TU Delft

Delft University of Technology

Evaluation of the Bridge Architecture

Kashyap, Shruthi; Rao, Vijay; Venkatesha Prasad, Ranga Rao; Staring, Toine

DOI
10.1007/978-3-030-85836-0_6

Publication date
2021

Document Version
Final published version

Published in
SpringerBriefs in Applied Sciences and Technology

Citation (APA)
Kashyap, S., Rao, V., Venkatesha Prasad, R. R., & Staring, T. (2021). Evaluation of the Bridge Architecture. In *SpringerBriefs in Applied Sciences and Technology* (pp. 63-71). (SpringerBriefs in Applied Sciences and Technology). Springer. https://doi.org/10.1007/978-3-030-85836-0_6

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Chapter 6
Evaluation of the Bridge Architecture

Chapter 5 gave an overview of some of the factors influencing the performance of the bridge architecture and discussed how the standard TCP/IP stack can be adapted to the time-slotted NFC channel. Two major problems related to packet drops and spurious retransmissions were identified as the major contributors to the system latency. They were solved by introducing an NFC channel sensing mechanism and a new way of estimating and updating the TCP RTO values. This chapter contains the verification results of these solutions. Additionally, this chapter provides some recommendations for implementing the bridge architecture.

6.1 Implementation Recommendations

Certain use-case scenarios that are encountered while implementing the bridge architecture are listed below. Possible methods to handle/implement such scenarios are also briefly described. This research, however, does not consider these scenarios while evaluating the performance of the bridge architecture.

1. **Non-identical NFC buffer and MTU sizes in PTx and appliance:**
   The appliances and the PTxs may have different versions of software implementations, and they could be from different manufacturers. So it is not necessary that the uplink and downlink characteristics of the communication channel between the two will be the same.
   The PTx and appliance may have different buffer sizes in their NFC modules. Before starting a TCP connection, it is necessary to exchange information regarding buffer sizes so that packets with appropriate sizes can be sent without causing buffer overflows. It is also important for the appliance to know the Maximum

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2021

S. Kashyap et al., *Cook Over IP*,
SpringerBriefs in Applied Sciences and Technology,
https://doi.org/10.1007/978-3-030-85836-0_6

Transmission Unit (MTU) size of the PTx. The TCP MSS size can then be adjusted accordingly to prevent packet drops.

2. **Increased communication overhead due to small packet buffer size:**
   The memory allocated for the TCP/IP packets by the stack should be large enough to hold an entire packet with maximum segment size. In the LwIP stack, a single TCP/IP packet is stored in multiple small packet buffers that are chained together. This type of storing increases the overhead in the packet and adds to the latency on the NFC channel.
3. **Upgrading the TCP/IP stack in the end-user devices:**
   The new algorithm proposed for handling the RTO mechanism requires modifications to be made in the TCP/IP stack. This would be easy for the appliance because its stack needs to support only the NFC-enabled kitchen applications. On the contrary, the end-user device cannot readily make these changes as its TCP/IP stack is shared by various other applications. The stack needs to be upgraded with the new algorithm such that it dynamically supports all types of applications and channels.
   As explained in Sect. 5.3.2.2, the algorithm sets the RTO of the packets by considering the NFC transmission rate, packet size and observing the delay on the Ethernet/Wi-Fi channel. Similarly, this method could also be used for applications that do not involve NFC channels. The stack can study the channel delay by constantly measuring the RTT of the packets and use this to calculate the delay experienced per byte on the channel. It can then set the RTO of the packets using the current packet size and the delay per byte factor. A better RTO estimation can be achieved with this method which would help in avoiding spurious and delayed retransmissions especially in high delay, low bandwidth channels. By upgrading the TCP/IP stack with this algorithm, it can dynamically adapt itself to different channels and support a wide variety of applications with improved performance.

6.2 Results

The performance of the bridge architecture is evaluated by carrying out various experiments with different NFC bit rates and data sizes. The performance is analyzed by measuring latency, throughput, number of retransmissions in the TCP sessions, NFC channel bandwidth utilization, etc.

6.2.1 Packet Retransmissions

Tables 6.1 and 6.2 show the number of retransmissions, DUP ACKs and keep-alive messages in the TCP session after using the mitigation techniques Sects. 5.3.1 and 5.3.2 described in Chap. 5, at 11.2 kbps and 24 kbps, respectively. The retransmitted

Table 6.1 Number of retransmissions at 11.2 kbps

NFC payload size (Bytes)	Retransmissions in TCP session			
	Original TCP/IP configuration	NFC channel sense	Optimum TCP RTO	NFC channel sense + Optimum TCP RTO
250	1R	1R + 1DA	1R	0R
500	2R	2R + 1DA + 2KA	1R	0R
1000	3R + 1DA	3R + 2DA + 2KA	1R	0R
1080	3R + 1DA	3R + 2DA + 2KA	1R	0R

Table 6.2 Number of retransmissions at 24 kbps

NFC payload size (Bytes)	Retransmissions in TCP session			
	Original TCP/IP configuration	NFC Channel sense	Optimum TCP RTO	NFC Channel sense + Optimum TCP RTO
250	1R	0R	1R	0R
500	1R + 1DA	0R	1R	0R
1000	1R	2R + 1DA + 2KA	1R	0R
1080	3R + 1DA	2R + 1DA + 2KA	1R	0R

packets are depicted by the symbol 'R', DUP ACKs by 'DA' and keep-alive packets by 'KA'. The experiments are carried out with TCP sessions exchanging single packets with NFC payload sizes of 250 bytes, 500 bytes, 1000 bytes and 1080 bytes at 11.2 kbps and 24 kbps.

The technique Sect. 5.3.2 introduced is to remove the spurious retransmissions by setting optimum initial RTO values for all the outgoing TCP/IP packets. Tables 6.1 and 6.2 show that by using only this solution, the total number of retransmissions can be brought down to one. The technique in Sect. 5.3.1 is an NFC channel sensing mechanism introduced to avoid packet drops at the NFC interface. As shown in the tables, using only technique in Sect. 5.3.1 the total number of packets increases compared to the respective original TCP sessions in most of the cases. However, when both these techniques are used together, all types of retransmissions, DUP ACKs and keep-alive packets are removed. Before concluding on the performance based on the number of packets in the TCP session, it is important to study the latency of the session, which is done in Sect. 6.2.2.

Figure 6.1 depicts the RTO values estimated by the new algorithm (Sect. 5.3.2.2) in a long TCP session with randomly varying packet sizes. These values are compared with the ones estimated by VJ's algorithm used in the LwIP stack and the packet RTT values obtained over an Ethernet channel with <1 ms delay. The estimations are, however, still carried out considering the Wi-Fi channel characteristics with a

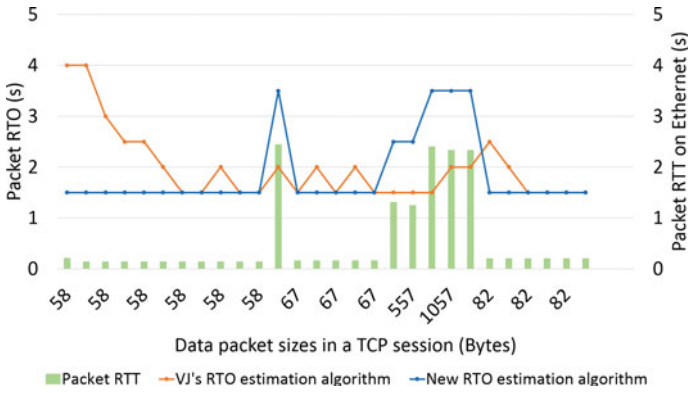

Fig. 6.1 Comparison of packet RTO values with the new and VJ's RTO estimation algorithms

minimum RTO of 1 s, which results in an offset of about 1 s between the measured RTT and the estimated RTO values as seen in Fig. 6.1. The new algorithm clearly gives a more accurate estimation of the RTO values compared to VJ's algorithm as it takes the packet sizes and bit rates of the channels into account, therefore avoiding all the spurious and delayed retransmission scenarios.

6.2.2 Latency

Reduction in the number of packets in the TCP session need not necessarily improve the latency of the session. This is because the time delay between packet generation, especially in the case of retransmitted packets, is also an important factor that affects the overall latency. Figures 6.2 and 6.3 show the graphs of latencies of TCP sessions with and without the mitigation techniques in Sects. 5.3.1 and 5.3.2 at 11.2 kbps and 24 kbps, respectively.

The percentage by which the latencies increase or decrease using the mitigation techniques compared to the original latency is indicated in the graphs. At lower NFC bit rates, for example, 11.2 kbps, the TCP session latency with only technique in Sect. 5.3.1 becomes higher than that with only technique in Sect. 5.3.2 when there are more number of retransmissions/DUP ACKs/keep-alive messages. This is because even though the packet drops are prevented, there will be too many extra packets to be transmitted over a low bandwidth channel. On the contrary, at higher bit rates like 24 kbps, the latency with only technique Sect. 5.3.1 will be lower than that with only technique Sect. 5.3.2 because when only technique in Sect. 5.3.2 is used, the time delay created by retransmission caused due to packet drop will be more significant compared to the packet transmission time on a relatively higher bandwidth channel. The TCP/IP stack has to wait for the timeout to realize that the packet is dropped and

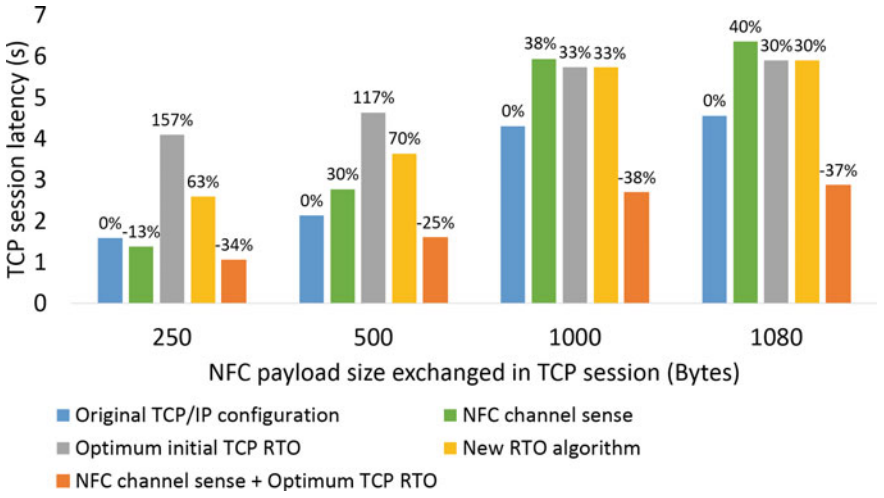

Fig. 6.2 Latencies of TCP sessions at 11.2 kbps

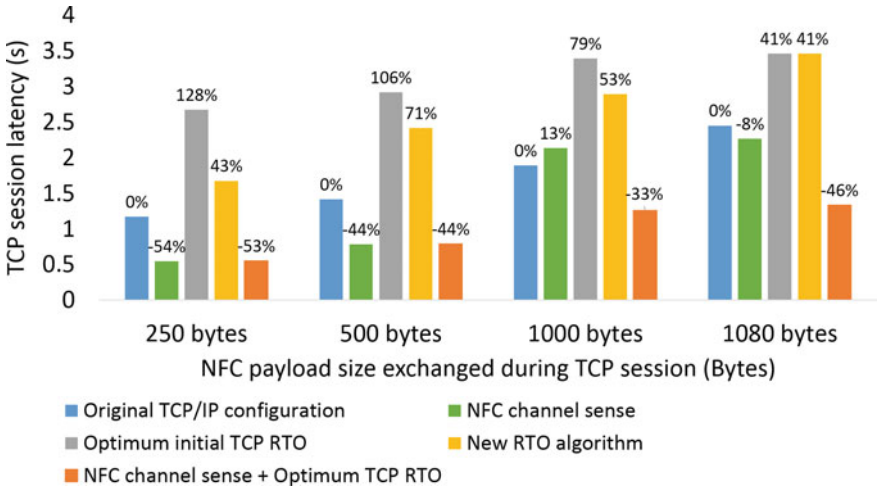

Fig. 6.3 Latencies of TCP sessions at 24 kbps

resend it. This waiting time will be longer compared to the time taken to transmit extra packets.

To achieve the best results, it is recommended to use both the mitigation techniques together. Using both, up to 38% reduction in latency can be achieved at 11.2 kbps and up to 53% at 24 kbps. Higher reduction is achieved at higher bit rates because of the same reason explained above. At higher bit rates, the time delay created because of packet drops will be more significant when compared to the total transmission

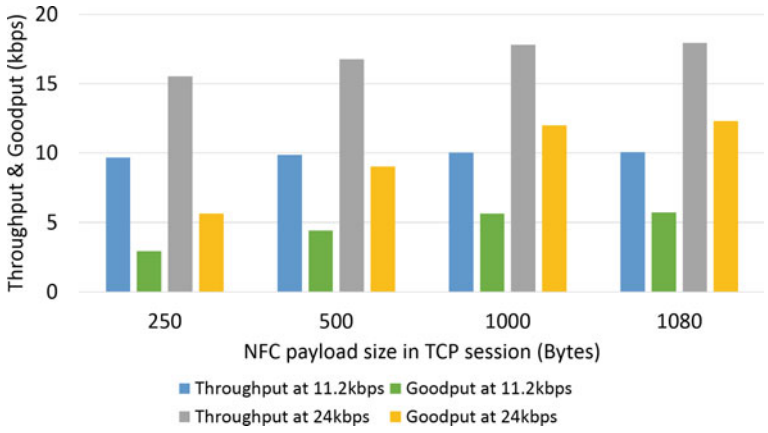

Fig. 6.4 System throughput at 11.2 and 24 kbps

time. So by removing this delay which is a bigger overhead, a higher gain in latency reduction can be achieved.

6.2.3 Throughput and Goodput

The throughput of the system remains the same with or without the retransmission mitigation techniques in Sects. 5.3.1 and 5.3.2. It is known that the techniques are used to reduce the latency, however, the reduction in latency is achieved by reducing the number of packets or bytes traveling through the channel. Therefore, the throughput, which is the number of bytes transferred per unit time, will be unchanged because with the mitigation techniques less packets/bytes travel through the channel which takes less time. So the overall throughput of the system technically remains constant.

Figure 6.4 depicts the throughput versus goodput of the system for different NFC payload sizes exchanged in the TCP session using both techniques in Sects. 5.3.1 and 5.3.2 at 11.2 kbps and 24 kbps. On an average, the throughput is 9.9 kbps at an NFC bit rate of 11.2 kbps and 17.01 kbps at 24 kbps. It can be seen that the goodput of the system improves with an increase in the payload size. This is because the overheads from the TCP/IP header and UI protocol become less significant with an increase in payload size. Choosing a bigger TCP MSS size will help in increasing the goodput of the system.

The throughput is lower for TCP sessions with small payload sizes, and it gradually increases with the increase in payload size. This is because with small payload sizes, the time spent waiting for a time slot will be significant compared to the packet transmission time. At higher bit rates, this becomes more noticeable because the transmission time will be even smaller. This affects the total transmission time and thus the throughput of the TCP session. The throughput could be improved by

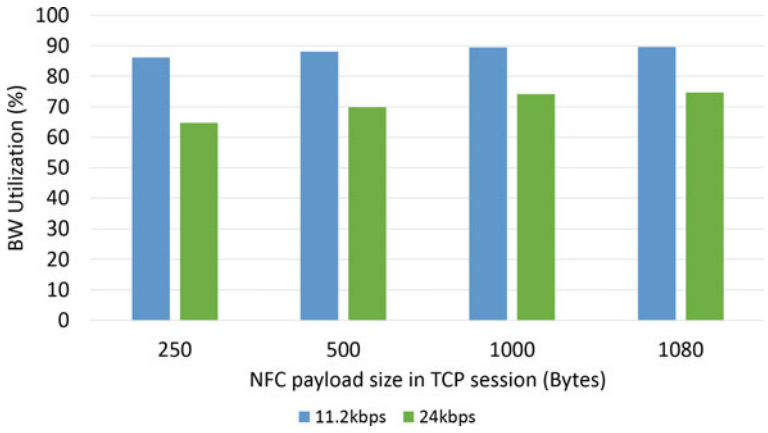

Fig. 6.5 Bandwidth Utilization at 11.2 and 24 kbps

1. using TCP/IP header compression techniques such as [1, 2].
2. employing the 6LoWPAN technology for the compression of TCP/IP packets over NFC as described in [3, 4].
3. letting the PTx detect and filter out the spuriously retransmitted packets and DUP ACKs coming from the appliance and the end-user device, similar to the technique proposed in [5]. This would reduce the number of packets on the NFC channel and improve the system performance.
4. modifying the NFC protocol in order to optimize the NFC handshake sequence as suggested in [6].

6.2.4 Bandwidth Utilization

The bandwidth utilization of the NFC channel for the experiments performed is illustrated in Fig. 6.5. It is calculated using the following equation:

$$BW\ Utilization = \frac{Throughput}{Theoretical\ BW} * 100\ (\%) \tag{6.1}$$

The average bandwidth utilization is found to be 88.4% at 11.2 kbps and 70.89% at 24 kbps. Lower bandwidth utilization is observed at higher bit rate because the processing time which includes packet transmission time over the UART and Wi-Fi/Ethernet channels, packet processing time by the stack, etc. remains constant irrespective of the NFC bit rate. This processing time overhead will be more significant at higher bit rates because it has smaller NFC transmission time. When Figs. 5.18 and 6.6 are compared, it can be seen that the UART Data Tx time is the same at both 11.2 kbps and 24 kbps, which is a fixed overhead. However, NFC Data Tx time is

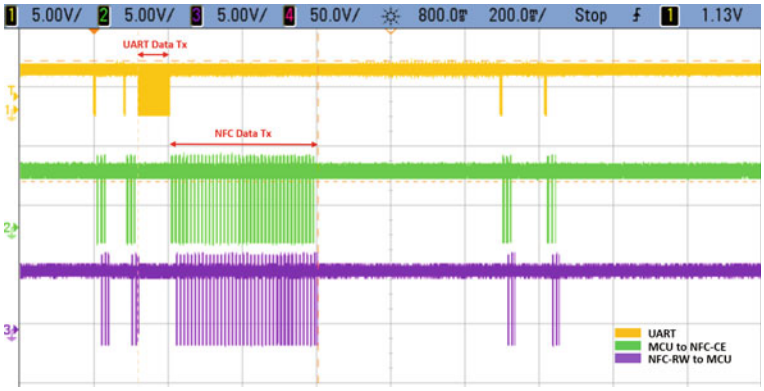

Fig. 6.6 TCP session capture in the direction from the appliance through the NFC-CE and NFC-RW modules at 24 kbps

smaller at 24 kbps compared to 11.2 kbps. This keeps the NFC channel idle for a longer period at a higher bit rate, thus reducing the bandwidth utilization. The main factors affecting the NFC bandwidth utilization of this system are

1. Packet processing time: The NFC channel remains idle while the TCP/IP packet is being processed and transferred over the UART from the stack to the NFC module.
2. Synchronization of data transfer with the communication time slot: The packet arrival time at the MCU can lie anywhere between two consecutive time slots. As explained earlier, the NFC module requires the packet to be available for transmission at least 2 ms before the time slot occurs. This may result in a waiting time of up to 12 ms for every packet (assuming that the subsequent chunks arrive on time), which adds to the total packet transmission time.

Some ways to improve bandwidth utilization are listed below. These techniques could not be tested due to the limitations in the available hardware.

1. Parallelizing the packet processing and packet transmission operations;
2. Increasing bit rate of serial communication (UART);
3. Eliminating the MCUs and directly interfacing the appliance and the PTx stacks to their respective NFC devices. This will reduce the processing delay caused by the serial communication.

References

1. V. Jacobson, Compressing TCP/IP headers for low-speed serial links. RFC **1144**, 1–49 (1990)
2. M. Degermark, M. Engan, B. Nordgren, S. Pink, Low-loss TCP/IP header compression for wireless networks. MobiCom '96 (1996)

3. J. Youn, Y. Hong, D. Kim, J. Choi, Y. Choi, Transmission of IPv6 packets over near field communication (2000)
4. J. Park, S. Lee, S. Bouk, D. Kim, Y. Hong, 6LoWPAN adaptation protocol for IPv6 packet transmission over NFC device, in *Seventh International Conference on Ubiquitous and Future Networks* (2015), pp. 541–543
5. Y. Kim, D. Cho, Considering spurious timeout in proxy for improving TCP performance in wireless networks. Comput. Netw. **44**, 599–616 (2004)
6. H. Sakai, A. Arutaki, Protocol enhancement for near field communication (NFC): future direction and cross-layer approach, in *Third International Conference on Intelligent Networking and Collaborative Systems* (2011), pp. 605–610