计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(5): 1337−1349, 2026

面向高速传感数据采集的轻量化嵌入式 RDMA 软件栈

杨 惠[1] 张毓涛[1] 黄曼蒂[1] 原玉磊[1] 汪 清[2] 李 韬[1]
[1](国防科技大学计算机学院 长沙 410073)
[2](天津大学电气自动化与信息工程学院 天津 300072)
(huihui19870124@126.com)

Lightweight Embedded RDMA Software Stack for High-Speed Sensor Data Acquisition

Yang Hui[1], Zhang Yutao[1], Huang Mandi[1], Yuan Yulei[1], Wang Qing[2], and Li Tao[1]
[1](*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073)
[2](*School of Electrical and Information Engineering*, *Tianjin University*, *Tianjin* 300072)

Abstract Remote direct memory access (RDMA) technology is widely used in high-performance computing and intelligent computing scenarios that require high bandwidth and low latency transmission, offering significant advantages over traditional TCP/IP protocol stacks in terms of bandwidth, efficiency, and latency. It has gradually become an industry consensus for high-speed sensors to access computing networks through RDMA technology. However, deploying RDMA in this scenario faces challenges due to the limited resources of embedded sensor nodes, making it difficult to deploy traditional software stacks. We conduct research on lightweight techniques for embedded RDMA software stacks, proposing a lightweight RDMA connection establishment method based on reliable UDP and a lightweight RDMA driver based on reliable Ethernet under conditions without high-performance CPU or operating system. An ultra-lightweight RDMA software stack suitable for embedded sensor nodes is designed. The experimental results show that the lightweight RDMA software stack for embedded systems can reduce the connection establishment overhead by approximately 94% and the software driver overhead by 84% compared with the standard RDMA software stack. The embedded sensor nodes and standard nodes use RDMA technology to perform sensor data acquisition and reliable transmission through one-level switching, with a system bandwidth of up to 98.04 Gbps, end-to-end latency as low as 4.58 μs, and CPU overhead of standard nodes within the system as low as 4.2%. In terms of protocol compatibility, as the number of queue pairs and data volume increase, the READ and WRITE bandwidths of embedded RDMA nodes can reach the peak performance of standard nodes. Meanwhile, the READ and WRITE latency remains at the microsecond level when the data volume is small.

Key words sensor data; embedded; RDMA; lightweight; software stack

摘 要 远程直接内存访问(remote direct memory access, RDMA)技术广泛应用于超算以及智算所需的高带宽和低延迟传输,在带宽、效能和延迟方面与传统 TCP/IP 协议栈相比具备显著优势。高速传感器通过 RDMA 技术接入计算网络已逐渐成为工业界共识。然而,在该场景中应用 RDMA 存在嵌入式传感端资源受限、软件栈难以部署的问题。对嵌入式 RDMA 软件栈轻量化技术展开研究,提出了无高性能 CPU 和

收稿日期: 2025-06-17; **修回日期**: 2025-12-19
基金项目: 国家国防科技工业局重点实验室基金项目(WDZC20245250113)
This work was supported by the Key Laboratory Foundation of State Administration for Science, Technology and Industry for National Defense(WDZC20245250113).
通信作者: 李韬(taoli_network@163.com)

无操作系统条件下基于可靠 UDP 的轻量化 RDMA 建链方法和基于可靠以太网的轻量化 RDMA 驱动,设计了适用于嵌入式传感端的超轻量化 RDMA 软件栈。实验结果表明,轻量化嵌入式 RDMA 软件栈相较于标准 RDMA 软件栈可降低约 94% 的建链开销和 84% 的软件驱动开销,嵌入式传感端与标准端基于 RDMA 技术经由一级交换进行传感数据采集与可靠传输,系统带宽可达 98.04 Gbps,端到端延迟低至 4.58 μs,系统内标准端 CPU 开销低至 4.2%。在协议兼容性方面,嵌入式 RDMA 端的 READ 和 WRITE 带宽随队列对(QP)数量和数据量的增加均能够达到标准端的峰值性能,READ 和 WRITE 延迟在数据量较小时保持在微妙级水平。

关键词 传感器数据;嵌入式;RDMA;轻量化;软件栈

中图法分类号 TP393

DOI: 10.7544/issn1000-1239.202550472 **CSTR:** 32373.14.issn1000-1239.202550472

远程直接内存访问(remote direct memory access, RDMA)凭借其零拷贝、内核旁路和 CPU 卸载的设计思想,可实现高带宽、低延迟、低 CPU 开销的网络传输服务,并已在高性能计算、数据中心[1]等领域得到广泛关注和研究。随着无人驾驶[2]、工业物联网[3]等技术的快速发展,各类传感设备的种类、数量和传输带宽持续增加,自动驾驶紧急避障和工业产线急停等安全关键场景下,对智能无人信息处理平台的传感数据传输和融合处理能力提出了严苛的要求。

传统总线方式的传感数据采集与处理存在结构烟囱化、宽带受限、可扩展性较差等问题,难以实现高通量传感数据的融合传输。传统以太网通信方式网络协议栈的软件处理开销过大,计算资源占用率高且延迟高,难以满足高清摄像头视频流和相控阵雷达等传感数据的高带宽和低延迟融合传输需求[4]。将在高性能计算等领域得到广泛关注 RDMA 技术扩展应用到高速传感数据采集与融合传输场景,可满足智能无人等系统中高带宽、低延迟、低算力开销的网络传输需求,实现多源传感数据的高性能可靠传输。

然而,当前 RDMA 相关研究主要聚焦面向数据中心应用领域的网卡计算资源[5]和存储资源管理[6]、流量控制[7]和拥塞控制[8]、分布式存储系统[9]和多租户性能隔离[10]等相关优化工作,在面向高速多源传感数据采集等嵌入式场景下的 RDMA 相关研究较少。现有数据中心领域相关研究主要关注 RDMA 硬件层面优化,通过将 RDMA 网卡相关处理逻辑和存储资源上载软件[11]、转移至对端[12]等方式尽可能减少硬件处理开销,这在数据中心场景中具备高性能 CPU 和完备操作系统的情况下是可行的,但在接入高速传感数据采集的嵌入式场景中,受限于嵌入式系统有限的计算和存储资源,嵌入式软件常使用轻量化操作系统或在无操作系统的裸机环境下执行,并不具备高性能 CPU 以支持通用操作系统和标准 RDMA 软件栈。

现有嵌入式领域相关研究主要面向航空电子系统[13]、分布式系统集群[14]、网络连接加速器[15]、超声数据采集[16]等领域展开。在软件层面,文献 [13] 直接针对航空电子系统通信语义将 RDMA libverbs 软件库进行迁移应用;ERNIC[17] 使用 Microblaze 软核处理器和裁剪版 Ubuntu 操作系统实现一定程序轻量化;eRDAC[14] 则通过轻量化中断处理程序获取远程访问完成通知以消除完成队列(completion queue, CQ)相关数据结构和处理线程的存储和计算开销。文献 [13] 直接复用标准 RDMA 软件栈,并未做轻量化处理,难以满足实际嵌入式场景需求;ERNIC 实际上仅对 RDMA 软件栈依赖的底层支撑操作系统进行轻量化,未对 RDMA 软件栈本身做轻量化处理;eRDAC 从 CQ 的角度对 RDMA 完成处理进行了轻量化实现,但只聚焦该单一模块优化,未从整体软件架构层面进行轻量化设计,对于高速传感数据采集这一传输导向明确的嵌入式场景而言仍存在功能冗余。

在此背景下,为应对在高速传感数据采集领域中 RDMA 软件栈面临的问题挑战,满足无人驾驶、工业物联网等嵌入式场景下的传感数据高性能传输需求,本文以高速传感数据采集场景为导向,基于 RISC-V 微核处理器设计基于可靠 UDP 的轻量化建链和基于可靠以太网的轻量化 RDMA 软件驱动,并实现了不依赖于高性能 CPU 和操作系统的超轻量化 RDMA 软件栈解决方案,为构建软硬件协同的嵌入式 RDMA 端系统提供软件层面技术支撑,并为无人驾驶等智能边缘设备的高性能传感数据传输提供新的思路和解决方法。

1 RDMA 标准软件栈

1.1 RDMA 传输

RDMA 通过网络协议栈的硬件卸载实现，允许网络设备在不经过双端 CPU 参与的情况下直接访问另一个网络设备的内存空间，从而在显著降低延迟和 CPU 开销的同时提高传输带宽。在实现形式上，RDMA 协议实现方式主要包括 InifiniBand[18]，iWARP[19]，RoCE[20] 三类。其中，InifiniBand 采用全自定义网络协议栈，需专用硬件网卡和交换机作为硬件支撑，具备成本极高、性能极佳、不兼容以太网的通信特点，常用于超算领域；iWARP 实现了以太网兼容，并通过 TCP/IP 协议栈保证传输可靠性，但 TCP/IP 协议栈的硬件卸载实现在成本和复杂度上仍然较高，在数据中心等领域尚未大规模应用；RoCEv1 同样进行了以太网兼容性设计，RoCEv2 则进一步使用 UDP/IP 作为网络层和传输层协议从而支持跨子网通信，并采用流量控制和拥塞控制等方式保证传输可靠性，目前凭借较低的实现复杂度和较好的以太网兼容性已经在数据中心领域得到了广泛关注。

RDMA 将网络协议栈相关报文处理以及可靠传输等功能卸载到网卡硬件实现，用户态 RDMA 驱动程序直接将 RDMA 操作下发至硬件网卡以启动 RDMA 传输，实现内核旁路，无需操作系统内核态参与。启动传输后由 RDMA 硬件网卡直接进行远端内存读写，无需进行用户态到内核态的多次拷贝和系统调用。从通信原语上，RDMA 为应用程序提供 RDMA WRITE/READ 等单边原语以及 SEND/RECV 双边原语。其中，单边原语在传输过程中不需要通信双方 CPU 参与，常用于数据搬移；双边原语则涉及对端 CPU 参与下发 RECV 操作并处理对应操作完成信息，常用于通信双方状态信息同步。从传输方式上，RDMA 提供可靠连接（reliable connected, RC）、不可靠连接（unreliable connected, UC）、不可靠数据报（unreliable datagram, UD）等多种传输方式。其中，RC 传输方式需在通信双方之间建立一对一连接并提供可靠传输服务，最大消息长度可达 2 GB；UD 传输方式则支持一对多通信，不需要保证传输可靠性且仅支持双边原语，最大消息长度仅 4 KB。

1.2 RDMA 软件栈工作流程

标准 RDMA 软件栈包括 RDMA 双端建链协议栈、RDMA 用户态和内核态驱动，整体工作流程如图 1 所示，主要包括前期配置、数据传输、资源回收 3 部分[21]。

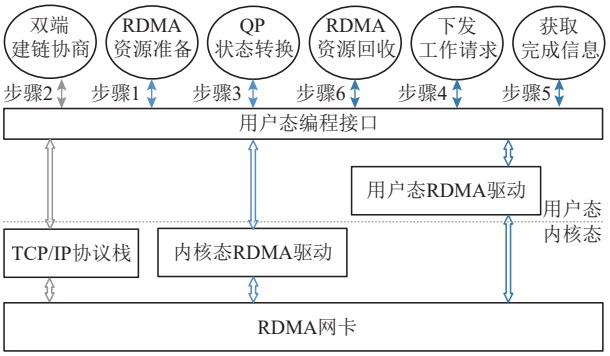


Fig. 1 Standard RDMA software stack

图 1 标准 RDMA 软件栈

前期配置阶段主要完成 RDMA 资源准备、双端建链协商以及队列对（queue pair, QP）状态转换，由 RDMA 内核态驱动参与完成。其中，资源准备主要涉及 RDMA 应用程序打开 RDMA 硬件网卡，并创建用于网卡进行内存数据读写访问的内存区域（memory region, MR）、CQ、QP 等资源。通信双端建链协商主要包括基于通信管理器（communication manager, CM）和基于 Socket TCP/IP 协议栈 2 种方式，其中 CM 建链需要一个 UD 类型 QP 专用于处理 CM 相关报文，且具备专为 CM 建链定制的编程接口，整体实现较为复杂。相比之下，Socket 建链方式只需按照标准 Socket API 进行通信即可，更适合面向嵌入式场景进一步优化，基于 Socket 协议栈提供的网络通信接口进行双端建链交互，调用 RDMA 驱动程序提供的 verbs 接口完成 RDMA 控制路径相关操作。具体 Socket 建链交互信息可见 2.3 节。建链完成后，通信双方已经得知与对端进行 RDMA 操作对应使用的 QP 相关信息，并进行 QP 状态转换，正常情况下 QP 在创建后依次从复位（reset, RST）状态转换至初始化（initialized, INIT）状态、准备接收（ready to receive, RTR）状态和准备发送（ready to send, RTS）状态。

数据传输阶段主要包括软件下发工作请求和获取完成信息，以及硬件网卡执行数据传输等环节。在软件层面，RDMA 应用程序首先向 QP 中的接收队列（receive queue, RQ）下发操作请求，预先为接收通信对端发来的 SEND 操作做好准备，之后可根据需求向 QP 中的发送队列（send queue, SQ）下发 WRITE 和 READ 等操作，并由硬件网卡根据请求进行报文封装和解析等操作。具体而言，应用程序调用相关函数后，相关工作请求（work request, WR）会被 RDMA 用

户态驱动程序解析为工作队列元素(work queue element, WQE)放入相应 SQ 或 CQ 中,并通过 Doorbell 机制通知硬件网卡有待处理的 WQE。网卡根据 Doorbell 机制和对应 WQE 解析需完成的 RDMA 操作请求构造 RDMA 请求报文并接收对端网卡响应;在报文封装和解析过程中进行远端内存读写,并在本次操作执行完成后构造完成队列元素(completion queue element, CQE)放入对应 CQ。应用程序可通过轮询或中断等方式获取 CQ 中的操作。

在资源回收阶段,RDMA 操作已全部执行完成,RDMA 应用程序依次销毁 QP, CQ, MR 等内存资源,并关闭 RDMA 硬件网卡设备,避免产生内存泄漏等问题。此过程同样需要 RDMA 内核态驱动程序参与。

2 面向高速传感数据采集的轻量化嵌入式 RDMA 软件栈

2.1 轻量化嵌入式 RDMA 软件栈设计目标

在高速传感数据采集领域中应用 RDMA 技术时,需重点考虑其嵌入式场景特点:首先,嵌入式系统受限于算力和存储资源,如采用低功耗处理器 ARM Cortex-M 系列等,难以直接运行标准 RDMA 软件栈;其次,面向高速传感数据采集的嵌入式系统,功能相对单一,无需支持完备的 RDMA 通信原语和传输方式;最后,嵌入式系统多运行轻量级实时操作系统,缺乏通用操作系统的多任务调度和虚拟内存管理等高级功能。鉴于上述嵌入式系统场景特点,进行完整的 RDMA 驱动程序和建链协议栈的直接移植难以满足场景需求且并不必要[14]。

因此,需面向高速传感数据采集的嵌入式场景实现轻量化 RDMA 软件栈,并满足以下设计目标:1)保证 RDMA 标准兼容性。保证 RDMA 标准兼容性一方面可以保证嵌入式端与商用 RDMA 网卡的互操作性,有利于高速数据采集嵌入式端直接接入现有 RDMA 通信网络;另一方面可以实现嵌入式 RDMA 软件栈对硬件标准 RDMA 网卡中的典型处理逻辑,如 RDMA 报文封装解析以及可靠传输保证等的兼容配置。2)支持硬件灵活配置。在高速传感数据采集场景中,不同类型传感设备往往具备不同的流量特征,如温湿度传感器等可能进行周期性数据采集并将其打包为独立的数据单位进行块式传输,摄像头则进行音视频流数据的持续采集并将其以连续字节流的形式进行流式传输。即便同类型传感设备,以摄像头为例,不同分辨率对应的帧图像大小均有不同,这需要嵌入式 RDMA 软件栈实现灵活的硬件 RDMA 网卡处理逻辑传输配置。3)避免嵌入式端弱核影响传输性能。在 1.2 节标准 RDMA 工作流程中,每次 RDMA 传输前的操作启动和传输后的完成处理均需要软件栈参与,在 RDMA 操作频繁的情况下,尤其是对流式传感数据传输,弱核参与会显著影响全系统传输性能,这需要嵌入式 RDMA 软件栈进行合理的软件功能卸载以提升系统整体传输效率。

2.2 轻量化嵌入式 RDMA 软件栈总体架构

标准 RDMA 软件栈实现依赖于操作系统,并需要 RDMA 应用程序、RDMA 用户态驱动、内核态驱动和用于建链的 Socket TCP/IP 协议栈协同工作。轻量化嵌入式 RDMA 软件栈不依赖操作系统实现,无需区分用户态和内核态,从而实现嵌入式 RDMA 软件栈的轻量化设计。

嵌入式 RDMA 软件栈整体功能架构包括 4 个主要模块:1)地址解析协议(address resolution protocol, ARP)请求响应模块。嵌入式 RDMA 端作为一个网络通信设备,应支持 ARP 协议以便处理 ARP 请求/响应,方便网络设备间地址解析与通信。2)存储空间管理模块。嵌入式 RDMA 软件栈需要在前期配置阶段创建 MR, CQ, QP 资源,并进行相关内存空间地址信息维护。由于嵌入式 RDMA 软件栈相关存储资源均使用物理地址,因此无需维护内存转换表(memory translation table, MTT)表项信息。3)通信双端建链交互模块。该模块主要负责定义建链结构体(包括通信对端内存地址等信息),完成 UDP 报文的封装与解析,并实现丢包重传处理,详见 2.3 节。4)RDMA 嵌入式端硬件配置模块。该模块主要负责嵌入式端 RDMA 硬件网卡处理逻辑所需元数据信息配置以及 RDMA 传输参数配置,详见 2.4 节。

嵌入式 RDMA 软件栈的工作流程为:在系统上电后首先对所采用的 RISC-V 微核进行初始化配置,由 ARP 请求响应模块构造 ARP 请求并处理响应,更新 ARP 地址解析信息。由存储空间管理模块进行 MR 等地址信息初始化,并由硬件 RDMA 配置模块通过自定义以太网报文对硬件 RDMA 处理逻辑正常通信所需 QPC(QP context)、CQC(CQ context)、SGID(RDMA 网络节点标识符)、SMAC(源 MAC 地址)等元数据信息进行配置。随后,由通信双端建链交互模块根据存储空间管理模块相关内存信息进行本端建链结构体初始化,并据此构造 UDP 报文进行双端通信,建链交互完成后由 RDMA 嵌入式端硬件配置模块进行 QPC 配置并实现 QP 状态转换。此外,

RDMA 嵌入式端硬件配置模块还会进行 WQE 携带数据量等传输参数配置。

相较于标准 RDMA 软件栈，嵌入式 RDMA 软件栈直接从嵌入式场景下的硬件配置需求出发进行设计，无需操作系统参与，省去了系统调用、虚实地址转换、资源申请注册及回收操作，并通过轻量化 UDP 建链、硬件卸载 RDMA 传输请求下发和传输完成处理进一步实现了嵌入式 RDMA 软件栈轻量化。在此对嵌入式 RDMA 软件栈未设计专门的资源申请注册即回收机制的合理性进行说明：1）高速传感数据采集应用场景固定，如自动驾驶传感数据采集设备往往只更新不移除；2）资源管理本身不依赖于操作系统，无需操作系统相关的资源申请和注册、管理等操作；3）传感数据采集设备移除时直接更新 QP 配置即可实现资源重新分配，无需复杂注册回收流程。RDMA 轻量化软件栈总体架构如图 2 所示。

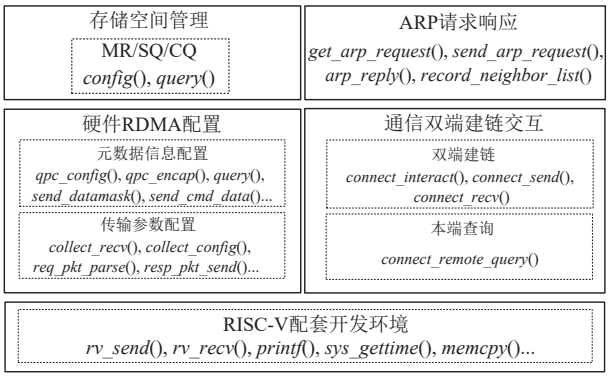


Fig. 2 Overall architecture of embedded RDMA software stack

图 2 嵌入式 RDMA 软件栈总体架构

2.3 基于可靠 UDP 的轻量化 RDMA 建链

RDMA 协议标准并未规定双端进行信息交互的建链方式或建链协议栈，本文针对高速传感数据采集嵌入式场景，提出一种基于可靠 UDP 的轻量化 RDMA 建链方案，以尽可能实现建链轻量化。从可选建链方式来看，TCP 卸载引擎虽能卸载 TCP/IP 协议栈，但因协议复杂性导致硬件成本过高；RDMA CM 建链依赖完整 RDMA 协议栈，与轻量化设计背道而驰；UDP 卸载引擎虽硬件复杂度低、算力占用少，适配嵌入式场景，但原生 UDP 协议不可靠，无法保障关键控制信息传输。对此，基于可靠 UDP 的轻量化 RDMA 建链可在不依赖于协议栈的情况下直接进行 UDP 报文构造和解析，并通过扩展序列号字段校验和超时重传保证传输可靠性。该方式针对 RDMA 建链这一特定需求进一步轻量化，避免了建链协议栈带来的系统资源开销。

如图 3 所示为 RDMA 轻量化软件栈中建链交互内容和交互报文格式。为实现 RDMA 协议标准兼容，建链交互内容包括：1）网络通信相关信息，即 SGID 和本地标识符（local identifier, LID），分别用于支持 RDMA 通信设备跨网和子网内通信。2）注册 MR 相关信息，即虚拟地址（virtual address, VA）和远程访问密钥（remote key, RKey），方便交互节点凭借 Rkey 校验权限和 VA 地址信息直接进行远端内存访问。3）QP 相关信息，即 QP 编号（QP number, QPN），在每个 RDMA 设备创建多个 QP 的情况下，QPN 交互对于通信双方建立连接和正确进行数据包处理是必要的。此外，在 RC 传输方式下，通信双方交互分组序列号（packet sequence number, PSN），用于检测报文是否按序接收进而进行可靠传输保证。在双端交互报文格式上，基于可靠 UDP 的 RDMA 建链遵循 UDP/IP 标准协议格式，帧类型、首部长度和上层协议等字段可直接指定，IP 首部校验和、UDP 首部校验和这 2 个字段根据校验算法进行填充。此外，为保证传输可靠性，在 UDP 报文首部基础上增加 2B 扩展序列号字段，该字段根据当前系统时间随机生成，并根据响应报文回送序列号进行可靠传输校验。

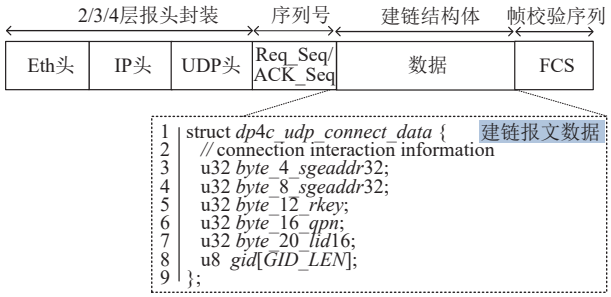


Fig. 3 Connection establishment interaction content and message format

图 3 建链交互内容及报文格式

在具体实现上，建链交互模块主要通过函数 *connect_send*, *connect_recv*, *connect_interact* 实现双端通信功能并为本端其他模块提供 *connect_remote_query* 查询接口。函数 *connect_send* 负责 UDP 建链请求报文各字段配置、校验和计算、序列号生成，并通过 RISC-V 处理器提供的底层报文发送函数 *rv_send* 进行报文发送。函数 *connect_recv* 则负责通过函数 *rv_recv* 进行报文接收和识别，对收到的 UDP 建链响应报文进行校验检查，并根据校验结果解析报文建链结构体信息或返回错误信息。函数 *connect_interact* 则作为建链交互模块对外提供的调用接口，根据主函数提供的本端建链结构体信息和 QPN 调用

函数 *connect_send* 发送报文, 并循环调用函数 *connect_recv* 进行响应报文接收和校验, 并在校验错误或超时未收到响应时重传报文。如图 4 所示为基于可靠 UDP 的轻量化 RDMA 建链工作流程示意图。

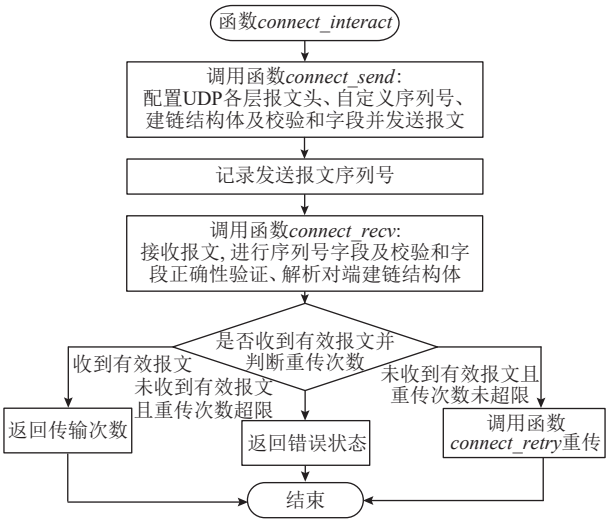


Fig. 4 Connection establishment interaction workflow

图 4 建链交互工作流程

基于可靠 UDP 的轻量化 RDMA 建链只需在商用 RDMA 网卡的上层软件中将 RDMA 应用程序中 TCP Socket 函数替换为 UDP Socket 函数, 并在 UDP 载荷中增加序列号字段即可, 无需修改 RDMA 硬件网卡逻辑或 RDMA 标准报文交互流程, 因此完全兼容标准 RDMA 协议, 实现嵌入式端与接入商用 RDMA 网卡标准端交互。

2.4 基于可靠以太网的轻量化 RDMA 软件驱动

RDMA 内核态驱动完成 RDMA 工作流程前期配置阶段的资源准备和 QP 状态转换部分以及资源回收阶段, RDMA 用户态驱动程序则主要完成数据传输阶段的操作请求下发和完成通知处理部分。本文提出一种针对高速传感数据采集嵌入式场景, 基于可靠以太网的轻量化 RDMA 软件驱动, 包括轻量化 RDMA 元数据信息配置方法以实现前期配置阶段的资源准备和 QP 状态转换部分、轻量化 RDMA 传输参数配置以及硬件卸载机制以实现数据传输阶段的操作请求下发和完成通知处理部分, 从而避免了将标准 RDMA 驱动程序直接进行嵌入式移植存在的实现复杂度高、功能冗余、资源占用高等方面问题。

2.4.1 轻量化 RDMA 元数据信息配置

轻量化 RDMA 元数据信息配置主要用于将 QP, CQ, MR 等资源相关参数配置给嵌入式端 RDMA 硬件网卡处理逻辑, 以保证网卡进行资源访问和 RDMA 报文处理。由于仅涉及嵌入式端软硬件间通信, 因此在配置方式上, 轻量化 RDMA 元数据信息配置无需考虑双端 RDMA 通信兼容性问题, 采用自定义网络附加控制协议(network attached control protocol, NACP)和 Mlbox 交互机制联合实现。NACP 协议遵循以太网报文格式, 并定义序列号字段和元数据信息配置命令字段作为以太网报文有效载荷。如图 5 所示, 其中, 序列号字段与 2.3 节中 UDP 载荷扩展字段作用一致, 用于保证 NACP 传输可靠性。元数据信息配置命令字段根据配置地址、配置数据量等信息进行了字段定义, 使用字段 *Type* 区分配置硬件元数据信息的请求/响应报文和查询硬件元数据信息的请求/响应报文, 使用字段 *Burst*/*Tag* 说明配置或查询数据字节数, 使用字段 *Addr* 区分配置或查询地址。轻量化 RDMA 软件驱动提供的配置请求报文和网卡硬件的响应报文携带配置和查询的元数据信息。在交互机制上, 元数据信息配置涉及 QPC、CQC、SGID、SMAC、内存保护表(memory protection table, MPT)用于校验内存访问权限等多项配置, Mlbox 交互机制用于传输元数据配置相关说明信息。

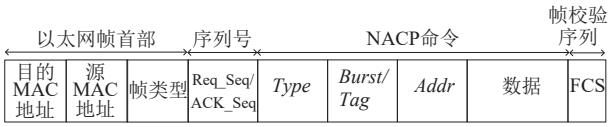


Fig. 5 NACP message format

图 5 NACP 报文格式

如图 6 所示, 以 QPC 配置为例, 软硬件通过 NACP 配置请求/响应报文首先进行 QPC 元数据信息的交互, 之后进行 QPC 配置掩码信息交互, 最后进行 QPC 配置来描述信息 ***cmd_data*** 的信息交互, 获取当前硬件是否处于可配置状态, 并根据查询状态决定继续查询或通过 ***cmd_data*** 完成本次元数据信息配置。

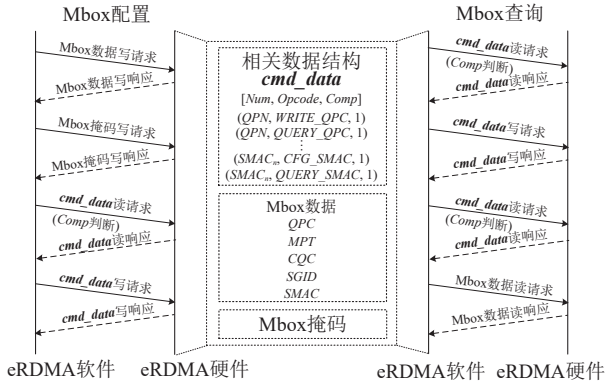


Fig. 6 Mbox interaction mechanism

图 6 Mbox 交互机制

上述 ***cmd_data*** 主要包括用于指定本次配置或查询对应的 QPN/CQN 等的目标编号 *Num*，用于区分具体元数据信息类型的操作码 *Opcode* 以及用于指示当前硬件配置状态的状态标识位 *Comp*。

在具体实现上，不同元数据信息配置流程基本一致。以需要实现 QP 状态转换且配置流程较为复杂的 QPC 配置为例，元数据信息配置主要包括 QPC 数据结构配置、NACP 命令构造、NACP 报文收发处理 3 部分。QPC 数据结构配置包括用于 QP 状态转换的 *qpc_config_INIT_RTR*, *qpc_config_RTR_RTS* 等函数，这些函数会进一步调用函数 *qpc_config* 进行 QPC、QPC 掩码以及 *cmd_data* 结构体赋值。NACP 命令构造包括作为外部函数调用接口以及用于 NACP 命令构造的函数 *qpc_encap*，用于掩码和 *cmd_data* 信息 NACP 命令构造的 *send_datamask*, *read_cmd_data*, *send_cmd_data* 等函数，这些函数除了进行对应 NACP 命令构造外，还会调用底层 NACP 报文收发接口进行报文收发。NACP 报文收发处理使用函数 *nacp_write_send* 作为外部调用接口，该函数调用请求报文封装函数进行 NACP 写请求报文封装，调用响应报文解析函数进行 NACP 写响应报文处理，并在响应报文超时或校验失败时进行请求报文重传以保证配置可靠性。

2.4.2　轻量化 RDMA 传输参数配置机制

标准 RDMA 用户态驱动需要参与传输启动和传输完成处理，为避免传感端的弱核处理器参与数据传输过程而影响 RDMA 传输性能，本文将上述 RDMA 用户态驱动参与的相关操作卸载到嵌入式端 RDMA 硬件网卡处理逻辑中实现，同时为嵌入式 RDMA 软件栈提供传输参数配置接口，兼顾传输性能和灵活性。

轻量化 RDMA 传输参数配置，主要用于传感端数据传输相关系数设置，包括 QPN 和 WQE 传输数据大小、传感端数据存储的 DDR（double data rate SDRAM）首地址与长度、传输启停状态等。传输参数配置同样使用 NACP 报文格式，通过以太网帧类型区分传输参数配置报文和上述元数据配置报文，配置内容则主要取决于高速传感数据采集端的流量特性。值得注意的是，嵌入式 RDMA 软件栈在收到通信对端的 UDP 采集控制报文后才可进行硬件传输启停参数配置。

在具体实现上，传输参数配置与上述元数据配置流程类似，调用 NACP 命令构造以及 NACP 报文交互相关函数即可。传输启停状态配置涉及用于通信对端进行 UDP 报文交互的相关函数，其中函数 *collect_recv* 用于对收到的 UDP 传输启停控制报文进行识别和校验，函数 *req_pkt_parse* 用于解析控制报文携带的传输配置信息，函数 *resp_pkt_send* 用于构造对应 UDP 传输控制响应报文，函数 *collect_config* 用于进行硬件传输启停状态配置。传感端卸载标准 RDMA 用户态驱动，由网卡硬件完成 RDMA 传输启动和完成处理操作，避免传感端弱核处理器参与数据传输过程而影响 RDMA 传输性能，嵌入式端网卡根据软件配置的传输参数进行 WQE 构造、SQ Doorbell 生成、CQE 解析以及 CQ Doorbell 下发。

轻量化 RDMA 元数据配置和传输参数配置，基于嵌入式传感端软硬件交互需求，通过对高速传感数据采集场景的轻量化软件栈设计，实现了与标准 RDMA 驱动程序一致的底层功能，在最小化软件开销的前提下，实现 RDMA 传输协议，并在支持硬件灵活配置的同时规避了传感端弱算力对传输性能的影响。

3　实验测试与评估

3.1　实验环境

实验物理环境如图 7 所示，实验平台主要包括用于运行嵌入式 RDMA 轻量化软件栈的 DP4C 开发板[22]、用于模拟采集传感数据生成和接入的 PC 主机和 ALINX FH1219 适配模块、用于运行硬件传输相关卸载模块和 RDMA 处理逻辑的 VCU118 开发板[23]，以及由飞腾 D2000 CPU、华为 Atlas 300I Pro 推理卡和 NVIDIA Mellanox ConnectX-5 网卡构成的 RDMA 通信对端。其中 VCU118 开发板作为嵌入式端 RDMA 网卡硬件逻辑载体，将 DP4C 下发的建链交互报文和 UDP 传输启停控制报文转发至 RDMA 通信对端，将 DP4C 相关元数据配置报文和传输参数配置报文解析后，进行嵌入式端 RDMA 网卡硬件模块配置，实现 RDMA 请求响应报文的收发处理。RDMA 通信对

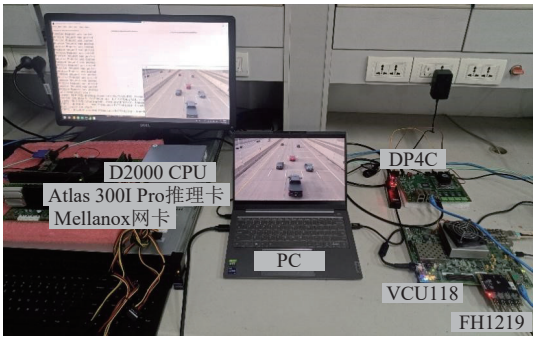


Fig. 7　Physical environment of experimental platform

图 7　实验平台物理环境

端由飞腾 D2000 支持标准 RDMA 驱动程序和操作系统运行以及与嵌入式端的建链和传输控制交互，以及由商用 Mellanox 网卡负责与嵌入式端的 RDMA 通信，验证嵌入式 RDMA 软件栈的功能正确性和标准 RDMA 协议兼容性，由华为 Atlas 300I Pro 推理卡完成从嵌入式端传输来的视频流的实时目标检测和分析处理。通信双端通过 100 Gbps 光纤互联，理论带宽最大可支持 100 Gbps。

在此对承载嵌入式 RDMA 软件栈的 DP4C 开发板做进一步说明。DP4C 开发板使用自研 RISC-V 芯片，该芯片集成 3 个通用 RISC-V 核处理引擎以及 1 个 AI 处理引擎，并提供配套软件开发环境，如上文所述函数 *rv_send*, *rv_recv* 以及系统打印和字符串处理等功能函数。嵌入式 RDMA 轻量化软件栈在 Linux 平台经 C 语言编程开发后，通过配套 RISC-V 交叉编译链和 DP4C 配置程序部署至 DP4C 开发板运行。

3.2 资源开销评估

由于嵌入式相关软件代码部署在 RISC-V 芯片微核上执行，无法直接与标准 RDMA 软件栈对比 CPU 利用率或内存占用率，因此本文从芯片占用面积、功耗和存储资源 3 个方面对其轻量化实现后的资源开销进行评估。

在芯片占用面积方面，DP4C 开发板中 RISC-V 芯片基于 130 nm 工艺流片，芯片总面积 100 mm^2，4 个处理引擎总面积约 37 mm^2，嵌入式 RDMA 软件栈只使用其中一个 RISC-V 核处理引擎，实际执行占用面积仅约 9 mm^2。在功耗方面，DP4C 芯片整体功耗约 1.4 W，嵌入式 RDMA 软件栈单个 RISC-V 核运行所占功耗远低于此，可以保证低功耗运行。

如图 8 所示，在存储资源方面，对于通信双端建链，使用 cloc 代码统计工具对 Linux6.15 版本下的标准 Socket TCP/IP 建链协议栈代码量进行了统计，标准 TCP 建链协议栈内核态代码约 24 148 行，主要包括 TCP 连接管理\状态机\拥塞控制等核心程序，用户态 Socket 封装代码约 2 552 行。相比于标准 TCP 建链协议栈，基于无操作系统可靠 UDP 的轻量化 RDMA 建链协议(可看作用户态代码)代码总量仅约 1 371 行，可降低约 94% 的建链开销。如图 9 所示，标准 RDMA 用户态驱动程序代码约 12 909 行，内核态驱动程序代码约 3 320 行，而基于可靠以太网的轻量化 RDMA 软件驱动代码总量仅约 2 503 行，可降低约 84% 的软件驱动开销。上述结果表明，本文设计的嵌入式 RDMA 软件栈可大幅降低 RDMA 软件栈所需存储空间和计算资源，并完整支持通信双端 RDMA 传输，达到了较好的轻量化效果，满足嵌入式应用场景特性。

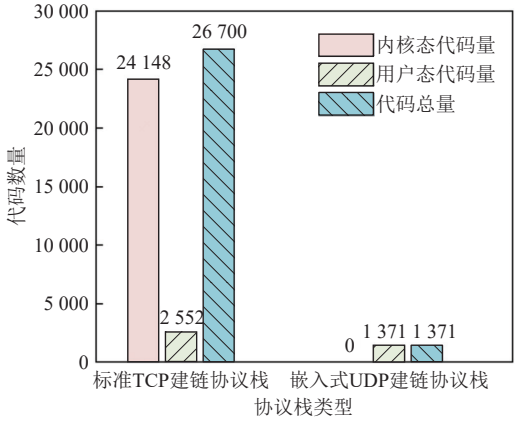


Fig. 8 Comparison of protocol stack code size

图 8 协议栈代码量对比

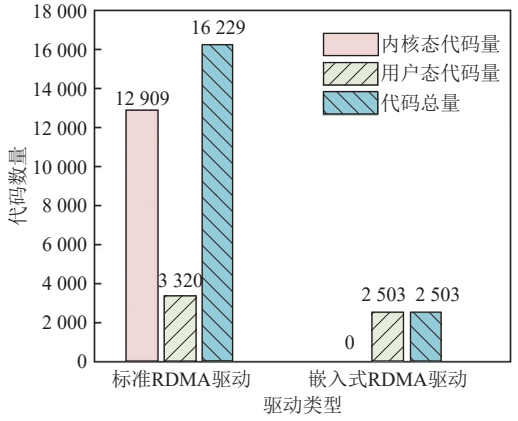


Fig. 9 Comparison of driver code size

图 9 驱动代码量对比

以上主要对嵌入式端 RDMA 轻量化软件栈资源开销进行评估，为对比嵌入式 RDMA 轻量级可靠 UDP 建链和标准 RDMA 基于 socket TCP 建链的 CPU 开销，分别搭建 2 个环境。环境 1: 请求端使用轻量化 RDMA 网卡，响应端为 Mellanox 商用百 G 网卡，双端通过轻量级 udp 建链。环境 2: 请求端使用 Mellanox 商用百 G 网卡，响应端同样为 Mellanox 商用百 G 网卡，双端通过 socket TCP 完成建链。为了更客观地统计双端 CPU 负载情况，我们将运行在请求端轻量化 RDMA 的 RISC-V 微核上的建链程序，代理运行在响应端商用 CPU 上，通过远程配置请求端，在同一 CPU 上统计负载占用率。环境 1 中，请求端轻量化 RDMA 建链 CPU 负载占用率为 22.8%，响应端的 CPU 负载为 11.8%。环境 2 中，建链请求端的 CPU 负载占用率为 54.0%，响应端的 CPU 负载为 26.9%。

测试结果如表 1 所示，轻量化 RDMA +轻量级可

Table 1 Comparison of CPU Load Between Reliable UDP Connection Establishment and Socket TCP Connection Establishment

表 1 可靠UDP建链与Socket TCP建链的CPU负载对比 %

测试场景	请求端 CPU 负载	响应端 CPU 负载
轻量化 RDMA+轻量级可靠 UDP 的建链	22.8	11.8
标准 RDMA+Socket TCP 的建链	54.0	26.9

靠 UDP 的建链相较于标准 RDMA+ Socket TCP 建链，能够显著降低 CPU 负载。

为了进一步验证轻量化 RDMA 应用于高速传感数据采集场景带来的低网络处理开销优势，本节在不同带宽模拟视频流输入情况下，对通信对端的 CPU 利用率进行评估，实验结果如图 10 所示，通信对端的 CPU 利用率相关开销稳定在 4.2%~5.6% 之间，主要用于 CQE 轮询以确保视频流实时分析处理。对于块式传输数据而言，通信对端的 CPU 利用率约占 0%。相比于基于 Socket TCP 的数据传输（使用 iPerf 在相同消息大小和 MTU 为 4 096 B 的配置下进行测试），基于嵌入式 RDMA 软件栈的高速数据采集系统在保证低 CPU 网络处理开销方面优势显著，能够满足嵌入式应用场景需求。

3.3 传输性能评估

本节对嵌入式 RDMA 轻量化软件栈配置下的系统传输性能进行测试。3.2 节中，采用 PC 模拟视频采集数据，通过 SDI 接口接入传感端的功能测试环境，系统传输性能受限于输入的 PC 模拟采集数据生成的视频流带宽，不能测试出系统传输性能上限。因此，本节性能测试环境将模拟传感数据首先存入嵌入式端 DDR，通过嵌入式 RDMA 将存入传感端 DDR 的视频流数据与对端进行 RDMA 通信来完成传输性能测试，重点评估嵌入式 RDMA 软件栈工作条件下的双端 RDMA 实际传输性能，主要分为传输带宽性能评估和传输延迟测量。为验证系统传输能力，在传感终端和标准算力终端之间接入交换机，支持单接口 100 Gbps 交换能力，以下带宽及延迟均基于下一跳交换进行。

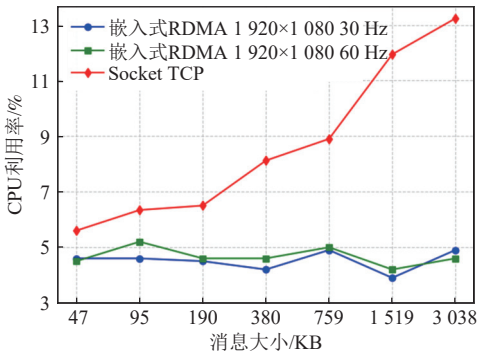


Fig. 10 Standard host CPU utilization

图 10 标准端 CPU 利用率

传输带宽性能评估旨在研究面向高速传感数据采集的嵌入式 RDMA 软件栈是否可满足多源传感数据的高带宽、高实时传输需求。实验使用 perftest 设置不同消息大小，并分别选取 MTU 为 512 B/1 024 B/2 048 B/4 096 B 及 QP 数量为 1/10/30 的情况下对其传输延迟进行测试，结果如图 11 所示。分析表明，系统吞吐量随 QP 数量、MTU 尺寸及消息大小的增加而显著提升。在系统时钟频率为 250 MHz 的条件下，当 QP 数量为 30、MTU 设置为 4 096 B、消息大小约为 1 MB 时，最大单向吞吐量达到 98.04 Gbps。这一结果验证了嵌入式 RDMA 轻量化软件栈在并发连接场景下具有良好的传输性能和扩展性，能够有效满足高带宽传感数据传输需求。我们同时注意到单 QP 条件下的峰值性能已逼近多 QP 场景下的性能，这在高速数据采集应用中具有十分重要的意义，因为嵌入式 RDMA 数据采集端通常使用较少的 QP 对进行海量传感数据融合传输。

如图 12 所示，标准 RDMA 协议栈在不同 QP 数量和 MTU 配置下的峰值传输带宽表现出与嵌入式 RDMA 协议栈相似的变化趋势。然而在 QP 数量为 1 时，标准 RDMA 协议栈 WRITE 与 READ 操作的峰值带宽仅有 54.91 Gbps 和 34.70 Gbps，比嵌入式 RDMA

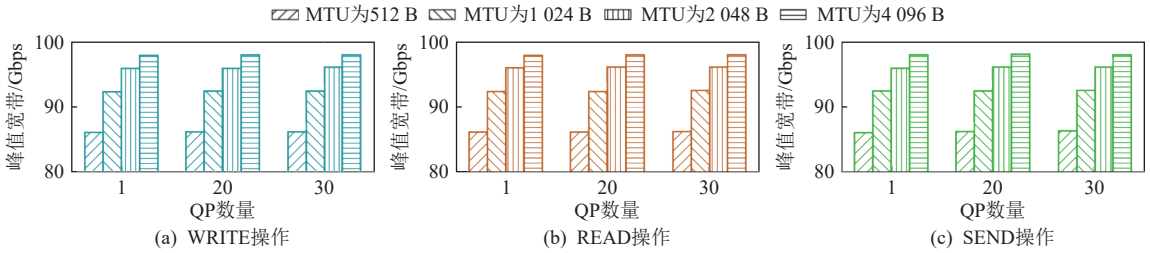


Fig. 11 Peak transmission bandwidth of the embedded RDMA under different QP counts and MTU sizes

图 11 不同 QP 数量和 MTU 大小下的嵌入式 RDMA 传输带宽峰值

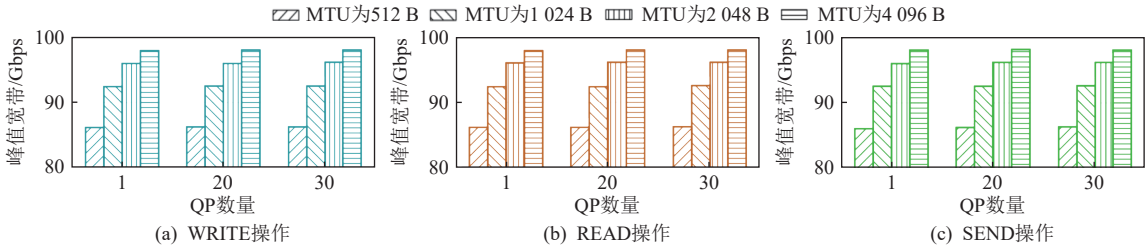


Fig. 12 Peak transmission bandwidth of the standard RDMA under different QP counts and MTU sizes

图 12 不同 QP 数量和 MTU 大小下的标准 RDMA 传输带宽峰值

协议栈分别低 52% 和 148%，充分说明嵌入式 RDMA 协议栈在高速数据采集应用中的优越性。

传输延迟测试用于验证面向高速传感数据采集的嵌入式 RDMA 软件栈是否能够满足高速传感数据的低延迟传输需求。测试条件同上，分别选择较小消息(256 B)以测试端到端 WRITE，READ，SEND，RECV 传输的最低延迟。在此说明，perftest 测试所得传输延迟为端到端总延迟，包含终端应用下发 WQE 与 Doorbell、RDMA 报文处理逻辑解析 WQE 并构造报文、链路传输、接收全部预期 ACK、RDMA 报文处理逻辑生成 CQE，以及终端应用接收并完成 CQE 处理的全流程耗时。如图 13 所示，传输数据量(消息大小)是影响端到端延迟的关键因素，且在系统时钟频率 250 MHz 的配置下，当采用 QP 数量为 1、MTU 设置为 2 048 B、消息大小约为 256 B 时，过一级交换，WRITE 操作的端到端传输延迟最低仅为 4.58 μs，其中单跳交换延迟约 2 μs。这一结果验证了算控网络在满足高带宽传感数据接入的同时，能够有效满足高速传感数据的低延迟实时传输需求。

如图 14 所示，标准 RDMA 协议栈在不同 MTU 大小下的延迟表现与嵌入式实现基本相当，然而标准 RDMA 协议栈在不同操作类型下随 MTU 尺寸变化的延迟表现存在显著差异，在较小 MTU 配置下，READ 操作的延迟明显高于 WRITE 与 SEND/RECV 操作，这是因为 READ 操作需要发起端首先发送 READ 请求报文至对端，等待对端返回数据报文，此双向交互过程在 MTU 较小时会产生更多的报文数量与更频繁的上下文切换，显著增加了端到端延迟。相比之下，嵌入式 RDMA 中 MTU 对于延迟的影响相对较小。

为验证嵌入式 RDMA 端与商用通信对端的协议兼容性，我们分别测试了二者的 WRITE 操作吞吐量和 READ 操作吞吐量随消息大小和 QP 数量的变化趋势，我们将 MTU 固定为 4 096 KB 以获取峰值带宽。如图 15 所示，随着 QP 数量和数据量增加，吞吐量稳定提高，在足够数量的 QP 以及大数据量的情况下，嵌入式 RDMA 端的 WRITE 和 READ 性能均能够达到商用通信对端峰值性能(98 Gbps 及以上)。单 QP 条件下，嵌入式 RDMA 端发起对商用通信对端的 READ 请求，带宽提升速率明显低于商用通信对端对嵌入式 RDMA 端的 READ 带宽提升速率，这是因为嵌入式 RDMA 中，传输状态及上下文信息均保存在网卡本地，减少了因访问 CPU DDR 内存带来的延迟，因此当嵌入式 RDMA 被读时，性能更优。

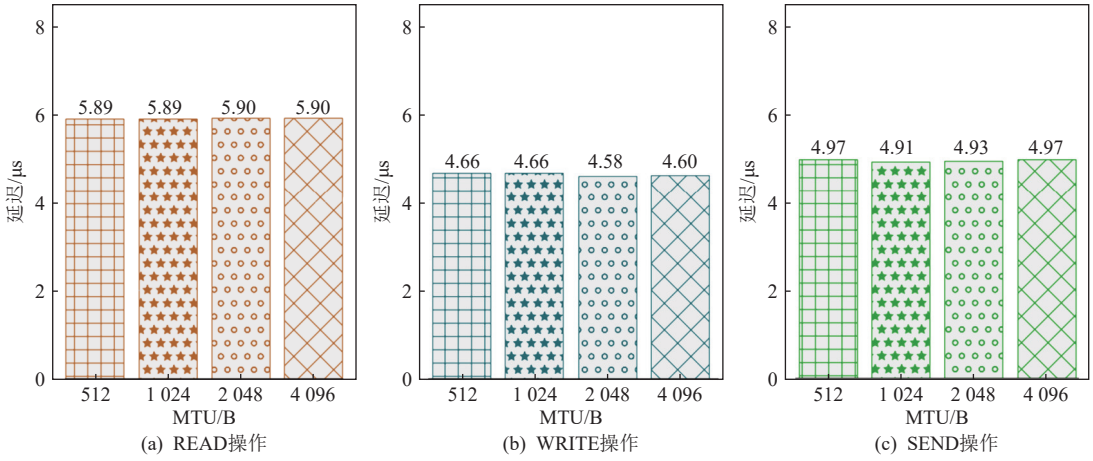


Fig. 13 Transmission latency of the embedded RDMA under different MTU sizes

图 13 不同 MTU 大小下的嵌入式 RDMA 传输延迟

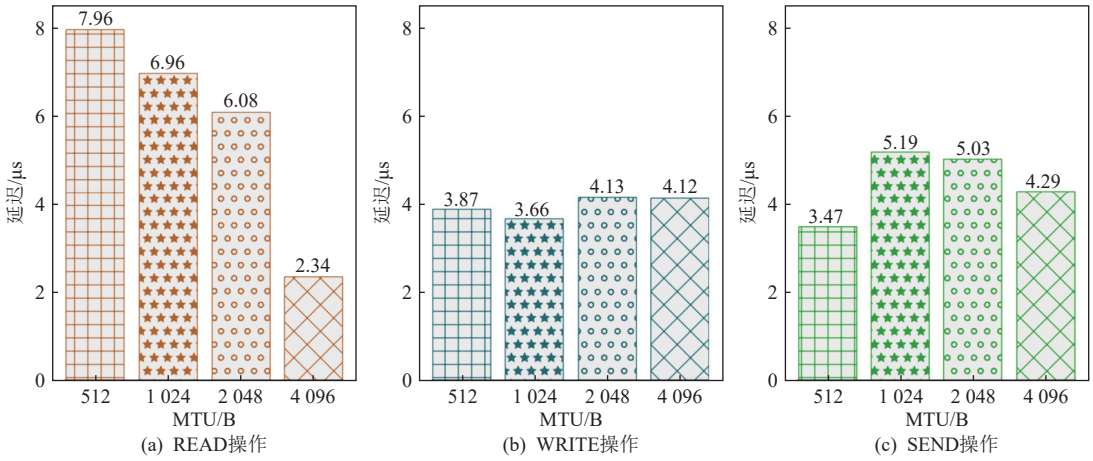


Fig. 14 Transmission latency of the standard RDMA under different MTU sizes

图 14 不同 MTU 大小下的标准 RDMA 传输延迟

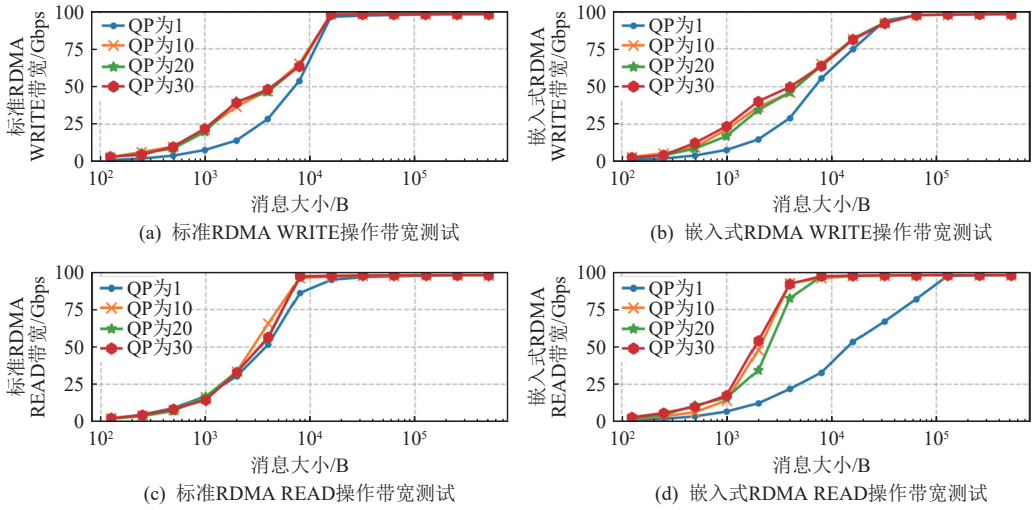


Fig. 15 Variation trend of system transmission bandwidth with QP counts and message sizes

图 15 系统传输带宽随 QP 数量和消息大小的变化趋势

嵌入式 RDMA 端与商用通信对端的延迟趋势测试是在 QP 为 1 的条件下进行的。如图 16 所示，当传输数据量较小时（≤10^4 KB），二者的 WRITE 和 READ 延迟始终维持在微秒级水平，随着数据量增大，二者延迟呈现同步增大趋势。得益于嵌入式 RDMA 端将部分软件栈功能卸载至硬件，嵌入式 RDMA 端主动向商用对端发起的 WRITE 操作延迟，以及商用对端向嵌入式 RDMA 端发起的 WRITE 操作延迟均相对较低。上述关于带宽和延迟的观察结果充分证明嵌入式 RDMA 能够实现与标准 RDMA 的高带宽轻量化传输兼容性，在主动 WRITE 和被动 READ 方面性能更优。

我们分别进行了 2 组长达 72 h 的压力测试，用于模拟并验证嵌入式 RDMA 在高速传感场景下的运行带宽和延迟稳定性。图 17 展示了 72 h 内带宽与延迟的动态变化情况，测试结果表明，在整个测试周

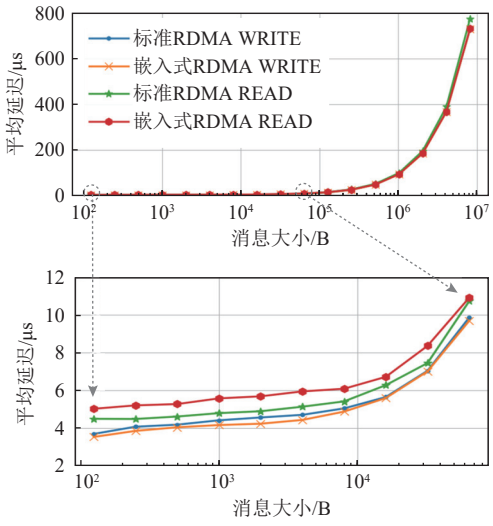


Fig. 16 Variation trend of system transmission delay with message size

图 16 系统传输延迟随消息大小的变化趋势

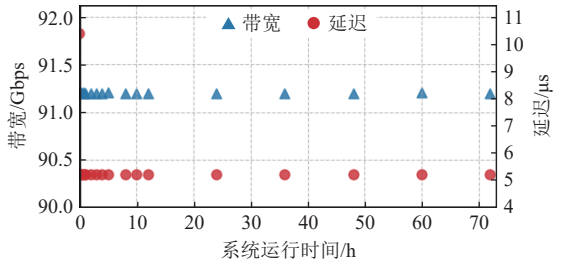


Fig. 17 Stability evaluation of throughput and latency in stress test

图 17 压力测试中吞吐量与延迟的稳定性评估

期内系统带宽始终稳定在约 91 Gbps 的高位水平，波动幅度小于 0.1%，表现出卓越的带宽稳定性。在另一组实验中，传输延迟在整个测试过程中始终维持在约 5μs 的稳定水平，未出现明显的性能抖动或异常波动。需要说明的是，由于系统启动会有短暂的突发状况，因此前期极少报文延迟达到了 10.3 μs，随着带宽的稳定，延迟快速收敛至 5 μs。这一长时间连续测试结果充分验证了嵌入式 RDMA 协议栈在持续高负载条件下的可靠性与鲁棒性，能够满足高速传感场景下对传输系统长期稳定运行的严苛要求，为实际工程部署提供了有力的性能保障。

4 总 结

本文对 RDMA 标准软件栈和处理流程进行了详细分析，并据此提出面向高速传感数据采集的嵌入式 RDMA 轻量化软件栈技术及实现方法，从 RDMA 建链和 RDMA 软件驱动 2 个方面进行 RDMA 软件栈轻量化。其中，基于可靠 UDP 的轻量化 RDMA 建链保证了通信双端前期配置阶段信息交互；基于可靠以太网的 RDMA 元数据配置保证了嵌入式端的 RDMA 标准兼容性；基于可靠以太网的 RDMA 传输参数配置和传输控制软件功能选择性卸载，在保障扩展灵活性的同时避免了传感端弱核影响嵌入式端传输性能，满足了嵌入式系统应用场景需求，提升全系统能力。实验结果表明，相较于标准 RDMA 软件栈，本文所设计的轻量化嵌入式 RDMA 软件栈可在无高性能 CPU、无操作系统的约束下正常运行，为高速传感数据采集场景的高带宽、低延迟网络传输提供了可靠的软件支撑，并构建了从传感端到标准端的完整系统解决方案。

作者贡献声明：杨惠和李韬提出并实现了基于可靠 UDP 的轻量化 RDMA 建链方法和基于可靠以太网的轻量化 RDMA 驱动，构建了论文整体框架，并撰写论文；张毓涛和黄曼蒂撰写论文，搭建原型系统与验证实验；原玉磊参与了轻量化 RDMA 建链和轻量化 RDMA 驱动的实现过程，提供了原型系统搭建条件；汪清和李韬负责论文审阅并提出指导意见。

参 考 文 献

[1] Huang Mandi, Li Tao, Yang Hui, et al. Survey on Ethernet RDMA network interface card[J]. Journal of Computer Research and Development, 2025, 62(5): 1262−1289(in Chinese)
(黄曼蒂, 李韬, 杨惠, 等. 以太网 RDMA 网卡综述[J]. 计算机研究与发展, 2025, 62(5): 1262−1289)

[2] Chowdhury A, Karmakar G, Kamruzzaman J, et al. Attacks on self-driving cars and their countermeasures: A survey[J]. IEEE Access, 2020, 8: 207308−207342

[3] Xia Dan, Jiang Chun, Wan Jiafu, et al. Heterogeneous network access and fusion in smart factory: A survey[J]. ACM Computing Surveys, 2023, 55(6): 113.1−113.31

[4] Luo Huimin, Zhang Jiao, Yu Mingxuan, et al. SeqBalance: Congestion-aware load balancing with no reordering in data center networks[J]. IEEE Internet of Things Journal, 2024, 12(13): 25707−25719

[5] Xue Jiachen, Chaudhry M U, Vamanan B, et al. Dart: Divide and specialize for fast response to congestion in RDMA-based datacenter networks[J]. IEEE/ACM Transactions on Networking, 2018, 28(1): 322−335

[6] Wang Xiaoliang, Song Hexiang, Nguyen C, et al. Maximizing the benefit of RDMA at end hosts [C] // Proc of the 2021 IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2021: 1−10

[7] Mittal R, Shpiner A, Panda A, et al. Revisiting network support for RDMA [C] //Proc of the 2018 ACM SIGCOMM Conf. New York: ACM, 2018: 313−326

[8] Li Yuliang, Miao Rui, Liu H H, et al. HPCC: High precision congestion control [C] // Proc of the 2019 ACM SIGCOMM Conf. New York: ACM, 2019: 44−58

[9] Bai Wei, Abdeen S S, Agrawal A, et al. Empowering Azure storage with RDMA [C] // Proc of the 20th USENIX Symp on Networked Systems Design and Implementation (NSDI). Berkeley, CA: USENIX Association, 2023: 49−67

[10] Kim D, Yu Tianlong, Liu H H, et al. FreeFlow: Software-based virtual RDMA networking for containerized clouds [C] // Proc of the 16th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2019: 113−126

[11] Wang Zilong, Luo Layong, Ning Qingsong, et al. SRNIC: A scalable architecture for RDMA NICs [C] // Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 1−14

[12] Wang Xizheng, Chen Guo, Yin Xijin, et al. StaR: Breaking the scalability limit for RDMA [C] // Proc of the 29th IEEE Int Conf on

Network Protocols. Piscataway, NJ: IEEE, 2021: 1−11

[13] Lee J B, Kim S J, Kim W H, et al. RDMA-based sampling port of ARINC-653[J]. IEEE Embedded Systems Letters, 2024, 16(4): 437−440

[14] Feng Junjie, Chen Xianzhang, Liu Duo, et al. eRDAC: Efficient and reliable remote direct access and control for embedded systems[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(11): 3685−3696

[15] Schelten N, Steinert F, Knapheide J, et al. A high-throughput, eesource-efficient implementation of the RoCEv2 remote DMA protocol and its application[J]. ACM Transactions on Reconfigurable Technology, 2022, 16(1): 1−23

[16] Cossettini A, Taranov K, Vogt C, et al. A RDMA interface for ultra-fast ultrasound data-streaming over an optical link [C] // Proc of the 2022 Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2022: 80−83

[17] AMD. AMD embedded RDMA enabled NIC LogiCORE IP product guide(PG332)[EB/OL].(2024-11-22)[2025-06-10].https://docs.xilinx.com/r/-en-US/pg332-ernic

[18] InfiniBand. InfiniBand architecture specification[EB/OL]. [2025-06-10]. https://www.infinibandta.org/ibta-specification/

[19] IETF Network Working Group. RFC5040: A remote direct memory access protocol specification[EB/OL]. (2007-10-01)[2025-06-10]. https://www.rfc-edi tor.org/rfc/rfc5040

[20] NVIDIA. RDMA over converged ethernet(RoCE)[EB/OL]. [2025-06-10]. https://docs.nvidia.com/networking/display/onyxv3103004/rdma+over+converged+ethernet+(roce)

[21] He Zhiqiang, Wang Dongyang, Fu Binzhang, et al. MasQ: RDMA for virtual private cloud [C] // Proc of the 2020 ACM SIGCOMM Conf. New York: ACM, 2020: 1−14

[22] OpenTSN Group. Yinhe Hengxin HX-DP4C Edge Intelligence Network Processor[EB/OL]. Hunan: OpenTSNGroup. (2023-08-30)[2025-06-10]. https://mp.weixin.qq.com/s/gh2no71EwjX0a-HGHLiPJg (in chinese)
(OpenTSN 工作组. 银河衡芯 HX-DP4C 边缘智能网络处理器[EB/OL]. 湖南：OpenTSN 工作组，(2023-08-30)[2025-06-10]. https://mp.weixin.qq.com/s/gh2no71EwjX0a-HGHLiPJg)

[23] AMD. The AMD Virtex UltraScale+ FPGA VCU118[EB/OL]. [2025-06-10].https://www.amd.com/zh-cn/products/adaptive-socs-and-fpgas/evaluation-boards/vcu118.html#tabs-e09e115b99-item-fb75e43802-tab



Yang Hui, born in 1986. PhD, associate professor. Her main research interests include RDMA and network processor.

杨　惠, 1986 年生。博士，副研究员。主要研究方向为 RDMA、网络处理器。



Zhang Yutao, born in 2000. Master. His main research interests include RDMA and data center congestion control.

张毓涛, 2000 年生。硕士。主要研究方向为 RDMA、数据中心拥塞控制。



Huang Mandi, born in 1999. Master. Her main research interest includes RDMA.

黄曼蒂, 1999 年生。硕士。主要研究方向为 RDMA。



Yuan Yulei, born in 1984. PhD, associate professor. Member of CCF. His main research interests include high-performance networks and network security.

原玉磊, 1984 年生。博士，副研究员。CCF 会员。主要研究方向为高性能网络、网络安全。



Wang Qing, born in 1982. PhD, professor. Her main research interests include communication and radar signal processing, deterministic networking, and intelligent information processing.

汪　清, 1982 年生。博士，教授。主要研究方向为通信雷达信号处理、确定性网络、智能信息处理。



Li Tao, born in 1983. PhD, associate professor. His main research interest includes high-performance network system and chip.

李　韬, 1983 年生。博士，副研究员。主要研究方向为高性能网络系统及芯片。