标。

由于网络通信底层技术不断改进，有线和无线的物理链路稳定性得到较大的提升[8]，RDUP 具有更坚实的应用基础和必要性，支持包括在线直播等在内的大规模新兴应用，弥补现有 TCP 和 UDP 的不足。而现有的 RUDP 协议可行性和性能分析都是在仿真或 FPGA 开发板的实验条件下取得的，没有经过长距离的实际网络实验验证。在实际的 Internet 中采用 RUDP 实现可靠性数据传输需要考虑更多更复杂的因素，其实验结果具有更高的应用价值和学术意义。为此，设计了一种新的 RUDP 系统结构，构建多重可靠性传输机制，以支持复杂网络环境下动态自适应选择；采用网络编程 Golang 语言实现[9]，对实验结果进行比较与分析，提出优化 RUDP 传输性能的途径。

1 新 RUDP 的设计与可靠传输机制

TCP 和 UDP 协议在 TCP/IP 网络体系结构中属于传输层的协议，一般地，RUDP 主要通过改进 UDP 报头字段和传输机制来提供可靠数据交付的服务功能。

1.1 RUDP 的网络模型结构

传统的 TCP/IP 网络体系结构为四层，从顶层到底层分别为应用层、传输层、网络互联层、网络接口层。RUDP 协议是基于 UDP 作为底层传输协议，在 UDP 上一层通过算法实现可靠传输。在应用层与传输层之间增加一层，即 RUDP 层，RUDP/IP 网络模型结构如图 1 所示。

本研究的新 RUDP 数据报报头格式如图 2 所示，报头字段含义及作用如下：1）conv——会话编号，双方一致才能通信；2）cmd——指令类型；3）frg——分片编号，表示倒数第几个分片；4）wnd——本方剩余接收窗口大小，即接收队列大小；5）ts——时间戳（ms）；6）sn——确认序列号；

7）una——代表小于 una 的数据分组都接收成功；8）len——数据长度；9）data——数据内容。其中，cmd 作为指令类型分组，包含如下 4 种操作指令：1）IRUDP_CMD_PUSH 为传输数据；2）IRUDP_CMD_ACK 为应答接收到数据分组；3）IRUDP_CMD_WASK 为探测接收端接收窗口大小；4）IRUDP_CMD_WINS 为通知接收窗口大小。

从图 2 所示的报头格式可以看到，新设计的 RUDP 协议将支持确认、序列号、重传、拥塞控制、滑动窗口、差错检测等可靠通信机制，保证传输数据的可靠交付。应用程序通过新 RUDP 协议进行通信时，应用层的协议数据单元将作为 RUDP 数据报的 data，封装成一个完整的 RUDP 数据报。根据 TCP/IP 网络体系结构，RUDP 数据报再经由 UDP 层封装为 UDP 数据报，接着由网络层封装成 IP 数据报，再由网络接口层进行封装和处理，传送到目的主机的网络接口层。

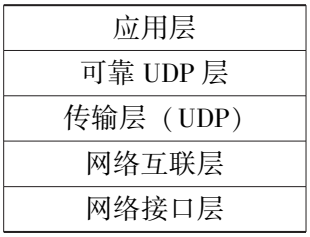

图 1　RUDP/IP 网络模型结构

Fig. 1　Model structure of RUDP-based IP network

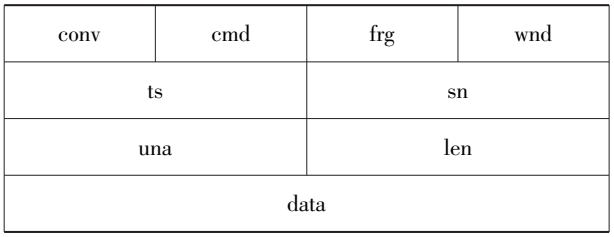

图 2　新 RUDP 数据报报头格式

Fig. 2　Header format of new RUDP datagrarm

1.2　序列号和确认机制

由于数据传输信道是不可靠的，因此接收方收到的数据分组可能出现丢失、乱序、重复等异常现象。引入序列号机制可以帮助判断数据分组传输是否出现上述问题。新 RUDP 数据报中的 sn 字段就是每个数据报的序列号，每个数据报的 sn 会等于当前的 sndNxt 值，而 sndNxt 值默认值是 0，每发送出一个数据报就递增 1。换言之，sn 序列号是从 0 开始按 1 为增量递增的。

确认机制作为 ARQ（automatic repeat request）实现可靠传输的一种关键性机制，在接收方正确收到数据分组之后会返回给发送端确认号。新 RUDP 采用的是 ACK + UNA 混合的确认机制，返回 ack 时发送的是接收到的分组的序列号和时间戳（填充到数据报头部），并填充上 IRUDP_CMD_ACK 的操作指令，data 部分为空。当发送者收到 ack 时，通过 cmd 判断出这个分组是 ack 信息，从中提取需要的信息就可以了。需要注意的是，RUDP 中有 2 种 ack 响应方式，一种是积累确认，ack 信息先写在缓冲队列中，等到合适的时间再一起发送出去；另一种是立即确认，只要有 ack 就马上发给发送端。

una 序列号保存在每个 RUDP 数据报中，当要发送数据报的时候会将 rcvNxt 的值填充给数据报的 una 字段，而 rcvNxt 的值在每次接收数据报的时候会进行更新。

1.3　超时重传

超时重传的原理是在消息发送时就启动一个计时器，如果传输过程按预期进行，发送方会在一定时间内收到接收方返回的确认信息，表示发出的分组收到了；如果因为网络波动或某种原因导致发送方在一定时间内没有收到确认信息，则被当作发出的分组对方没有收到，从而重新发送对应的分组。而这个超时的指定时间被称作重传超时时间 RTO（retransmission timeout）。

新 RUDP 采用的 RTO 计算方法是 RFC6298 中的指数加权移动平均算法[10]，在每次收到 ack 时都通过更新 RTT 来计算新的 RTO 值。一个数据报发出时会带有当时的时间戳和 RTO，计算出 RTO 超时的时间戳，当 RUDP 在定时地遍历发送缓冲时，会根据这个超时时间戳来判断是否需要超时重传，如果需要超时重传则给这个分组更新 RTO 和超时时间戳，并写入发送缓冲等待下一次发送。

1.4　快速重传

快速重传是超时重传的改进，改善了超时重传要等待超时才重传的缺陷，将时间驱动的方式改为数据驱动设置。设置 RUDP 可以快速重传的 ack 分组次数的阈值，当收到某个分组的 ack 信息次数超

过该阈值则认为该分组丢失，立即重发。与超时重传类似，快速重传一个分组需要记录这个分组 ack 信息次数的计数器，并把这个分组写入缓冲等待定时发送，收到发送成功确认信息后重置该计数器为 0，循环调用，记录新的 RUDP 分组的 ack 信息次数。

2 新 RUDP 协议数据传输流程 Golang 实现

2.1 新 RUDP 协议数据传输流程

新 RUDP 协议数据传输流程如图 3 所示，采用 Golang 语言实现这一工作流程[8,11]，包括协议结构、函数方法、调度和并行处理等。用户层面的数据存放在 sndQueue 和 rcvQueue 中，协议层面的数据存放在 sndBuf 和 rcvBuf 中。

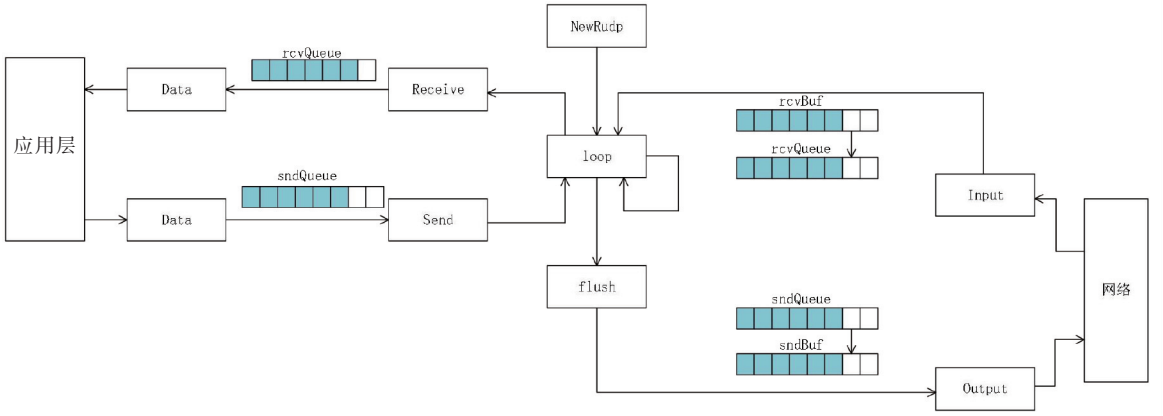

图 3 新 RUDP 协议数据传输流程

Fig.3 Workflow of new RUDP protocol

消息发送流程为应用层输入 Data 后，调用 Send 函数发送数据，将 Send 函数加入到发送队列 sndQueue，等待定时循环 loop 来调用 flush 函数，flush 函数做完相应处理后调用 Output 将数据发送到网络，而 Output 函数为回调函数，由 UDP 传输协议实现。

Send 函数会将传入的数据封装成若干个 RUDP 数据报，并计算 frg 碎片编号，之后把这些数据报加到 sndQueue 末端。代码实现如下：

```
//Send 把 buffer 中的数据转化成分片，添加到 send_ queue 末尾
func (rudp *RUDP) Send(buffer []byte) int {
iflen(buffer) == 0 {
return -1
}
var count int
//计算新分片数量
iflen(buffer) < int(rudp.mss) {
count = 1
} else {
count = (len(buffer) + int(rudp.mss) - 1) /int(rudp.mss)
}
…
fori := 0;i < count;i++ {
var size int
iflen(buffer) > int(rudp.mss) {
size = int(rudp.mss)
} else {
```

```
size = len(buffer)
}
seg: = rudp. newSegment(size)
copy(seg. data,buffer[:size])
buffer = buffer[size:]
seg. frg = uint8(count - i - 1)
rudp. sndQueue = append(rudp. sndQueue,seg)
}
return 0
```

flush 函数的任务是发送数据和更新 RUDP 状态。首先会发送 ackList 中的 ack 消息，然后根据需要发送窗口探测和通知报文；再根据滑动窗口来及时将 sndQueue 中的消息转移到 sndBuf 中，将 sndBuf 中满足条件的消息发送出去；最后根据丢包情况执行拥塞发生算法和快速恢复算法，以更新 cwnd 和 ssthresh。sndBuf 中的分片如果是第一次发送，则填充 RTO 和超时时间戳后发出；如果不是第一次发送，则判断是否触发快速重传和超时重传。flush 的返回值是所有分片的最小 RTO，将该最小 RTO 作为下一次定时执行 flush 的时间。

消息发送到对端后，RUDP 执行 Input 来将网络中的消息读取到 rcvBuf 中，并将正确的数据从 rcvBuf 转移到 rcvQueue 中；再由 loop 执行 receive 函数从 rcvQueue 中读取出数据。

Input 函数会从缓存中读取出所有数据分片。如果是合法的分片，则根据 cmd 指令来执行对应操作，然后更新 RTO 和 cwnd。当 cmd 为 IRUDP_CMD_PUSH 时，说明这是个数据分片，如果这个分片在接收窗口范围内，则接收这个分组，添加对应 ack 分组到 ackList 中；如果这个分片不是重复收到的分组，则将这个分片插入到 rcvBuf 中对应的位置，然后从 rcvBuf 中转移有序的数据到 rcvQueue 中。当 cmd 为 IRUDP_CMD_ACK 时，说明这个分组是 ack 信息，把对端确定收到的分组从 sndBuf 中移除，然后根据收到的 ack 信息去更新还没收到的分片的快速重传计数器。当 cmd 为 IRUDP_CMD_WASK 时，说明对端在询问窗口大小，下次发送分片需要带上窗口大小。当 cmd 为 IRUDP_CMD_WINS 时，说明对端在通知窗口大小，读取分片中的窗口大小即可。

2.2　基于 UDP 协议的新 RUDP 工作流程

RUDP 需要结合 UDP 才能完成会话端的工作，新 RUDP 工作流程如图 4 所示。

服务器端维护一个 map 存放正在监听的 session，新建一个客户端时会建立一个循环监听协程，当 UDP 读到消息后调用 packetInput。在 l. packetInput 中，如果 map 中存在对应的 session，则直接调用该 session 的 rudpInput，否则在 map 中新存入对应 session，然后再调用 rudpInput。协程是 Golang 语言中特有的专业术语，类似于 Java 中的线程。下面是部分代码实现：

```
func (l * Listener) defaultMonitor() {
    buf: = make([]byte,mtuLimit)
    for {
    if n,from,err: = l. conn. ReadFrom(buf);err = = nil {
    l. packetInput(buf[:n],from)
    } else {
    l. notifyReadError(errors. WithStack(err))
    return
    }
}
}
```

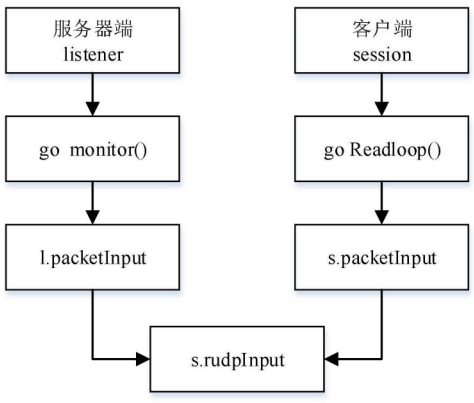

图 4　基于 UDP 的新 RUDP 工作流程

Fig.4　Workflow of new RUDP based on UDP

可以看到对消息的监听使用的是 UDP 的 ReadFrom 方法。该方法会将 UDP 读到的消息存放在 buf

中，返回值分别是读取字节数、来源地址、错误信息。

客户端启动一个循环的 Readloop 协程，思路与 monitor 类似，但是在读取消息时要判断来源地址是否正确。s. packetInput 只做一些消息的合法性判断，然后调用 s. rudpInput。s. rudpInput 先调用了 RUDP 的 input 来读取消息到 rcvBuf 和 rcvQueue 中，并将缓存中等待发送的消息调用 UDP 发送出去。部分代码实现如下：

```
func (s * UDPSession) rudpInput(data [ ]byte) {
varrudpInErrors uint64
s. mu. Lock( )
ifret: = s. rudp. Input(data,true,s. ackNoDelay);ret ! = 0 {
rudpInErrors + +
}
ifn: = s. rudp. peekSize( );n >0 {
s. notifyReadEvent( )
}
waitSnd: = s. rudp. WaitSnd( )
ifwaitSnd < int(s. rudp. sndWnd) &&
waitSnd < int(s. rudp. rmtWnd) {
s. notifyWriteEvent( )
}
s. uncork( )//UDP 使用 WriteTo 发送消息
s. mu. Unlock( )
}
```

其中，notifyReadEvent 和 notifyWriteEvent 能够通过 Golang 特有的 channel 实现事件通知，通知 Read 和 Write 函数分别去开始工作并调用 RUDP 的 Receive 和 Send 函数。如果在一定时间内没有收到对应的事件通知，读写函数会抛出读写超时错误。

3 新 RUDP 性能测试与分析

3.1 实验条件和方案

1）实验条件：将本研究开发的 RUDP 部署在 2 台云服务器，一台在中国杭州，另一台在美国洛杉矶，云服务器操作系统为 CentOS 7. 6 64 位，2 核虚拟 CPU（vCPU），内存为 2 GB。安装有 tcnetem、Ethr、iperf 等实验模拟和测试工具。本地终端设备为华硕笔记本，操作系统为 Windows 10 专业版，CPU 为 Intel（R）Core（TM）i7 - 875H，主频 2. 20 GHz，内存 16 GB。

2）实验方案：重点评估 RUDP 在不同数据载荷、不同链路质量下的这 2 台云服务器之间网络传输性能；数据载荷分为 1 B 和 5 KB，分别测试数据小于分组最大长度 MTU（maximum transmission unit）和大于 MTU 的情况。对于所有测试，窗口大小均为 1 024 个，缓冲区大小为 4 096 000 B，MTU 为 1 400 B，数据块来回发送次数 20。通信链路的基准性能用时延和丢包率来表示，时延分为 10、50、100、200 ms四档，丢包率为 0% ~30%，每间隔 5% 为一档。

3）端到端网络传输性能评价指标：采用带宽占用率、时延、时延抖动、丢包率等作为网络传输性能评价指标[12-13]，从多个不同的角度对 RUDP 网络的端对端传输性能进行评价。对每个测试点的性能指标，采用 n 组平行实验，实验结果表示为：$\mu = \bar{x} \pm 2s$；$(x + a)^n\bar{x} = \sum_{i=1}^{n} x_i/n$；$s = \sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2/(n-1)}$。其中：$\mu$ 为评价指标的测量值（置信度为95%），x_i 为性能指标第 i 次实验测试结果，n 为平行实验次数；本文中 $n = 7$，$\bar{x}$为 x_i 的平均值，s 为 x_i 的标准偏差。

3.2　实验结果与分析

当应用层数据载荷为 1 B 和 5 KB 时，RUDP 和 TCP 传输的时延、时延抖动的实验结果如表 1 和表 2 所示。

表 1　RUDP 和 TCP 在不同通信链路状态下发送 1 B 应用数据时的端到端时延和时延抖动

Tab. 1　End-to-end delay and jitter of RUDP or TCP-based network for sending 1 B application data under different communication links

通信链路状态		端到端时延/ms		端到端时延抖动/ms	
链路时延/ms	丢包率/%	RUDP	TCP	RUDP	TCP
10	0	10.20 ± 0.20	10.20 ± 0.10	0.07 ± 0.21	0.03 ± 0.04
50	0	50.30 ± 0.10	50.20 ± 0.10	0.05 ± 0.07	0.02 ± 0.06
100	0	100.30 ± 0.10	100.30 ± 0.20	0.01 ± 0.01	0.11 ± 0.23
200	0	200.30 ± 0.10	200.30 ± 0.20	0.04 ± 0.06	0.09 ± 0.18
10	5	11.60 ± 2.10	13.20 ± 10.30	1.00 ± 1.60	7.00 ± 10.80
50	5	55.31 ± 8.00	62.90 ± 0.30	4.70 ± 5.00	0.16 ± 0.31
100	5	112.60 ± 21.00	128.50 ± 56.30	14.20 ± 11.50	12.80 ± 47.20
200	5	218.90 ± 27.10	223.70 ± 55.40	14.60 ± 25.00	27.30 ± 43.40
10	10	16.70 ± 8.90	46.60 ± 48.40	6.50 ± 8.20	20.70 ± 51.90
50	10	62.60 ± 9.50	81.40 ± 31.80	5.90 ± 6.20	19.00 ± 28.50
100	10	112.80 ± 31.50	136.70 ± 32.80	13.70 ± 24.80	16.60 ± 26.60
200	10	227.60 ± 44.90	241.10 ± 57.30	40.70 ± 29.00	31.10 ± 66.00
10	15	14.60 ± 6.60	61.90 ± 58.40	5.80 ± 2.60	40.50 ± 45.50
50	15	67.50 ± 18.70	113.80 ± 69.70	12.70 ± 15.6	46.10 ± 63.40
100	15	146.10 ± 105.30	162.60 ± 74.80	33.30 ± 104.0	48.80 ± 58.10
200	15	261.60 ± 112.00	250.70 ± 89.60	42.00 ± 104.8	55.90 ± 65.40
10	20	15.70 ± 10.70	66.60 ± 61.40	6.90 ± 13.40	41.80 ± 55.10
50	20	76.10 ± 10.80	115.40 ± 55.30	4.90 ± 11.60	33.30 ± 62.20
100	20	136.90 ± 46.20	218.40 ± 78.60	30.50 ± 41.30	32.60 ± 74.20
200	20	275.40 ± 63.70	368.20 ± 245.50	32.20 ± 49.20	146.90 ± 287.60
10	25	19.60 ± 9.00	209.2 ± 385.00	5.90 ± 6.70	150.70 ± 354.00
50	25	75.70 ± 33.40	172.80 ± 106.10	21.70 ± 23.10	59.40 ± 95.70
100	25	141.30 ± 33.60	238.10 ± 142.70	19.90 ± 30.10	95.60 ± 147.00
200	25	274.10 ± 86.60	454.00 ± 346.60	68.50 ± 59.60	200.40 ± 432.40
10	30	25.20 ± 16.70	205.80 ± 423.30	5.60 ± 8.40	156.30 ± 355.60
50	30	85.50 ± 24.00	290.30 ± 309.80	13.20 ± 12.60	192.10 ± 370.50
100	30	181.80 ± 55.10	240.10 ± 79.70	30.00 ± 41.30	44.80 ± 65.40
200	30	318.70 ± 130.00	503.50 ± 598.30	73.10 ± 100.40	243.50 ± 476.30

从表 1 中可知，当通信链路的丢包率为 0% 时，RUDP 和 TCP 的端到端传输时延均与通信链路设定的基准时延相接近，它们在时延上没有显著性差异；但是，在同一设定基准时延的通信链路中，随着链路的丢包率增大，RUDP 和 TCP 两种传输的时延均呈现出非线性波动变化的特性，总体上是增大的。

在同一通信链路中，RUDP 传输的时延均值在绝大多数情况下比 TCP 的要小；并且当通信链路的丢包率在 10% 及其以上时，采用双样本 T 检验（置信区间为 95%），除了通信链路时延为 200 ms、丢包率 15% 和时延 100 ms、丢包率 15% 的这两种情况外，RUDP 的时延与 TCP 的时延存在显著性差异。

同时，RUDP 和 TCP 的时延抖动随着通信链路质量的变化比较复杂，当通信链路的丢包率超过 10% 时，两者之间的时延抖动存在显著性差异，RUDP 的时延抖动均值比 TCP 的要小。

在表 2 中，当用户数据大于 5 KB 时，超过了接收方的 MTU，如以太网的分组数据为 1 500 B，就出现分组分片问题；因此，在相同的通信链路中，5 KB 的时延均值比 1 B 的要大；时延和时延抖动随着通信链路变化的特性与表 1 中用户数据为 1 B 时的相类似。在同一基准时延的通信链路中，随着通信链路的丢包率增大，RUDP 和 TCP 两种传输的端到端时延均呈现出非线性波动变化的特性，总体上是增大的，但 RUDP 传输的时延均值、时延抖动均值在多数情况下比 TCP 的要小。

表 2 RUDP 和 TCP 在不同通信链路状态下发送 5 KB 应用数据时的端到端时延和时延抖动

Tab. 2 End-to-end delay and jitter of RUDP or TCP-based network for sending 5 KB application data under different communication links

通信链路状态		端到端时延/ms		端到端时延抖动/ms	
链路时延/ms	丢包率/%	RUDP	TCP	RUDP	TCP
10	0	10.30 ± 0.10	10.30 ± 0.50	0.04 ± 0.07	0.20 ± 0.70
50	0	50.60 ± 0.50	50.50 ± 0.80	0.26 ± 0.48	0.50 ± 0.80
100	0	100.40 ± 0.30	100.40 ± 0.60	15.00 ± 0.35	0.40 ± 0.60
200	0	200.41 ± 0.10	200.30 ± 0.10	0.03 ± 0.04	0.10 ± 0.10
10	5	14.20 ± 4.50	17.90 ± 16.20	3.00 ± 4.30	10.70 ± 13.40
50	5	63.40 ± 13.30	59.30 ± 19.00	7.60 ± 13.80	147.00 ± 10.10
100	5	116.10 ± 16.70	127.90 ± 68.00	8.50 ± 15.30	341.00 ± 85.20
200	5	248.00 ± 39.20	214.70 ± 19.60	23.10 ± 27.20	10.10 ± 21.70
100	10	17.70 ± 5.40	28.70 ± 49.70	2.20 ± 5.00	27.00 ± 45.30
50	10	74.20 ± 15.50	72.80 ± 14.30	8.10 ± 14.4	8.30 ± 10.40
100	10	146.10 ± 42.20	157.20 ± 64.70	32.40 ± 40.50	35.60 ± 63.00
200	10	296.40 ± 79.70	267.10 ± 102.30	40.00 ± 870.00	80.70 ± 76.60
10	15	23.90 ± 6.50	40.90 ± 27.90	3.40 ± 6.80	7.90 ± 17.50
50	15	93.20 ± 19.80	119.10 ± 113.30	15.10 ± 16.30	78.60 ± 119.00
100	15	183.30 ± 25.70	170.70 ± 81.90	18.30 ± 17.70	47.10 ± 65.80
200	15	324.80 ± 74.50	319.90 ± 383.80	53.10 ± 56.00	190.50 ± 511.60
10	20	34.20 ± 48.00	84.80 ± 80.10	21.90 ± 61.40	46.60 ± 79.20
50	20	106.90 ± 23.30	127.80 ± 91.20	14.50 ± 13.70	40.00 ± 64.70
100	20	215.50 ± 80.90	224.40 ± 131.30	431.00 ± 64.90	90.10 ± 104.80
200	20	397.30 ± 142.30	381.70 ± 210.60	85.90 ± 139.60	99.80 ± 183.10
10	25	64.20 ± 20.80	71.60 ± 87.60	10.20 ± 12.50	61.80 ± 65.20
50	25	109.30 ± 23.20	175.30 ± 154.70	15.60 ± 20.50	114.20 ± 187.20
100	25	230.70 ± 56.10	306.70 ± 249.70	40.70 ± 38.10	130.90 ± 170.90
200	25	416.90 ± 61.60	401.40 ± 179.30	45.90 ± 23.40	110.90 ± 177.90
10	30	44.70 ± 41.00	269.40 ± 587.70	16.60 ± 31.50	357.30 ± 657.90
50	30	144.00 ± 54.20	293.20 ± 20.11	28.10 ± 23.80	80.50 ± 99.50
100	30	246.70 ± 80.00	314.20 ± 299.00	46.80 ± 80.00	188.80 ± 363.00
200	30	482.60 ± 62.50	445.10 ± 421.20	52.50 ± 54.00	213.40 ± 341.50

当通信链路的带宽为 100 Mbit/s 时，在 2 种链路基准性能（延时 40 ms、丢包率 20%；延时 200 ms、丢包率 30%）条件下，采用 RUDP、TCP 和 UDP 三种方式分别传输相同的一批文件，文件大小从 8 ~ 1 118 MB，它们的带宽占用率、重传率和丢包率测试结果如表 3 所示。

表 3　RUDP、TCP 和 UDP 在 2 种通信链路状态下带宽占用率、重传率和丢包率

Tab. 3　Bandwidth occupancy, retransmission rate and packet loss rate of RUDP, TCP or UDP-based network in two communication link states　　单位:%

通信链路状态		带宽占用率/%			重传率/%		丢包率/%	
链路时延/ms	丢包率/%	RUDP	TCP	UDP	RUDP	TCP	RUDP	UDP
40	20	35.20 ± 5.60	26.80 ± 16.60	99.00 ± 6.20	19.90	0.35	10.70	33.00
200	30	7.00 ± 2.20	2.50 ± 9.00	99.00 ± 6.20	28.90	0.63	15.90	25.00

在表 3 中，RUDP、TCP 的带宽占用率与 UDP 的带宽占用率之间存在着显著性差异（置信度为 95%），而 RUDP 的带宽占用率与 TCP 的带宽占用率之间无显著性差异（置信度为 95%）。因此，在相同的通信链路基准状态下传输同一批数据文件，UDP 的带宽占用率在 90% 以上，而 TCP 和 RUDP 的相对较低。为了分析这一实验结果的原因，对 iperf 网络测试工具的源码进行了拓展，给 RUDP 增加了 DefaultSNMP 模块，监听和统计 iperf 客户端与服务器之间的通信信息，结果发现 UDP 存在较高的丢包率，分别为 33.0% 和 25.0%；RUDP 的分组重传主要是由链路基准丢包造成的，重传后丢包率为 0%。同时，iperf 监测结果发现 RUDP 的分组重传率比自身的丢包率要高，这意味着 RUDP 的分组重传率除与通信链路基准丢包率密切相关外，还与协议相关参数的设置密切相关，如快速重传阈值、重传计时器更新方式等，如果这些参数值设置过小，就会在传输途中、接收待处理的数据报分组或确认分组中误判为丢失，导致 RUDP 重传率升高。

4　新 RUDP 参数优化

为了减少对分组丢失的误判，提高新 RUDP 实际的数据传输性能，降低时延，对新 RUDP 的部分参数进行了优化。

4.1　调整快速重传阈值

RUDP 可以根据需求手动调节快速重传的重传阈值，以获得流速提升的效果。快速重传阈值 fastResend 参数为 0 时，表示关闭快速重传。不同快速重传阈值在常规 Wi-Fi 环境下（通常指无线链路的传播延迟≤40 ms、丢包率 20%）与较差 Wi-Fi 环境下（通常指无线链路的传播延迟≥80 ms、丢包率≥40%）的传输性能测试结果如表 4 所示。

表 4　不同 Wi-Fi 环境下快速重传阈值对传输性能的影响

Tab. 4　Impact of fast retransmission thresholds on transmission performance in different Wi-Fi environments

快速重传的阈值	端到端时延/ms		传输速率/(Mbit · s^{-1})	
	常规 Wi-Fi 环境	较差 Wi-Fi 环境	常规 Wi-Fi 环境	较差 Wi-Fi 环境
fastResend = 0	990 ± 296	2913 ± 1129	467 ± 148	206 ± 119
fastResend = 1	871 ± 265	2935 ± 516	509 ± 141	229 ± 91
fastResend = 2	799 ± 224	2738 ± 1141	549 ± 146	232 ± 99
fastResend = 3	982 ± 290	2602 ± 792	514 ± 80	233 ± 88

从表 4 实验结果得到以下 2 个结论：1）fastResend 大于 0 时就开启快速重传，其时延比关闭快速重传时要低，而传输速率明显增加。这里存在一个新的潜在问题，却在所传输的数据报在信道中没有丢失，且接收端成功接收并返回了 ack。如果 ack 返回得较慢也可能被认定为丢包而发生重传，造成多余的重传，造成部分宽带资源的浪费，因此合理预设 fastResend 的阈值有助于避免这部分带宽浪费。此外，在常规 Wi-Fi 环境下，fastResend 的阈值呈现凸函数特性，最优值在 2 左右。2）无线链路质量对 fastResend 参数阈值设定影响较大，在较差的较差 Wi-Fi 环境下，fastResend 的阈值呈现单调函数特性，以阈值较大一些为宜，传输性能指标相对好一些。

4.2 调整超时重传计数器更新方式

超时重传机制能够让 RUDP 在一个分组超过 RTO 的时间都没收到确认的情况下认定这个分组丢失然后重发，而不管这个分组究竟是丢失、正在传输途中，还是对端 ack 丢失、对端接收处理太慢。TCP 中一个分组重传一次后新的 RTO 是原来 RTO 的两倍，这样做能够尽可能多地利用带宽，但是如果这个分组又一次丢失了，它的超时时间会不断翻倍。为了提高流速，RUDP 把 RTO 的更新机制换成可以通过超时重传计数器参数让 RTO 变成原来的 1.5 倍而不是 2 倍，适当降低了 RTO 翻倍系数。

不同超时重传参数在常规 Wi-Fi 环境、较差 Wi-Fi 环境下的传输效率测试结果如表 5 所示。当启用 noDelay 选项后，因为 RTO 的增长方式不再是一直翻倍，导致 RUDP 数据报的时延有明显降低，而数据传输速率变化不明显。

表 5 不同 Wi-Fi 环境下超时重传计数器对传输性能的影响

Tab.5 Impact of timeout retransmission counter on transmission performance in different Wi-Fi environments

超时重传计数器参数	端到端时延/ms		传输速率/(Mbit · s^{-1})	
	常规 Wi-Fi 环境	较差 Wi-Fi 环境	常规 Wi-Fi 环境	较差 Wi-Fi 环境
noDelay = 0	99.0 ± 29.6	273.8 ± 114.1	51.0 ± 13.8	23.2 ± 9.9
noDelay = 1	81.9 ± 25.0	259.5 ± 58.4	51.0 ± 12.9	22.9 ± 10.1

4.3 关闭拥塞控制

发送窗口大小指的是能够同时发送的分组的数量，窗口太小，则发送效率低，窗口越小等待确认所占的时间比例就越大，停止等待 ARQ 可以看作是窗口大小为 1 的特例。而窗口太大，容易造成网络拥堵，窗口无限大则看作可以毫无节制地发送数据，而网络拥堵，丢包率就上升，重传的分组增加了，数据传输速率就降低了。

拥塞控制可以根据网络实时状况动态调整窗口大小，以达到流量无私退让的效果。RUDP 可以通过拥塞窗口参数来关闭拥塞控制，让发送窗口固定大小，不再考虑网络中的其他用户，通过资源独占策略来降低时延、提升传输速率。不同拥塞窗口参数在常规 Wi-Fi 环境、较差 Wi-Fi 环境下的传输效率测试如表 6 所示。当 noCwnd = 1 关闭拥塞控制后，时延有明显缩短，且有效的数据传输速率显著提高；关闭拥塞控制虽然可以减少维护窗口的过程开销，但提高了网络拥塞的风险。

表 6 不同 Wi-Fi 环境下拥塞控制对传输性能的影响

Tab.6 Impact of congestion control on transmission performance in different Wi-Fi environments

拥塞窗口参数	端到端时延/ms		传输速率/(Mbit · s^{-1})	
	常规 Wi-Fi 环境	较差 Wi-Fi 环境	常规 Wi-Fi 环境	较差 Wi-Fi 环境
noCwnd = 0	102.60 ± 43.60	366.40 ± 141.90	2.70 ± 8.09	1.80 ± 8.40
noCwnd = 1	87.00 ± 25.34	273.80 ± 114.00	54.90 ± 14.60	23.20 ± 9.90

5 结论

针对网络新应用的低时延和高可靠服务质量需求，设计和实现了一种新的 RUDP 协议，支持确认、序列号、重传、拥塞控制、滑动窗口、差错检测等多重可靠通信机制，保证传输数据的可靠交付。经实际 Internet 远程传输测试，实验结果证明新 RUDP 协议是可行的，具有良好的重传机制，与 TCP 的一样，丢包率为 0%；同时，新 RUDP 的端到端时延和时延抖动总体上都 TCP 的要低，在一些常见的通信链路中两者之间存在显著性差异；新 RUDP 的带宽占有率比 UDP 的低，并存在显著性差异，这意味着新 RUDP 可比 UDP 提供更高的有效数据传输速率。新 RUDP 将比 TCP、UDP 更好地满足这些网络新应用的服务质量要求。

在不同基准性能的通信链路下，对新 RUDP 的协议参数进行了部分优化，提高了新 RUDP 的传输

性能，降低带宽的浪费。未来可以与智能算法相结合，自适应设置新 RUDP 的协议参数值，为网络的新应用提供性能更好的传输服务。

[参 考 文 献]

[1]CHAPMAN A,KUNG H T. Enhancing transport networks with Internet protocols[J]. IEEE Communications Magazine,1998,36(5):100-104.

[2]BOVA T,KRIVORUCHKA T. Reliable UDP protocol [EB/OL]. (2022-10-02)[2023-06-25]. https://tools. ietf. org/id/draft-ietf-sigtran-reliable-udp-00. txt.

[3]STEWART R,METZ C. SCTP:new transport protocol for TCP/IP [J]. IEEE Internet Computing,2001,5(6):64-69.

[4]LE T,KUTHETHOOR G,HANSUPICHON C,et al. Reliable User Datagram Protocol for airborne network [C]//Proceedings of the 2009 IEEE Military Communications Conference. New York:IEEE,2009:1-6.

[5]HERRERO R. Reliable transmission of stream transported media in wireless real time communications [J]. Wireless Networks,2019,25:4727-4736.

[6]HUH J H. Reliable user datagram protocol as a solution to latencies in network games [J]. Electronics,2018,7(11):1-18.

[7]CHEN Y Y,HU S,LI X M,et al. An improved RUDP for data transmission in embedded real-time system[C]. //Proceedings of the 2019 IEEE International Conference on Signal,Information and Data Processing. NEW York:IEEE,2019:1-5.

[8]LIU W,QUEVEDO D E,JOHANSSON K H,et al. Stability conditions for remote state estimation of multiple systems over multiple markov fading channels[J]. IEEE Transactions on Automatic Control,2023,68(7):4273-4280.

[9]MEYERSON J. The go programming language [J]. IEEE Software,2014,31(5):104-101.

[10]PAXSON V,ALLMAN M,CHU J,et al. Computing TCP's retransmission timer [S]. IETF,2011:1-11. https://datatracker. ietf. org/doc/html/rfc628[2024-07-02].

[11]FAVA D S,STEFFEN M. Ready,set,Go! Data-race detection and the Go language [J]. Science of Computer Programming,2020,195:1-23.

[12]QIAN HH,CHEN Y Q,SUN Y D,et al. Vehicle safety enhancement system:sensing and communication [J]. International Journal of Distributed Sensor Networks,2013,2013:1-19.

[13]WANG Y C. Construction and simulation of performance evaluation index system of Internet of Things based on cloud model [J]. Computer Communications,2020,153:177-187.

(责任编辑　朱雪莲　彭海滨　英文审校　黄振坤)