2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) | 978-1-6654-0876-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/WoWMoM54355.2022.00052

WIP: When RDMA Meets Wireless

Tong Li[†‡], Ke Xu[§¶‖], Hanlin Huang[§], Xinle Du[§], Kai Zheng[†]
Huawei[†], Renmin University of China[‡], Tsinghua University[§], BNRist[¶], PCL[‖]
Email: tong.li@ruc.edu.cn, xuke@tsinghua.edu.cn, (hhl21, dxl18)@mails.tsinghua.edu.cn, kai.zheng@huawei.com

***Abstract*—The emerging applications including AR/VR interactive gaming, ultra-high-definition live streaming, 4K wireless projection, Metaverse, etc. imply the demand for ultra-low latency and ultra-high bandwidth wireless transmission. The legacy kernel TCP stack is not fully satisfactory because it induces the CPU bottleneck on hosts. In this paper, we propose Wireless-RDMA (W-RDMA) that enables RDMA in wireless networks to tackle the CPU bottleneck issue on wireless hosts. The feasibility of W-RDMA is demonstrated through testbed experiments. Technical challenges and future opportunities are further discussed. We believe it is a small but crucial step for enabling RDMA for wireless transmission.**

***Index Terms*—RDMA, wireless network, CPU offloading, high bandwidth**

I. Introduction

In recent years, the popularity of video-based interactive applications, such as AR/VR interactive gaming, Ultra-High Definition (UHD) live streaming, 4K wireless projection, and even Metaverse, has grown by leaps and bounds with the ubiquity of high-resolution mobile terminals [1]. Despite the emerging applications, TCP/IP is still the dominant transport/network stack in modern networks. However, the legacy kernel TCP stack is not fully satisfactory as discussed below.

CPU bottleneck restricts performance of high-bandwidth technologies. With the development of new communication technologies such as 5G, WiFi 6, and WiFi 7, the wireless transmission capability of wireless hosts has been greatly enhanced. For example, WiFi 6 based on IEEE 802.11ax can provide a maximum bandwidth of 9.6 Gbps, and WiFi 7 based on IEEE 802.11be is expected to have a maximum transmission bandwidth of at least 30 Gbps. However, it is well studied that when transferring a large volume of traffic via the legacy kernel TCP stack, the system kernel has to handle operations such as context switching and memory copies, which induce high CPU overhead on the hosts [2]. Based on these observations, although wireless hosts are empowered with ultra-high bandwidth from the hardware perspective, a wide gap between the actual transmission capacity and the target value might exist due to the host CPU bottleneck, which restricts the performance of new technologies.

Ultra-low latency of wireless applications requires ultra-high bandwidth. High-resolution and video-based interactive applications such as wireless projection require ultra-low latency. For example, a live 4K video requires at least 30 frames per second, which means for each frame the handling progress including encoding, transmitting, and decoding has to be completed within 60 ms [3]. However, even without considering transmission latency, the software-based codec latency of a 4K video (180-250 ms) is far beyond what can be tolerated [4]. Applying hardware-based codec can alleviate the problem in a way; however, the flexibility of customization makes the software-based way still widely used in commercial products [5]. This reveals that the CPU-intensive codec is becoming the bottleneck of high-resolution and interactive applications in commercial devices. In this case, reducing the overhead of codec (or even sending raw frames without codec) but asking for more bandwidth, has the potential to achieve lower end-to-end latency. In other words, ultra-high bandwidth validates ultra-low latency.

In summary, the CPU bottleneck on hosts is becoming the hurdle for wireless transmission to achieve ultra-low latency and ultra-high bandwidth, and the usage of the legacy kernel TCP stack is one reason. It is also well studied that RDMA (Remote Direct Memory Access), a technique to access memory remotely and directly, can bypass the system kernel and perform packet parsing in hardware without software involvement. Specifically, RDMA reduces over 95% of the CPU utilization on hosts [2]. Recently, almost all studies [2], [6] related to RDMA are in the context of data center networks. They have demonstrated that RDMA-based protocol stack (e.g., RDMA over Converged Ethernet (RoCE) v2) achieves lower latency and higher bandwidth compared to kernel TCP stack. While RDMA-based applications are currently in the wired domain. Considering the great potential of RDMA in tackling the CPU bottleneck issue, enabling RDMA for wireless transmission to achieve ultra-low latency and ultra-high bandwidth would be a relevant contribution.

In this paper, we for the first time propose the concept of Wireless-RDMA (W-RDMA). W-RDMA enables RDMA for wireless transmission and tackles the CPU bottleneck issue on wireless hosts. While W-RDMA shows great potential for wireless transmission improvement, we expose several technical challenges that should be overcome for wide deployment in practice. For example, (1) lossless transmission guarantee, (2) interface adapter design, and (3) dedicated hardware support. Then we design a testbed to demonstrate the feasibility of W-RDMA and further discuss the gap between the testbed and the full implementation of W-RDMA protocol stack.

This work is supported by Huawei, the China National Funds for Distinguished Young Scientists (61825204), the NSFC Projects (61932016 and 62132011), and the Beijing Outstanding Young Scientist Program (BJJWZYJH01201910003011).

978-1-6654-0876-9/22/$31.00 ©2022 IEEE
DOI 10.1109/WoWMoM54355.2022.00052

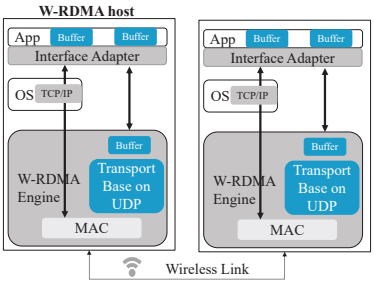

Fig. 1. The architecture of W-RDMA

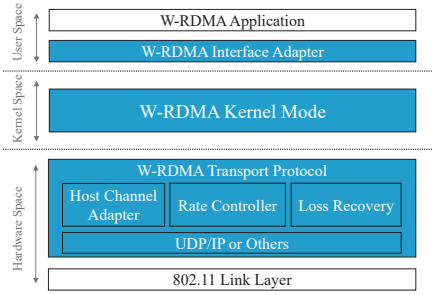

Fig. 2. The W-RDMA protocol stack

II. Related Work

To resolve data forwarding bottlenecks, some software-based kernel bypass solutions [7] such as Data Plane Development Kit (DPDK), provide a set of user-space libraries and drivers that accelerate packet-processing workloads. These methods reduce interruptions and memory copies, however, (1) protocol stack is only moved to the user space which is still CPU-consuming. (2) They waste CPU resources when hosts are low-loaded.

TCP Offload Engine (TOE) extends the TCP/IP protocol stack so that some function of the TCP/IP protocol is transferred from the CPU to the TOE hardware. The checksum calculation of the IP header, TCP header, and UDP header is calculated by the NIC instead of CPU, reducing the burden on the CPU. However, (1) TOE only provides speedups under a limited set of workload conditions [8]. (2) The context switch and system calls of TOE still take considerable CPU overhead.

Nowadays, a number of RDMA-based transport layer designs in data center have been proposed [2], [6]. These studies start from the causes of congestion and address them by some mechanisms such as receiver-side driving, selective ECN marking, and dynamic RTT changes. However, for wireless networks, existing commercial wireless routers do not support ECN and bandwidth is prone to change, which are significantly different from data center networks. Note that the non-decompression method [9] can be applied to cooperate with RDMA to further reduce CPU overhead on hosts, improving performance and storage efficiency. However, this is out of the scope of this paper.

To the best of our knowledge, this paper is the first work that investigates the feasibility, challenges, and opportunities when RDMA meets wireless, in order to fundamentally tackle the CPU bottleneck issues in wireless networks.

III. W-RDMA Overview

RDMA is a technique to access memory remotely and directly that bypasses the operating system and kernel and performs packet parsing in hardware without host CPU involvement. On this basis, W-RDMA is a scheme designed to enable RDMA for wireless transmission.

A. The Architecture of W-RDMA

Figure 1 shows the module interactions of two protocols, i.e, TCP/IP and W-RDMA, respectively over wireless links. For TCP/IP, data in the application buffer are sent through Socket APIs, passed into the operating system for processing, and then transmitted to the receiver through wireless MAC. For W-RDMA, data at the application layer does not pass through the operating system, but directly enters the cache of the W-RDMA engine and is transmitted to the other receiving host via UDP frames. At the receiver, the data packets likewise do not pass through the operating system, but are directly handed over to the W-RDMA engine for processing and passed to the upper-layer application afterwards. In this architecture design of W-RDMA, proprietary hardware is expected to support RDMA, and W-RDMA oriented protocols and application interfaces are also required, which we will discuss next.

B. The W-RDMA Protocol Stack

As illustrated in Figure 2, the W-RDMA protocol stack obeys a hierarchical design, including hardware space, kernel space, and user space.

Hardware space. Inspired by RoCEv2, the W-RDMA transport protocol should be implemented in the hardware space. Different from the RDMA protocol stack in the wired domain, the link layer is based on IEEE 802.11 standards, including 802.11a/b/g/n, 802.11ac, 802.11ax, etc. In addition, the W-RDMA transport protocol should include a Host Channel Adapter. This is because that wireless channel is the medium for wireless network to send and receive data, and running on the right channel helps to speed up connection speed [10]. The number of channels varies for different frequency bands, and adjacent channels are prone to interfere with each other. Specifically, the Host Channel Adapter is to detect the channel usage and select the channel that suffers the least interference to ensure the advantage of high bandwidth of W-RDMA. Besides, the W-RDMA transport layer can reuse UDP as the transport-layer carrier but add the specific modules such Rate Controller (similar to DCQCN for RDMA) and Loss Recovery (e.g., Go-Back-N).

Kernel space. Unlike the kernel space of TCP/IP, W-RDMA's kernel space performs fewer operations including capturing and forwarding packets, encapsulating data, and performing task scheduling.

User space. The user space of W-RDMA requires a carefully designed W-RDMA Interface Adapter between the application layer and kernel space. Generally, W-RDMA Interface Adapter

should at least include a framework of unified north APIs and wireless channel selection APIs. The unified north APIs provide original interfaces for data transmission, such as SEND, RECEIVE, WRITE, READ, etc. The wireless channel selection APIs provide a way for applications to customize the policy of channel selection. The design of these APIs is not straightforward but requires overcoming some challenges as discussed in the next section.

C. Challenges for Applying W-RDMA

While W-RDMA shows great potential for wireless transmission improvement, according to the above-discussed design of W-RDMA, we list several technical challenges that should be overcome for wide deployment in practice.

Lossless transmission guarantee. It is well-known that ensuring lossless transmission is a prerequisite to guarantee the performance benefits of RoCEv2. This is because RDMA adopts the Go-Back-N retransmission mechanism, which seriously loses performance when the loss occurs. Due to poor wireless signal strength caused by natural or human-made interference and limited software and hardware resources in mobile terminals, wireless transmission suffers from more unpredictable network conditions than wired one. Hence, it is more challenging for the W-RDMA based transmission control to guarantee lossless transmission. On the one hand, W-RDMA should avoid packet loss as much as possible. On the other hand, W-RDMA should recover as quickly as possible in the case of packet loss without inducing much overhead.

Interface adapter design. As mentioned above, the W-RDMA Interface Adapter may include unified north APIs and wireless channel selection APIs. For the unified north APIs, W-RDMA should basically reuse as many as possible the functions of legacy RDMA verbs APIs, including SEND, RECEIVE, WRITE, READ, etc. However, (1) different from the service-oriented data center networks, W-RDMA might be commonly used in application-oriented wireless networks. Besides, (2) the RDMA verbs APIs are relatively complicated for some applications of other platforms such as ARM-based Android. Based on these observations, novel and relatively simple, a set of unified north APIs is required for W-RDMA to support both zero-copy and hardware offloading for high-performance transmission. For wireless channel selection APIs, they are designed to interact with the Host Channel Adapter at the hardware layer. This enables a cross-layer application-customizable wireless channel adaption.

Dedicated hardware support. Kernel bypass capability of RDMA hosts is mainly supported by dedicated hardware, i.e., NICs. Although nowadays the Wi-Fi 6 and 5G communication modules have been widely equipped, no dedicated W-RDMA enabled chips exist for mobile terminals such as smartphones. It is therefore a great challenge for the W-RDMA deployment with the absence of dedicated smart NICs for W-RDMA.

IV. A Testbed of W-RDMA

The topology of the W-RDMA testbed is illustrated in Figure 3. Two hosts are equipped with RDMA-enabled smart

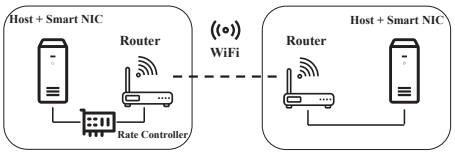

Fig. 3. Testbed topology

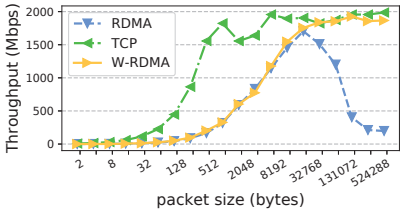

Fig. 4. Throughput trend with packet size

NICs with a high bandwidth capability of 10 Gbps, and each host is connected to a 10-GbE router through a wired link to form a logic wireless terminal, and the communication between the two logic wireless terminal is still over Wi-Fi links (e.g., Wireless Distribution System (WDS)).

Packet loss significantly impacts performance of RDMA-based protocol stack. In order to control the sending rate of RDMA hosts, as shown in Figure 3, the W-RDMA testbed requires a Rate Controller implemented between the host and the router. Specifically, the Rate Controller first sets up a *rate adapter* to fit the rate to wireless link capacity, then adopts a *PFC controller* as the last resort mechanism to guarantee lossless transmission. In addition, a *forwarding module* needs to be added for L2 layer data forwarding.

Discussion: There are some notable points that are important for reasoning why this testbed can evaluate of the feasibility of W-RDMA. We briefly describe these points below.

First, the proto verbs operations of RDMA are still used without unified north APIs in this testbed. Since it does not affect the performance of W-RDMA wireless transmission, the evaluation of feasibility for W-RDMA remains constant.

In addition, to ensure lossless transmission, we control the sending speed of the host by adding a rate controller between it and the router. Although it does not provide a perfect set of congestion control protocols, it proves the feasibility of W-RDMA to be utilized by wireless applications through adequate experiments. In the future, a complete set of congestion algorithms should be designed to better ensure that W-RDMA does not lose packets and recovers quickly after packet loss in a wireless environment.

V. Testbed Evaluation

In this section, we evaluate the performance of W-RDMA. As shown in Figure 3, two hosts (DELL PowerEdge R610) are equipped with RDMA smart NIC supporting RoCE v2 protocol. The rate controller is a common server with two 82599ES NICs, which serve as both incoming and outgoing

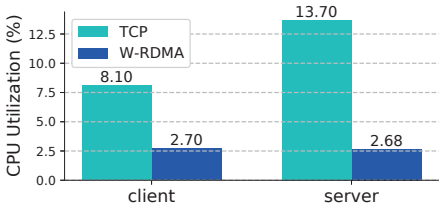

Fig. 5. CPU utilization

ports for route forwarding. Both routers are WiFi 6 wireless routers (ASUS RT-AX89X).

Throughput. We first explore the throughput trend of W-RDMA when packet size varies. Specifically, W-RDMA tests are performed by the ib_send_bw proto operation, compared with TCP's *iperf 3* with the packet size in the range of [2, 1048576] bytes. We also run tests using the legacy RDMA without the extra Rate Controller introduced in Section IV. Figure 4 shows the results. Generally, throughput increases with the increase of packet size. This is because the on-off flow pattern cannot fully utilize bandwidth when packet size is small. When packet size is large enough (e.g., over 32768 bytes), RDMA's sending rate exceeds the wireless bandwidth, which induces packet losses and retransmissions. This seriously affects the performance of RDMA. W-RDMA outperforms RDMA because an extra Rate Controller is introduced to avoid losses. Figure 4 also demonstrates that W-RDMA approximates TCP in throughput when transmitting large-size packets. Note that the maximum throughput of both TCP and W-RDMA are below 2 Gbps; this is due to the bandwidth limitation of the WDS link between two wireless routers. However, this is out of the scope of this paper.

CPU utilization. We then compare the CPU utilization of kernel TCP and W-RDMA with the same packet size. In particular, we set packet size as 30000 bytes to ensure that they achieve a similar throughput (i.e., TCP is 1704 Mbps and W-RDMA is 1693.92 Mbps). Figure 5 shows the results. It is obvious that W-RDMA is able to significantly reduce CPU overhead on hosts. Specifically, W-RDMA saves 66.67% and 80.44% of CPU resources on the client host and the server host, respectively. This validates W-RDMA in a way.

VI. Future Opportunities

This work is the first transition of RDMA from wired domain to wireless domain. We have tried to verify the feasibility of W-RDMA via testbed experiments. In addition to the challenges discussed in Section III-C, the proposed W-RDMA further provides new possibilities and opportunities for the future development of RDMA over wireless.

Minimalist device-to-device transport protocols. The IP layer in the W-RDMA hardware space (see Figure 2) is not as mandatory as in the wired domain. This is because the local wireless networks are relatively confined, and can forward packets according to the MAC address space. In this case, the IP-based UDP may be replaced by a minimalist device-to-device transport protocol that abandons the IP layer. This further enhances the high performance of W-RDMA.

Cross-layer optimizations. W-RDMA still follows the IEEE 802.11-based standard in the link layer, but it should apply a cross-layer design on transmission control. Specifically, upon receiving the MAC data frames, the NIC does not send back the MAC-layer block-ACKs immediately. Instead, it processes the data frames and generates flow control information. This information carried by MAC-layer block-ACKs is then fed back to the W-RDMA transport layer. If no ACK event is received within a certain period of time, W-RDMA might retransmit the packet and adjusts sending rate accordingly [11].

VII. Conclusion

This paper proposes W-RDMA and investigates its feasibility of enabling RDMA in wireless transmission. W-RDMA differs from legacy RDMA in the following aspects. (1) The link layer of W-RDMA is based on the IEEE 802.11 standards. (2) W-RDMA transport protocol should include a Host Channel Adapter for better channel management. (3) It is more challenging for W-RDMA to avoid packet loss and to recover losses efficiently. (4) A set of simpler user APIs is required for W-RDMA to support both zero-copy and hardware offloading for high-performance transmission. (5) Dedicated W-RDMA enabled smart NICs. The future work includes overcoming the above mentioned challenges as well as the design of minimalist device-to-device transport protocols, and cross-layer optimizations.

VIII. Acknowledgment

This work was partly done when Tong Li was a full-time Chief Engineer at Huawei. We thank Kun Tan, Binzhang Fu, Jie Li, Yalei Wang, and Bojie Li for feedback throughout this project. Hanlin Huang is the corresponding author.

References

[1] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Tack: Improving wireless transport performance by taming acknowledgments," in *ACM SIGCOMM*, 2020.
[2] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," *ACM SIGCOMM CCR*, 2015.
[3] J. Deber, R. Jota, C. Forlines, and D. Wigdor, "How much faster is fast enough? user perception of latency & latency improvements in direct and indirect touch," in *ACM CHI*, 2015.
[4] D. Xu, A. Zhou, X. Zhang, and et al., "Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption," in *ACM SIGCOMM*, 2020.
[5] Agora, "Agora sdk," https://www.agora.io/en, 2021.
[6] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," *ACM SIGCOMM CCR*, 2015.
[7] Intel, "Dpdk," https://www.dpdk.org/, 2021.
[8] N. L. Binkert, A. G. Saidi, and S. K. Reinhardt, "Integrated network interfaces for high-bandwidth tcp/ip," in *ACM ASPLOS*, 2006.
[9] F. Zhang, J. Zhai, X. Shen, O. Mutlu, and X. Du, "POCLib: a high-performance framework for enabling near orthogonal processing on compression," *IEEE TPDS*, 2022.
[10] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Revisiting acknowledgment mechanism for transport control: Modeling, analysis, and implementation," *IEEE/ACM TON*, 2021.
[11] T. Li, K. Zheng, and K. Xu, "Acknowledgment on demand for transport control," *IEEE Internet Computing*, 2021.