IP Packet Loss and RTT Calculation Simulation Using Low-Cost Embedded Real-Time Systems

Conference Paper · August 2018

CITATIONS
0

READS
25

2 authors, including:

Damian Valles
Texas State University
16 PUBLICATIONS **3** CITATIONS

SEE PROFILE

IP Packet Loss and RTT Calculation Simulation Using Low-Cost Embedded Real-Time Systems

B. DasGupta[1], **and D. Valles**[1]
[1]Ingram School of Engineering, Texas State University, San Marcos, Texas, USA

Abstract - *Research and development in the field of congestion control in computer networks is involving machine learning. However, the training process can be expensive and complicated because of the need for a pre-existing computer network with which to grade the success of an algorithm. Network simulators such as ns-2 and ns-3 also may not help because they are not real-time simulators. This paper proposes to capture the pattern of data that the neural network is identifying and the embedded system records. The interpolated data with similar patterns are generated and presented to the neural network to test if the trigger has been successfully identified. For the purposes of this paper, the data is a record of delays in multiple TCP packets.*

Keywords: Networking simulation, data interpolation, RTT.

1 Introduction

In computer networks, a major field of study and innovation is in congestion control. When computers communicate with each other on a network, packets are usually sent via networking devices such as switches and routers that act as communication hubs. These networking devices have fast processors and optimized communication buffers that allow them to temporarily hold on to information when they cannot transmit packets as fast as they receive them. When two devices that lie on the same network communicate through a switch, the complexity of routing algorithms are low, and most commercial switches designed for the purpose are more than adequate to handle such tasks. However, when the network architectures complexity increases, or when a large number of clients on a network attempt to communicate large amounts of data with each other simultaneously, these switches and routers may run out of buffer space and become choke points. This, in turn, means that switches and routers are usually the bandwidth choke points in networks of networks. Because of this inherent problem, the study of congestion control is ever growing.

Microcontrollers can be used to replace the use of real production networks during experimentation. Instead of using the delays inherent to real networks, as shown in Figure 1, a microcontroller can be used to create artificial delays, as shown in Figure 2. The NXP LPC1768 is a microcontroller that uses an ARM M3 Cortex processor, supports Ethernet MAC, has a dedicated Direct Memory Access (DMA) controller, a Serial Peripheral Interface (SPI) controller, two Inter-Integrated Circuit (I2C) ports, and four Universal Asynchronous Receiver-Transmitters (UARTs). The Ethernet MAC controller can be used to connect to a network, the SPI and UARTs can be used for communication between multiple microcontrollers, and the DMA controller can be used for faster message modification and transmission necessary for computer networking experiments.

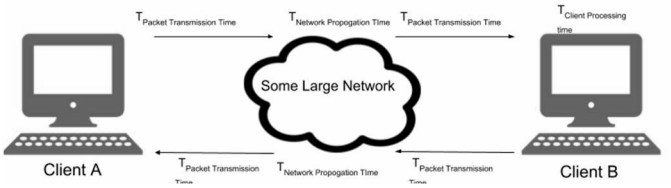

Fig. 1. Round trip time for real networks.

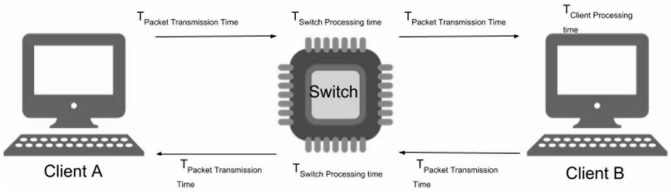

Fig. 2. Simulated Round Trip Time Diagram.

1.1 Edge Devices

Since network switches and routers are so expensive, they are designed to be used for many years. However, the edge devices, the user-controlled devices that lie on the metaphorical edges of a network topology are replaced relatively often. In some cases, consumers upgrade to newer and faster devices yearly. Since any changes made to these devices reach the market faster, it is a convention to apply software and hardware changes more aggressively to these edge devices. Therefore, most congestion control methods, especially better round-trip time calculations are applied to end-user devices and software.

1.2 Motivation

Using Neural Networks (NN) for machine learning in the field of computer networks is certainly not new. Research such as [2] [3] [4] and [5] all focus on training NNs for use with computer networks. Each of the previously listed efforts has its own way of creating the tests that evaluate the success of the NNs. In [2] and [3] train multiple types of machine learning to identify and classify simulated loss causes in an ns-2 environment. However, the different machine learning

algorithms used did not operate in real-time and therefore may or may not be applicable as solutions to real-time problems. Creating a NN that is trained to handle large real-time streams of information requires a fast and inexpensive method of creating real-time data.

2 Procedure

By recording Round trip times (RTT) collected during real network simulations, artificial RTT datasets can be generated using the following equations:

$$S[n] = A[n-x] + \frac{(A[n+x]-A[n-x])}{2x}(n-(n-x)) \quad (1)$$

$$S[n] = A[n-r_1] + r_2 \;\; | \;\; r_1 > 0 \;\; and \;\; 0.5 > r_2 > 0 \quad (2)$$

The artificially created RTT times will be determined by equation (1), a linear interpolation polynomial [1] creating the periods of normal information exchange from A, the originally recorded times, and equation (2) creating the offset in time [1] by variable r1, and the minute differences in the delay with r2.

The microcontroller chosen for this experiment is the NXP LPC1768. As shown in Figure 3, The LPC1768 has many features that make it a desirable option: it uses a power efficient ARM M3 Cortex processor, supports Ethernet MAC, has a dedicated Direct Memory Access (DMA) controller, a Serial Peripheral Interface (SPI) controller, and four Universal Asynchronous Receiver-Transmitters (UARTs). The microcontroller is also cost-effective than larger computational options, making it more accessible and easier to conduction proof-of-concept experiments.

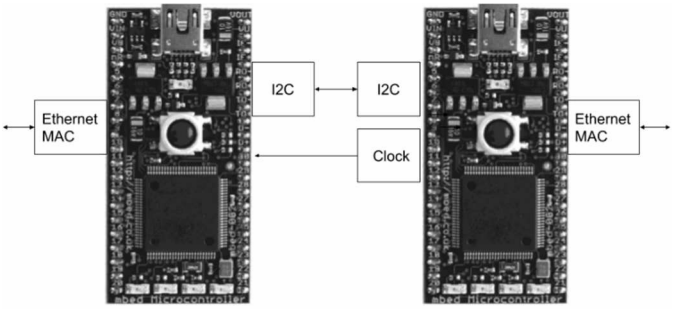

Fig. 3. Two LPC1768s to allow for two Ethernet ports.

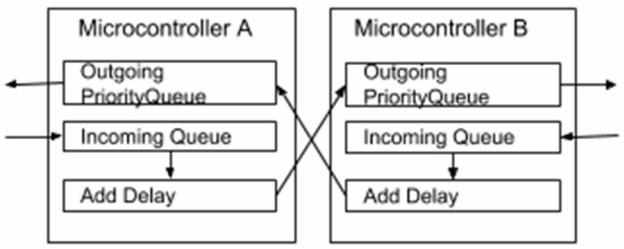

Fig. 4. Communication process.

Since each LPC1768 supports one Ethernet connection, two LPC1768s were used in conjunction as shown in Figure 3. The Serial Clock (SCL) pins and Serial Data (SDA) pins were connected between the two boards to establish an I2C connection, and each board had one Ethernet connection facing out. As shown in Figure 4, each LPC1768 has an incoming message queue and an outgoing message priority queue. When one LPC1768 receives a message, it pulls the next pre-calculated time from a file in memory and sends that data to the other LPC1768, which adds the message to the outgoing message queue based on the delay. To simulate packet loss, a percentage of packets are intentionally not added to the outgoing queue.

The microcontroller then sends the message to the outgoing port if the delay time was met. This way, to the communication endpoints, this micro- controller is invisible, but still creates the delay necessary for neural network testing. Two experiments will be conducted. One experiment will simulate a lossy network similar to a weak wireless signal, where RTT times are slow and 25 percent of packets are lost. The other experiment will simulate a busy, overburdened network, where packets are not lost but RTT times are slow due to congestion.

3 I^2C Process

For the sake of simplicity, I^2C was chosen over SPI as the method of communication between the two devices. Though I^2C has a slower rate of transfer than SPI, the difference is marginal at most. The benefits of SPI, however, far outweighed the cost. Initially, SPI was chosen to connect the two devices. However, minute differences in clock rates and sensitive resistor biasing caused data to be mangled at speeds higher than 350KHz. I^2C, however, has one bit of metadata for every eight bits of data sent. This Acknowledge(ACK) / No-Acknowledge (NACK) metadata bit is very useful since it preserves the parity of the data. Since even one lost bit in a 1500-byte packet will cause a checksum failure, I^2C was chosen as the more reliable solution.

The transfer process implementation involved a communication window. One LPC1768 was set as the master and the other LPC1768 as the slave. Each device would listen for Ethernet packets on a thread while the I2C communication was handled on a different thread. As soon as a packet was received it was added to an incoming message queue. The master sends a write write-read request to the slave, and the message in the front of the message queue is sent as a 600-byte fixed-size array to the master. The master then adds the message to the outgoing priority queue with an interrupt timer. When the timer expires, the message is sent out over the ethernet. The master to slave transfer is almost identical with the exception that the master sends a write request to the slave.

4 Expected Outcomes

The proposed solution should be able to create delays between two networked devices and simulate network round

trip time delays on the hardware level. Figure 5 and Figure 6 show examples of artificial delays created using (1) and (2). Along with creating such delays, it should be able to systematically drop a certain number of packets to simulate packet loss. For the purpose of this experiment, the device should be able to recreate the delays inherent to two different loss classification models: a lossy connection, and a large, overburdened network. If the device functions as expected, the interpolated data should be viable for neural network training.

An ideal output scenario for this experiment will be a device that can create network delay patterns that are so similar to real delays upon which an NN is trained, that the NN should recognize the patterns and behave accordingly. This outcome would indicate that a low-cost device can indeed be used to create real-time delays that can simulate and thus replace expensive networks for at least some phases of experimentation involving network protocol modification.

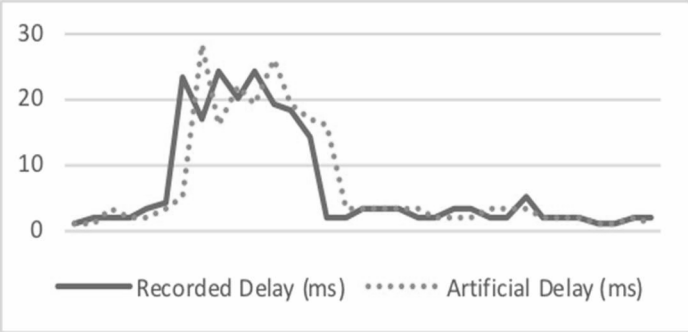

Fig. 5. Simulated congestion round trip times created using data interpolation.

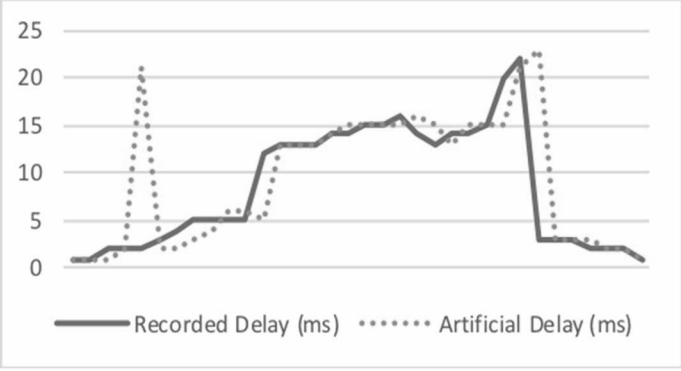

Fig. 6. Simulated weak wireless signal round trip time created using data interpolation.

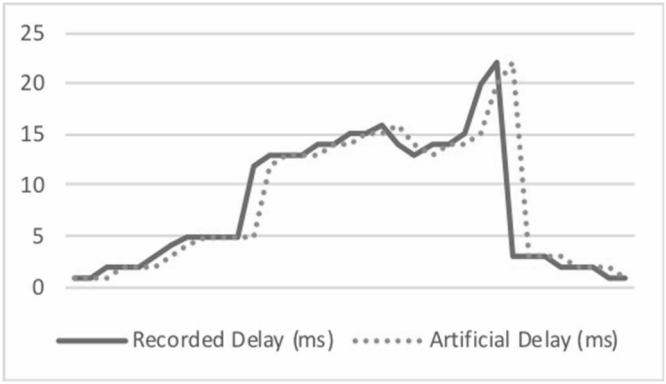

Fig. 7. Comparison of two artificially created delays using same original delay data.

5 Initial Results

An experiment was conducted to create some baseline results. First, ping tests were conducted between a server and a workstation on two sides of campus. A network test revealed that they were separated by four routers, a different domain controller, and an SSH tunnel through a bastion host. A long series of pings were sent between the computers and the results were recorded. Next, two computers were connected on both sides of the device. The average hardware delay of the device and equations (1) and (2) were used to create artificial delays to make it as similar to the recorded values as possible. Each computer sent a long sequence of ICMP pings with 56-bytes of payload data and 28-bytes of header, tail, and encapsulation data. Figure 8 shows the recorded RTT delays of the computers separated by a large network compared to the delays of the computers separated by the device.

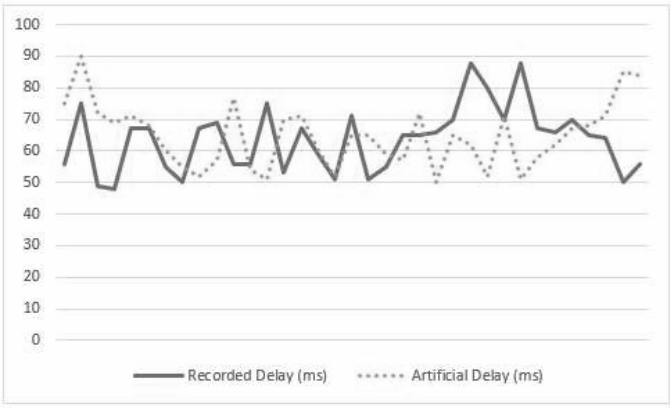

Fig. 8. Recoded round trip times compared to the recorded hardware created artificial delays.

To test the device's ability to simulate congestion delays, two workstations that were normally heavily interconnected in a domain-based large-scale network were disconnected from the network and attached to each other with the device. A packet analysis revealed that many packets were being dropped

because of the high intensity of the traffic. Many Domain Name Service (DNS) requests were being sent along with a large number of ports continuously sending out retransmissions because of the lack of acknowledgments from network resources that timed out when the computers were disconnected. Figure 9 shows the RTT delays of this experiment.

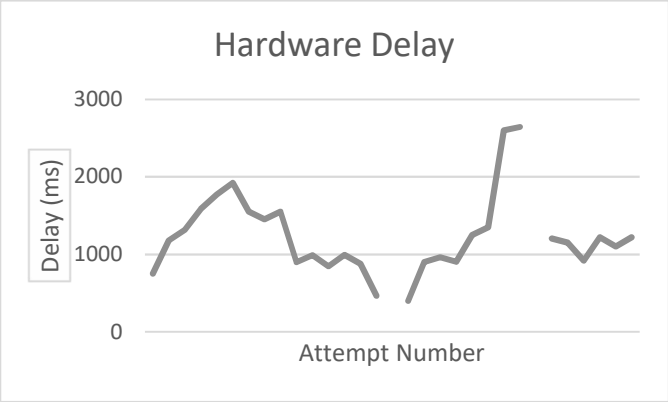

Fig. 9. Recorded round-trip times using hardware delays. The gaps in the graph represent the dropped packets.

6 Conclusion

The results obtained with these initial tests appear promising enough to require further research. The interpolated delays applied to real-time traffic creates a level of randomness caused by factors that simply cannot be accounted for or introduced in software simulations. For example, it can be very difficult to use software simulations to simulate large-scale networks with application-driven traffic. Since application data is as unpredictable as the needs of the people using them, creating a simulation complex enough to accurately model such traffic is impossible today. However, simply recording delay information during key times and applying interpolated delays during experimentation may prove to be an invaluable tool to researchers and network engineers.

By using hardware to simulate the network delays, two observations can be made: that hardware can be used in a limited fashion to replace the testing portion for NNs that work with real-time data, and that such a solution has many benefits over existing methods that involve the use of expensive production networks.

The first significant benefit of replacing complex networks with microcontrollers for loss simulation purposes is the cost. Production switch/servers used to purposely congest networks can be prohibitively expensive whereas a microcontroller is generally a few dollars at most. Another benefit of using a microcontroller to simulate packet loss is, in fact, its limited memory. The smaller the user addressable memory on a microcontroller, the more packets lost because of simulated overflow.

The portability that a microcontroller-based solution offers goes a long way. A microcontroller can sit between two routers to simulate a long wire with a long propagation delay, or it can sit behind two desktops and simulate the delays that would be experienced if the two computers were separated by a congestion burdened network. Two reasons that production switches reside in specialized server rooms are that they are very power hungry, and they release a lot of heat that must be vented out. This embedded solution would cut the electricity usage and would never noticeably heat up.

We hope that the proposed solution can create artificial delay patterns that are similar enough to real delays to be used as a training and validation tool for NN training.

7 References

[1] J.M. Natali and J.M. Pinto, "Piecewise polynomial interpolations and approximations of one-dimensional functions through mixed integer linear programming," in *Optimization Methods and Software,* vol. 24, no. 4-5, pp 783-803, 2009.

[2] P. Geurts, I. E. Khayat, and G. Leduc, "A machine learning approach to improve congestion control over wireless computer networks," in *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, Nov 2004, pp. 383–386.

[3] I. El Khayat, P. Geurts, and G. Leduc, "Enhancement of tcp over wired/wireless networks with packet loss classifiers inferred by supervised learning," *Wireless Networks*, vol. 16, no. 2, pp. 273–290, Feb 2010. [Online]. Available: https://doi.org/10.1007/s11276-008-0129-y

[4] M. J. Arshad and M. Saleem, "A simulation-based study of fast tcp compared to sctp: Towards multihoming implementation using fast tcp," *Journal of Communications and Networks*, vol. 12, no. 3, pp. 275–284, June 2010.

[5] R. Alvizu, S. Troia, G. Maier, and A. Pattavina, "Matheuristic with machine-learning-based prediction for software-defined mobile metro- core networks," *IEEE/OSA Journal of Optical Communications and Networking,* vol. 9, no. 9, pp. D19-D30, Sept 2017.