Multimedia Tools and Applications (2023) 82:28389–28404
https://doi.org/10.1007/s11042-023-14781-8

An implementation and evaluation of MPTCP-based IoT router

Taichi Tsuru[1] · Mikio Hasegawa[2] · Yozo Shoji[3] · Kien Nguyen[1,4] · Hiroo Sekiya[1]

Received: 2 November 2021 / Revised: 4 May 2022 / Accepted: 5 February 2023 /
Published online: 3 March 2023

Abstract

The evolving Internet of Things (IoT) promisingly improves the quality of life and transforms many industries. However, the IoT application challenges the wireless networks since the resource-constrained IoT devices typically need to send data to the cloud or edge server. Therefore, it is necessary to introduce an intermediate device between IoT devices and the servers, for example, to reduce the cost of direct communication between them. In another case, the device may move and collect the data from IoT devices before transmitting it to the server. The intermediate device should be designed to have resilient Internet connections and sufficient bandwidth in such a context. This work implements and evaluates a Multipath TCP (MPTCP) IoT router, which uses multiple radios to connect a server to address the demanding design. The router leverages MPTCP, an extension of TCP for simultaneous transmission over several paths on top of Wi-Fi interfaces. MPTCP has also supported several working modes for throughput and (or) resilience enhancements. First, we implement the MPTCP kernels, which can run on the popular IoT devices Raspberry Pi 3B+ and 4. Second, we extensively evaluate the performance of IoT routers in a static and mobility scenario. The static scenario's evaluation results show that the MPTCP-based router

✉ Kien Nguyen
nguyen@chiba-u.jp

Taichi Tsuru
mrt726625@icloud.com

Mikio Hasegawa
hasegawa@ee.kagu.tus.ac.jp

Yozo Shoji
shoji@nict.go.jp

Hiroo Sekiya
sekiya@faculty.chiba-u.jp

[1] Graduate School of Engineering, Chiba University, Chiba, Japan

[2] Department of Electrical Engineering, Tokyo University of Science, Tokyo, Japan

[3] Social-ICT System Laboratory, National Institute of Information and Communications Technology, Tokyo, Japan

[4] Institute for Advanced Academic Research, Chiba University, Chiba, Japan

can achieve seamless handover and bandwidth aggregation. In the mobility scenario, the MPTCP router with one backup path performs better than the single-path TCP. Besides, the MPTCP routers are more energy-efficient than TCP on the same hardware.

Keywords MPTCP · IoT · Evaluation

1 Introduction

The recent advances in electronics, sensing, and communication technologies have facilitated the IoT applications that improve human quality of life and open industrial opportunities. We have seen various IoT applications such as smart homes [33], smart cities [5, 9], air quality monitoring [31], or even fighting COVID-19 [21]. In the IoT applications, IoT services are typically provided by exchanging information and data by many IoT devices to edge or cloud servers via internetworks. On the other hand, the IoT devices, which have sensors that sense the environment, are generally resource-constrained. In several applications with many connected IoT devices, keeping direct device-server communication may cause extra communication costs and sophisticated network management. Moreover, the IoT devices' sensing data are usually small; it may be more efficient if stored and relayed with a sufficient amount. The solution for those cases is to introduce a device that acts as an intermediate between various IoT devices and the edge or cloud servers. The intermediate device should have an Internet connection with sufficient bandwidth to face the demand for the former exemplary application. In the latter one, the Internet connection should be resilient since the device may move to collect the sensing data.

There are many previous works in the literature to realize such devices, named IoT gateways, which commonly have many communication technologies to support heterogeneous IoT devices. However, the works mainly focus on the link between IoT devices to the gateway while using one wireless link for the Internet connections. Moreover, most of them are specialized in platforms that are much more powerful than IoT devices. If the IoT gateway can be built on top of the IoT device, it will speed up the IoT development. Therefore, the challenges of a resilient Internet connection and sufficient bandwidth remain. One viable approach for adequate bandwidth provision is to equip the router with high-throughput, reliable wireless radio. However, it increases the cost and adds extra overheads to the device's size, software, power consumption, etc. Another way for the bandwidth and resilience demand is to deploy several IoT gateways for the IoT application. Unfortunately, that deployment may significantly add complexity to the network deployment. We follow a novel approach to building the device, which has several low-cost wireless radios to address the challenges in this work.

Although there are potential benefits of introducing several radios on the IoT device, the radios do not themselves guarantee the expected bandwidth and resilience. We need to devise new techniques and mechanisms that can enable applications to use multiple radios simultaneously. To this end, Multipath TCP, which can aggregate the bandwidth over radio links and realize the high throughput, is a promising technology [7]. MPTCP can also enhance resilience against path failure due to the flexibility in adding and removing paths. MPTCP has been actively developed on various platforms (e.g., Linux, Android, Apple iOS). Besides, MPTCP has been successfully deployed and evaluated in many wireless networks, including Wi-Fi, 4G, 5G, etc. However, there have no official MPTCP kernels for IoT devices yet. Moreover, the operation of MPTCP on IoT devices or IoT routers has not been investigated in the literature. Those issues motivate us to build and evaluate an IoT

router that exploits the advantages of MPTCP in the IoT context. We name our device IoT router since MPTCP needs the information from the routing layer to operate efficiently and effectively.

In this paper, we implement and evaluate an MPTCP-based IoT router, which can concurrently use several Wi-Fi radios to communicate to an application server. The router not only uses the hardware of IoT devices but also achieves bandwidth aggregation and resilience enhancement. More specifically, our contributions are as follows.

- We have implemented the MPTCP-based IoT routers, which use new MPTCP kernels and Wi-Fi interfaces, on Raspberry Pi models 3B+ and 4. We have then evaluated the IoT routers in actual networks under static and mobile conditions.
- In the static scenario, the evaluation results show that the achieved MPTCP throughput of the IoT router is higher than the TCP one via each Wi-Fi network. Moreover, the MPTCP-based router's communication is resilient against path failures. In the mobility scenario, the MPTCP-based router can achieve seamless communication.
- The evaluation results also indicate that the MPTCP communication is more energy-efficient than the TCP one.

The remainder of the paper is organized as follows. Section 2 describes related works. In Section 3, we introduce our implementation of the MPTCP-based IoT router. Section 4 presents the evaluation results. Finally, Section 5 includes conclusion and future works.

2 Related works

It is widely agreed that it is necessary to have a device that acts as an intermediate between various IoT devices and the cloud/edge platform over the Internet. This paper's IoT router concept is close to the IoT gateways proposed in the literature [6, 11, 32]. The IoT gateway is a bridge to connect wireless sensor networks with sensing capabilities to the Internet, forming the Internet of Things. The IoT gateway normally integrates networking protocol, manages storage, and preprocesses sensing data. There have been many works focusing on the interface between IoT devices and IoT gateways. They have aimed to solve the scalability issue and automatic association when an IoT device joins a coverage area of an IoT gateway. The gateway may require a manual configuration of IoT devices, for example, the passive IoT gateway in [3]. Besides, semi-automatic gateways can automatically create the link to IoT devices but are not ready for applications [3]. In [14], the authors proposed a fully automatic IoT gateway, which can be self-configured. The gateway can discover and update IoT devices automatically. In [1], a mobile gateway that is a smartphone opportunistically interacts with IoT devices. Those works, however, focused on the side between the gateways to IoT devices. They considered a stable connection from the IoT gateways to the Internet. This work addresses the problem of sufficient and resilient Internet connection of IoT gateway/router.

The work in [10] investigated the connection from an IoT gateway to the Internet. The authors proposed a smart gateway with several wireless interfaces, each of which can connect to the Internet. Moreover, they introduced a smart switching mechanism that switches Internet links to provide a seamless connection to IoT devices. Unfortunately, they only presented the concept without implementation or simulation. This paper also considers the multiple links to a server, but we provide a practical solution with an MPTCP-based device. In [29], the authors introduced a 5G-enabled IoT gateway to support a large number of IoT devices and a vast amount of IoT traffic. They also proposed an uplink compression scheme

to optimize the IoT gateway's Internet connection usage. However, the work exploited the advanced features of 5G, such as millimeter-wave (mmwave), C-RAN, Massive MIMO, etc., which generally required specialized, powerful hardware. Our work targets an IoT gateway built on top of IoT hardware that also provides the expected resilience and bandwidth sufficiency with MPTCP.

MPTCP has already been available on many platforms, including Linux [18], FreeBSD [8], Android [19], iOSs [2]. The Linux community has recently merged the MPTCP kernel into the mainstream kernel. The broad range and speed of MPTCP adoption originate from its evolvability and efficiency. First, MPTCP can be used similarly to TCP without modifications to the application or networking infrastructure. Second, the MPTCP-enabled application can use diverse paths, achieving throughput and resilience enhancements. MPTCP was originally designed for 4G mobile networks with LTE and Wi-Fi [27]. Since then, there have been many improvements to the MPTCP kernel for such a scenario. For example, in [24], the authors proposed a new MPTCP kernel with a fast and secure initialization to improve the MPTCP performance. The work in [22] added path-awareness information to the MPTCP scheduler to deal with the imbalance of 4G/LTE and Wi-Fi paths. MPTCP also finds its application in various mobile wireless systems, including Software defined wireless networks [23], mmwave wireless [25], or 5G [17]. Regarding the MPTCP implementation on an IoT device, the work in [20] introduced an MPTCP-based Robot Operating System (ROS) kernel on Raspberry Pi. However, the authors did not mention the implementation in detail. Moreover, the evaluation is limited to a static scenario. Unlike previous works, this paper exploits the MPTCP kernel implementation on different models of Raspberry Pi (i.e., IoT devices). Besides, we have extensively evaluated the MPTCP-based devices on static, mobile scenarios, considering various performance parameters.

3 MPTCP and MPTCP-based IoT router

3.1 MPTCP overview

MPTCP is a protocol that extends TCP for communication over multiple paths. MPTCP inserts a virtual MPTCP layer between the application and transport layers. The application's byte stream goes to the TCP and lower layers without modification because the MPTCP layer uses a socket-like interface (i.e., metasocket). MPTCP has three essential components: path manager, scheduler, and congestion control. The path manager determines how to add subflows to existing connections. MPTCP supports the fullmesh, ndiffports, and binder path managers. The fullmesh one will create a full mesh of subflows among all available IP addresses. More specifically, each subflow is formed by a pair of IP addresses at two ends of a connection if an available route exists between them. Ndiffports will determine a predefined number of subflows across the same pair of IP addresses by modifying the source port. Binder integrates with the loose source routing function, mainly for aggregating subflows between a relay host and gateways, both MPTCP capable. The MPTCP scheduler is in charge of specifying the amount of data each will send on a subflow. MPTCP's default scheduler chooses the path that sends the subflow on the path with the smallest round trip time (RTT). Allocate any amount of data to the selected subflow, depending on the congestion window size. MPTCP can use TCP congestion controls on each subflow (i.e., uncoupled congestion control) or coupled ones. The latter ones are designed with the consideration of all subflows' status, such as Linked Increase Algorithm

(lia) [30], Opportunistic Linked Increase Algorithm (olia) [16], Balanced Linked Adaption Congestion Control Algorithm (balia) [26].

Since MPTCP has various implementations of different components, they can cooperate with others to form an operation mode depending on the user demands. Moreover, MPTCP can work under a backup mode, in which a subset of subflow is configured to be idle in the regular operation. They will be active for MPTCP transmission only when all other subflows are unavailable. Due to the dynamic of traffic and wireless networks, a significant difference in the throughput of each subflow may occur. In such a case, although consuming more resources, the aggregate throughput of MPTCP may be lower than that of TCP. The backup mode is essential in this context to maintain a good level of throughput. When a subflow is assigned the backup mode, the switching process is performed without delay.

3.2 MPTCP-based IoT router

3.2.1 Hardware

We aim to construct an MPTCP IoT router on a typical IoT device hardware instead of the powerful specialized one. Since the most active development of MPTCP is on Linux kernels, the device is expected to support Linux environments well. For those reasons, we select Raspberry Pi as the basic hardware for the IoT router development. Raspberry Pi is a small single-board computer increasingly popular in IoT applications because of its sufficient capability and reasonable price. Besides, Raspberry Pi is ready for various Linux distros and easy to customize. Up to date, there have been many versions of Raspberry Pi. We focus on the two recent ones in this work: Raspberry Pi model 3B+ and 4, both using ARM chips. The 3B+ model uses Broadcom BCM2837B0, Quad-core Cortex-A53 (ARMv8) 64-bit SoC, running at 1.4 GHz. Meanwhile, the 4 model has Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC, 1.5GHz. Moreover, the 3B+, 4 models have 1G, 4G RAM, respectively.

Regarding the connectivity, by default, there are several communication modules on the Raspberry Pi devices. The model 3B+ has Ethernet, Wi-Fi, Bluetooth 4.2, and Bluetooth Low Energy (BLE). As an improved version, the 4 model has better networking capability: Gigabit Ethernet, Wi-Fi, Bluetooth 5.0, BLE. Since the IoT device mainly uses wireless, we ignore the wired interface on the router. Moreover, Bluetooth, BLE is not native for TCP/IP, required for MPTCP operation; we hence select the Wi-Fi technology for the communication to the server. On each Raspberry Pi, there is only one Wi-Fi card; we attach an additional USB Wi-Fi card to create multiple physical paths. Technically, we could introduce other wireless technologies (e.g., LoRa, LTE, etc.) to the router in a similar way (via USB ports or GPIO). However, it may cause an extra cost.

3.2.2 Building MPTCP kernel

As there is no existing official MPTCP kernel for IoT devices, including Raspberry Pi, it is necessary to build a new one. There are two ways of creating a Linux kernel for a Raspberry Pi device: on the device (local building) or another device (cross-compiling). We select the latter since it takes a shorter period. In the following, we are going to describe the process of building a new MPTCP kernel for Raspberry Pi on a machine with Ubuntu 18.04 LTS. The process includes several steps as follows.

Step 1: Initially, we install the essential tools and dependencies for compiling the kernel, such as git, bc, bison, flex, libncurses5, libssl, etc. The detailed information and the software

Table 1 Tools and dependencies

Package	Version
build-essential	12.6
git	1:2.20.1-2+deb10u1
bc	1.07.1-2
bison	2:3.0.4.dfsg-1build1
flex	2.6.4-6
libncurses5-dev	6.1-1ubuntu1.18.04
libssl-dev	1.1.1-1ubuntu2.1 18.04.7

version are shown in Table 1. Besides, we also need to clone and install the ARM toolchain from the github repository on the Ubuntu machine. The toolchain is mandatory for cross-compiling.

Step 2: We then prepare the MPTCP kernel code for Raspberry Pi by merging Raspberry Pi's Linux kernel branch and a branch of the MPTCP kernel (i.e., 4.19 and 0.95 in our implementations). Thanks to the supportive feature of git, we can combine the two kernels using git clone, git remote add, git fetch, and git merge. The first command is for downloading the Raspberry Pi kernels from [28]. The second and third ones are for adding and updating the MPTCP kernel from [18]. Finally, the last command achieves the merging task. In this step, typically, there have been many conflict errors caused by the different software and Raspberry Pi versions. Hence, we have to debug and fix all of them. After resolving all of the errors, we have the fresh MPTCP kernel code for Raspberry Pi.

Step 3: In this step, we construct the configuration file for kernel compiling. Depending on the targeted Raspberry Pi, we need to set up the correct parameters for the build configuration. The two parameters for kernel and the make option are presented in Table 2, where RPi3, RPi4 stand for Raspberry Pi 3, 4, respectively. First, we configure the default parameters of the MPTCP kernel, such as supported congestion controls, default congestion control, scheduler, path management, etc. Second, we merge the MPTCP configuration file with the default one of Raspberry Pi Linux.

Step 4: We start to compile the kernel (zImage), modules, and Device Tree blobs (dtbs) with the above configuration. After that, we install all of them by using make modules_install. This step takes a long time; we can use the "-j" option to allocate the number of cores for speeding up.

Step 5: After a successful compilation, we copy the kernel, modules, and dtbs to an SD card that will be mounted on a Raspberry Pi device.

3.3 Routing and operational mode

Since we use two Wi-Fi interfaces to connect the IoT router to a server, the router has at least two IP addresses (i.e., received from the DHCP servers on Access Points (APs)). If the IoT router sends data according to the routing based on the destination address, the MPTCP

Table 2 Configuration for different Raspberry Pis

	RPi3	RPi4
KERNEL	kernel7	kernel7l
make option	bcm2709_defconfig	bcm2711_defconfig

communication using the two paths will not be possible. Therefore, the router's routing layer needs to be configured to support the transport layer's MPTCP operation. We use policy routing or source routing to route IP packets following the source address in this case. More specifically, a routing table is set for each IP address of the IoT router. The tables guarantee the next hop of the packet with specific IP is the LAN interfaces of APs.

In the default operational mode, the MPTCP router uses the all available IP address to form a full mesh of paths to a server's destination IP address (a pair of IPs determine a path). After that, the MPTCP path manager, scheduler, and congestion control will control the communication data to communicate over the multiple paths. Besides, MPTCP has supported several other modes in which a subset of paths is used. One of the efficient modes is the backup mode, where a formed path is not used for data communication. Instead, it is in an idle state, which then transforms to an active one if the main path fails. To realize the backup mode, we install the iptool-mptcp package [13], which allows changing the interface mode on MPTCP IoT routers.

4 Evaluation

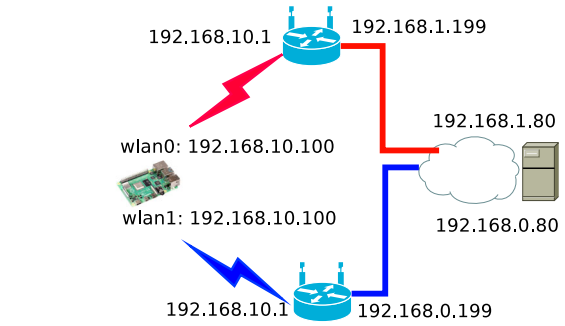
This section introduces the evaluation of the MPTCP-based IoT routers in a static and mobility scenario compared to TCP on the same hardware. In the former, we show the throughput aggregation, resilience enhancement; in the latter, the seamless communication has been evaluated. Finally, we compare the power consumption.

4.1 Experimental setup

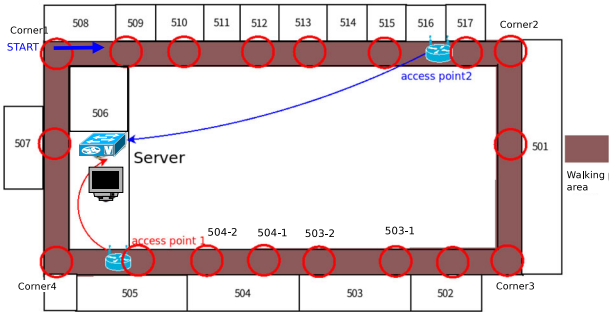
We evaluate two MPTCP IoT routers on Raspberry Pi 3B+ and 4 (i.e., RPi3 and RPi4) in a stationary and mobility scenario. First, we describe the network connection in the static case, shown in Fig. 1a. In the figure, the leftmost is an MPTCP IoT router with two Wi-Fi interfaces: the onboard Wi-Fi card and the USB IO-DATA WN-AG300U. The two Wi-Fi interfaces are associated with two APs, TP-LINK Archer A10. The APs are configured to operate on different Wi-Fi channels to minimize the interferences. Table 3 shows the channel settings. The APs assign IP addresses for the IoT router's interfaces via DHCP in operation. As a result, there are two forming paths (i.e., path1, path2) for the communication between the IoT router and the server. Figure 1a also shows an example of the IP address setting for RPi4 with two interfaces, wlan0 and wlan1. In the RPi3's case, only the last octets in the two IP addresses on the router's Wi-Fi interfaces are different. As mentioned in

Section 3.3, the policy routing is correctly set on each router. More specifically, we create two routing tables (Tables 1, 2) for source routing of the Wi-Fi interfaces, as shown in Table 4. In the static experiments, MPTCP is configured with the normal operation mode, which uses the fullmesh path manager, the lowest RTT scheduler, and the balia congestion control.

Second, we present the network topology in the mobility scenario. We use all the same devices as in the static evaluation. However, their locations are different, as shown in Fig. 1b. The figure illustrates the fifth floor of the first engineering building of Chiba University, Japan. The numbers (i.e., 501, ..., 517) represent the floor's rooms. The access points are far from each other on the floor's two sides, aiming to isolate their coverage. We initially find the correlation between link quality and TCP throughput over each access point at the 18 red circles in Fig. 1b. Note that we use the iperf3 software in all evaluations, which is a popular tool to assess network throughput [12]. In each experiment, an iperf3 client on the IoT router



(a) Stationary



(b) Mobility

Fig. 1 Evaluation scenarios

sends TCP traffic to the iperf3 server installed on the server in 30 seconds. Regarding the link quality evaluation, we use Raspberry Pi 4 and assert the RSSI values reported by the Wi-Fi driver. With RSSI, we can calculate the link quality following the below equation.

$$\text{Link Quality}(\%) = \begin{cases} 100 & (\text{RSSI} \geq -40) \\ \frac{110+\text{RSSI}}{70} * 100 & (\text{RSSI} < -40) \end{cases} \quad (1)$$

After calculation, we plot the link quality of two Wi-Fi links at 18 locations in Fig. 2a. It is clear that the link quality values vary following the distances to the APs. We then investigate the associated TCP throughput at each location to find the correlation between throughput and quality. The measured throughput values are shown in Fig. 2b. We can see that, in general, the better link quality gives the higher throughput value. Moreover, even though the associations with APs exist in several locations (indicated by the link quality), the TCP throughput over one access point may reach zero. Therefore, we use the MPTCP backup mode to exploit the capability of achieving seamless communication of the IoT routers.

Table 3 Wi-Fi parameters

Access Point	Standard	Channel
AP1	802.11n	1
AP2	802.11n	6

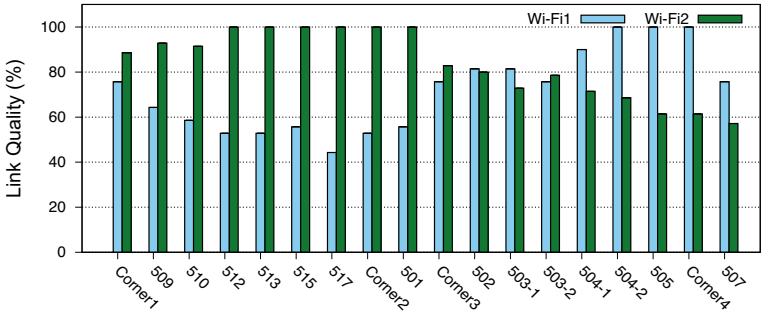
Table 4 Policy routing

	Routing information
Table 1	default via 192.168.10.1 dev wlan0
	192.168.10.0/24 dev wlan0 scope link
Table 2	default via 192.168.11.1 dev wlan1
	192.168.11.0/24 dev wlan1 scope link
Default gateway	default via 192.168.10.1 dev wlan0
	default via 192.168.11.1 dev wlan1 metric 10

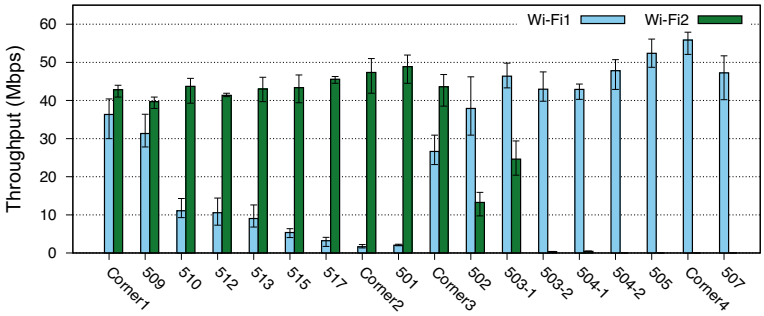
4.2 Static scenario

4.2.1 Throughput evaluation

This evaluation compares the TCP performance over each path and our MPTCP implementations on RPi3 and RPi4. In the evaluation, TCP used the cubic congestion control while MPTCP used the balia one. We repeated each experiment ten times. After that, we tracked and calculated the average, minimum, and maximum throughput values. The results are shown in Fig. 3a, where TCP1 and TCP2 stand for the throughput performance over path1 and path2, respectively. The first observation is that the TCP throughput values of path1 and path2 are different on both devices. The inequality may be caused

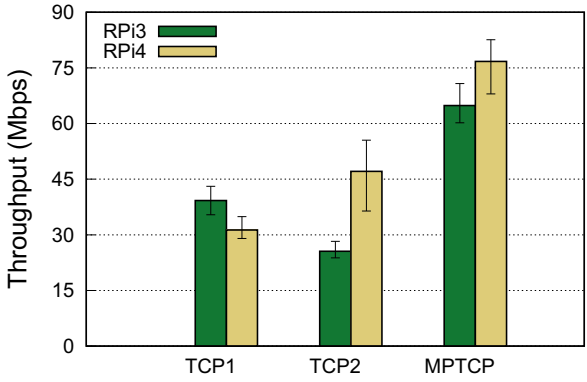


(a) Link quality at different locations

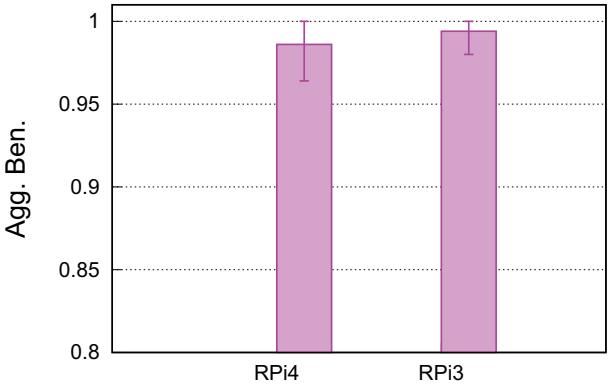


(b) Throughput at different locations

Fig. 2 Thoughput and link quality investigation



(a) Throughput comparison of TCP and MPTCP



(b) Benefit function of MPTCP

Fig. 3 Throughput and benefit function

by the difference between Wi-Fi cards (onboard vs. USB). Moreover, we can see that the RPi3 and RPi4 have different TCP throughputs on the same path. Notably, RPi3's onboard card gives better performance than RPi4's one. Both RPi3 and RPi4 have a similar Cypress CYW43455 Wi-Fi chipset and the firmware BCM4345/6 version 7.45.154. Therefore, the difference in performance may be caused by integrating hardware components into each board. However, regardless of the hardware, our MPTCP kernels always have aggregated throughput over two paths. In the RPi4 case, the TCP1's average throughput is about 31.31 Mbps, and TCP2's one is about 47.1 Mbps. Meanwhile, the MPTCP's average throughput is approximately 76.75 Mbps, close to the sum of TCP1 and TCP2. In the RPi3 case, the three values are 39.23 Mbps, 25.58 Mbps, and 64.86 Mbps, respectively. Therefore, we can conclude the MPTCP IoT routers can achieve throughput aggregation.

We have further looked at the aggregate benefit function of RPi4 and RPi3 to get more insights into the results. The benefit function is a numerical value to show how much a multipath communication benefit is, compared to its related single path ones [15]. The function is expressed by

$$Ben(M) = \begin{cases} \frac{G - C_{\max}}{\sum_{i=1}^{n} C_i - C_{\max}}, if G \geq C_{\max} \\ \\ \frac{G - C_{\max}}{C_{\max}}, if G \leq C_{\max} \end{cases}, \quad (2)$$

where Ci is the TCP throughput of path i; C max is the highest value of all TCP throughput; G is the throughput of MPTCP; M is the number of paths. $Ben(M)$ is the MPTCP benefit function, whose values is in the range [-1:1]. When the value of $Ben(M)$ is larger than 0, MPTCP has higher performance than TCP, or MPTCP has its aggregation benefit. On the other hand, if the value is smaller than 0, it indicates that there is a single path with higher throughput than MPTCP. $Ben(M)$ allows comparing not only the aggregation benefit of MPTCP against TCP but also different MPTCP aggregation scenarios. Figure 3b shows the $Ben(M)$ values of RPi4 and RPi3 in our experiments. The y-axis shows the average, minimum, and maximum values in the figure. We can confirm the early findings of the throughput aggregation since, in both cases, the $Ben(M)$ values are all bigger than 0 and close to 1. Moreover, we can see that RPi3 achieves the aggregation duty slightly better than RPi4 does.

4.2.2 Path failure

Since TCP can not handle the path failure case, we exclude TCP in this evaluation. Note that the evaluation in this section has been conducted independently from the previous section. We investigated the behavior of MPTCP on RPi4 and RPi3 in path failure scenarios, which happened during the 30-second iperf3's MPTCP communication as follows. We created a path failure event by disconnecting the cable on path2 at the fifth second after the beginning of an experiment. About fifteen seconds later, we issued a recovery event by reconnecting the cable. The results of MPTCP throughputs variation on RPi4 and RPi3 are shown in Fig. 4a and b, respectively.

The two figures show that MPTCP achieves the throughput aggregation until the disconnection moment (i.e., the sight of high throughput). After that, the MPTCP throughput was reduced from the aggregated level to the TCP1's one. That is because all the data is changed from transmissions on two paths to a single one (path1). After the reconnection, both the IoT routers can recover the communication from one path to two paths. When there is a reachable IP address, the adding subflow function of MPTCP achieves that (i.e., reestablishing TCP handshake for subfow on path2). In these experiments, the iperf3 flows continue for 30 seconds without zero throughputs, although both experience failures. Therefore, we conclude the MPTCP-based IoT routers achieve seamless communication against path failures.

4.3 Evaluation of mobility scenario

This evaluation aims to investigate the IoT routers with MPTCP in backup mode, in which one Wi-Fi link is used to transmit MPTCP packets while the other is idle for the MPTCP traffic. When the primary link is broken, the backup link will be active and convey the ongoing MPTCP data. In the evaluation, we let the IoT routers move around the floor at a walking person's speed while keeping the iperf3 running continuously. Each round of movement was around 150 seconds. We compared three mobility cases: MPTCP with backup mode, only TCP via AP1 (TCP1), and only TCP via AP2 (TCP2). The evaluation results for RPi3, RPi4 are shown in Fig. 5a, b, respectively.

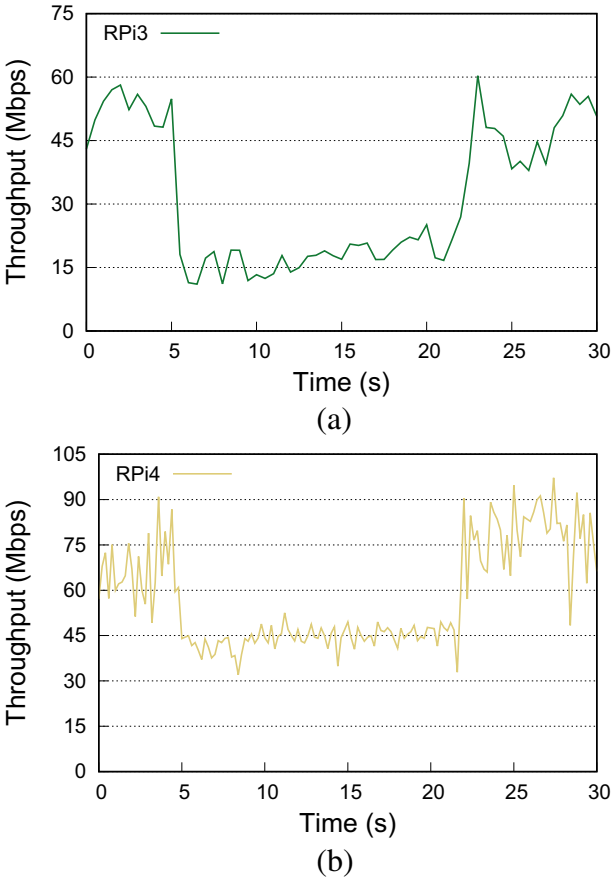
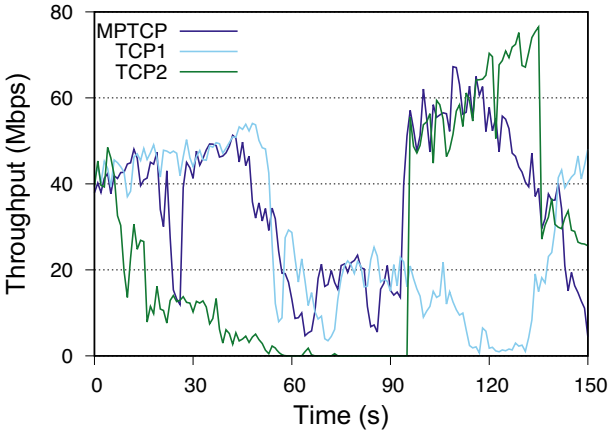


Fig. 4 MPTCP throughput variation in path failure scenarios

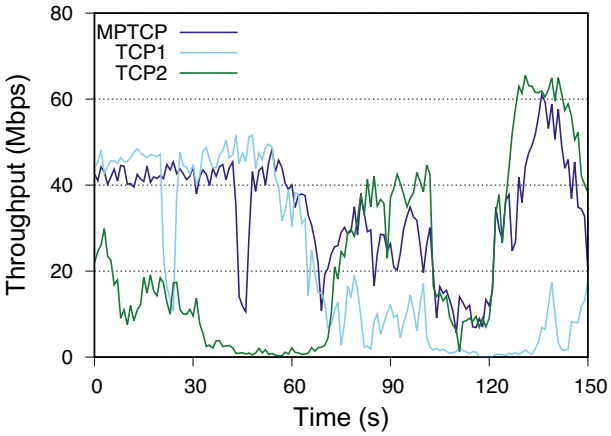
As shown in the two figures, one access point's coverage could not cover the whole floor in the TCP cases. For example, in Fig. 5a, during 60-90 seconds, the TCP2 throughput was zero, and the TCP1 became significantly low during 110-130 seconds. However, there was always one path with good conditions during mobility in all cases. It suggests using such a path for data transmission to avoid the low-quality Wi-Fi link's effects. It is precisely the operation of the MPTCP IoT routers which can handle the situations by turning on and moving the traffic to a better path. We can see that RPi3 and RPi4 maintained seamless communication to the server without disruptions in both figures.

4.4 Power consumption evaluation

Energy efficiency is a critical issue in IoT. Hence, it is necessary to investigate the power consumed by our MPTCP kernels on RPi3 and RPi4. In the evaluation, similar to those previously done, we compared the power consumption of MPTCP to TCP1's and TCP2's. Note that, in this case, we used the default MPTCP mode and the static scenario. In each experiment, different from the previous ones, we used iperf3 to send a 1G dummy file to



(a) RPi3



(b) RPi4

Fig. 5 Throughput variation in mobility scenario

the server. We used a USB current-voltage checker [4] to measure the voltage and current values on the IoT device during the data transfer period for each traffic type. We then calculated the power consumption from those values. Figure 6 shows the average, maximum, and minimum power consumption after ten runs.

In the figure, we can see that TCP1 has lower values than TCP2 in both RPi3 (43.718 Joule vs. 44.968 J) and RPi4 (37.08 J vs. 41.062 J). That means the onboard Wi-Fi card of Raspberry Pi consumed less power than the USB attached one. Moreover, RPi4 was more energy-efficient than RPi3 in all cases with TCP and MPTCP. That is understandable since RPi4 is a newer model with many improved features. Moreover, the most important observation is that MPTCP has the lowest power consumption values on each router. Although using both two Wi-Fi links, the 1G-file transfer in MPTCP finished much sooner than in TCP on a single path.

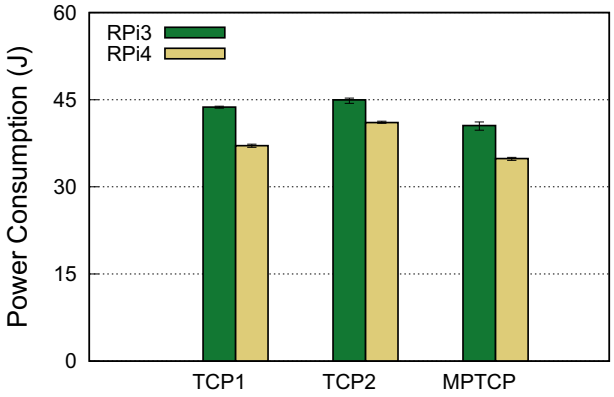


Fig. 6 Power consumption comparison

5 Conclusion and future works

This paper addresses the necessity of an IoT router with multiple wireless interfaces that can form multipath communication to an application server. We first build a new MPTCP kernel for IoT devices, which can operate well on Raspberry Pi3B+ and 4. We have then implemented and evaluated the MPTCP-based IoT routers, which use the MPTCP kernels and two Wi-Fi interfaces in the static and mobility scenarios. In the static scenario, the MPTCP routers can achieve throughput aggregation and seamless handover. More specifically, in the former investigation, the MPTCP throughput is approximately the sum of the two single-path TCP throughputs. In the latter, the MPTCP router can maintain the communication in the path failure scenario. We let the MPTCP IoT routers move around with the backup mode in the mobility scenario and evaluate the throughput variation. The results show that MPTCP can shift the traffic to a better link when the ongoing communication link has lower quality. Moreover, the power consumption results indicate the efficiency of MPTCP against TCP. MPTCP is more energy-efficient than TCP when sending the same amount of data.

In the future, we will investigate the performance of other MPTCP components such as scheduler, path manager, and congestion control on the IoT routers. Moreover, we plan to add and explore different wireless technologies to the router. We will also investigate the performance variation in the mobility scenario with more APs and the router's different operation mode.

Funding This work was supported in part by the Japan Society for the Promotion of Science (JSPS) under Grant 20H0417 and in part by the Japan Science and Technology Agency (JST) through the establishment of university fellowships towards the creation of science technology innovation, Grant Number JPMJFS2107.

Declarations

Conflict of Interests The authors have no conflicting/competing interests to declare that are relevant to the content of this article.

References

1. Aloi G, Caliciuri G, Fortino G, Gravina R, Pace P, Russo W, Savaglio C (2017) Enabling iot interoperability through opportunistic smartphone-based mobile gateways. J Netw Comput Appl 81:74–84

2. Apple Inc (2020) Improving network reliability using multipath TCP. https://goo.gl/MDQZFh. Accessed Feb 2020
3. Bimschas D, Hellbrück H, Mietz R, Pfisterer D, Römer K, Teubler T (2010) Middleware for smart gateways connecting sensornets to the internet. In: Proc International workshop on middleware tools, services and run-time support for sensor networks, pp 8–14
4. COOWOO Power Meter (2021) http://www.coowootech.com/jp/. Accessed Oct 2021
5. Cirillo F, Gómez D, Diez L, Elicegui Maestro I, Gilbert TBJ, Akhavan R (2020) Smart city iot services creation through large-scale collaboration. IEEE Internet Things J 7(6):5267–5275
6. Datta SK, Bonnet C, Nikaein N (2014) An iot gateway centric architecture to provide novel m2m services. In: Proc. IEEE world forum on internet of things (WF-iot), pp 514–519
7. Ford A, Raiciu C, Handley M, Bonaventure O (2013) TCP Extensions for multipath operation with multiple addresses. RFC, 6824
8. FreeBSD (2018) Multipath TCP for freeBSD. http://caia.swin.edu.au/newtcp/mptcp/ Accessed Oct 2021
9. García L, Jiménez JM, Taha M, Lloret J (2018) Wireless technologies for iot in smart cities. Netw Protoc Algorithms 10:23–64
10. Guoqiang S, Yanming C, Chao Z, Yanxu Z (2013) Design and implementation of a smart iot gateway. In: Proc IEEE international conference on green computing and communications and IEEE internet of things and IEEE cyber, physical and social computing, pp 720–723
11. Hao C, Xueqin J, Heng L (2011) A brief introduction to iot gateway. In: Proc IET ICCTA, pp 610–613
12. Iperf3 (2021) https://iperf.fr. Accessed Oct 2021
13. Iproute-mptcp (2021) https://github.com/multipath-tcp/iproute-mptcp. Accessed Oct 2021
14. Kang B, Kim D, Choo H (2017) Internet of everything: a large-scale autonomic iot gateway. IEEE Transactions on Multi-Scale Computing Systems 3(3):206–214
15. Kaspa D (2011) Multipath Aggregation of Heterogeneous Networks. PhD thesis University of Oslo
16. Khalili R, Gast N, Popovic M, Le Boudec J-Y (2013) Mptcp is not pareto-optimal: Performance issues and a possible solution. IEEE/ACM ToN 21(5):1651–1665
17. Kibria MG, Nguyen K, Villardi GP, Ishizu K, Kojima F (2018) Next generation new radio small cell enhancement: Architectural options, functionality and performance aspects. IEEE Wirel Commun 25(4):120–128
18. MPTCP Linux kernel (2021) https://github.com/multipath-tcp/mptcp. Accessed Oct 2021
19. MPTCP kernel for Android (2021) http://multipath-tcp.org/pmwiki.php/Users/Android. Accessed Oct 2021
20. Mondal A, Kabbinale AR, Shailendra S, Rath HK, Pal A (2018) Ppos: a novel sub-flow scheduler and socket apis for multipath tcp (mptcp). In: Proc IEEE twenty fourth national conference on communications (NCC), pp 1–6
21. Ndiaye M, Oyewobi SS, Abu-Mahfouz AM, Hancke GP, Kurien AM, Djouani K (2020) Iot in the wake of covid-19: A survey on contributions, challenges and evolution. IEEE Access 8:186821–186839
22. Nguyen K, Golam Kibria M, Ishizu K, Kojima F, Sekiya H (2019) An approach to reinforce multipath tcp with path-aware information. Sensors 19(3)
23. Nguyen K, Ishizu K, Kojima F (2017) An evolvable, scalable, and resilient control channel for software defined wireless access networks. Computers & Electrical Engineering 57(Supplement C):104–117
24. Nguyen K, Kibria MG, Ishizu K, Kojima F (2019) Enhancing multipath tcp initialization with syn duplication. IEICE Trans Commun E102.B(9):1904–1913
25. Nguyen K, Kibria MG, Ishizu K, Kojima F (2019) Performance evaluation of ieee 802.11ad in evolving wi-fi networks. Wirel Commun Mob Comput 2019(Article ID 4089365):11
26. Peng Q, Walid A, Hwang J, Low SH (2015) Multipath tcp: analysis, design, and implementation. IEEE/ACM ToN PP(99)
27. Raiciu C, Paasch C, Barre S, Ford A, Honda M, Duchene F, Bonaventure O, Handley M (2012) How hard can it be? designing and implementing a deployable multipath TCP. In: Proc. USENIX NSDI, pp 399–412
28. Raspberry Pi Linux kernel (2021) https://github.com/raspberrypi/linux. Accessed Oct 2021
29. Saxena N, Roy A, Sahu BJR, Kim H (2017) Efficient iot gateway over 5g wireless: a new design with prototype and implementation results. IEEE Commun Mag 55(2):97–105
30. Wischik D, Raiciu C, Greenhalgh A, Handley M (2011) Design, implementation and evaluation of congestion control for multipath tcp. In: Proc. USENIX NSDI, pp 99–112
31. Xiaojun C, Xianpeng L, Peng X (2015) Iot-based air pollution monitoring and forecasting system. In: Proc IEEE international conference on computer and computational sciences (ICCCS), pp 257–260
32. Zhu Q, Wang R, Chen Q, Liu Y, Qin W (2010) Iot gateway: Bridgingwireless sensor networks into internet of things. In: Proc IEEE/IFIP international conference on embedded and ubiquitous computing, pp 347–352

33. Zielonka A, Sikora A, Woźniak M, Wei W, Ke Q, Bai Z (2021) Intelligent internet of things system for smart home optimal convection. IEEE Trans Indust Inform 17(6):4308–4317