Dynamic NSTART variation for better channel utilization

Kunal Verma (2019UCP1358)
Pratha Jaiswal (2019UCP1360)
Sachin Raj (2019UCP1328)

Project Supervisor
Dr. Arka Prokash Mazumdar, Assistant Professor

May 11, 2023

Outline

- Introduction
- Motivation
- Objectives
- Literature review
- Proposed Methodology
- Result analysis
- Conclusion and future scope
- References

Introduction

- CoAP - Constrained Application Protocol
  1. NSTART value
  2. Congestion
  3. RTO - Retransmission Time out
  4. Exponential Backoff
- CoCoA - Congestion Control Advance
  1. Variable Backoff Factor

Motivation

- RFC-7252 states a fixed value of NSTART to avoid congestion.
- CoCoA also defines a static value for NSTART.
- Static value of NSTART
  1. Under utilization in case of high bandwidth
  2. Congestion in case of low bandwidth

Objective

- How should we vary NSTART parameter for improved throughput and less congestion in the network?

Contributions

- Kunal Verma - Algorithm Research and Implementation
- Sachin Raj - Algorithm and Testing Research
- Pratha Jaiswal - Testing and Performance Evaluation

Related Literature

- Carsten Bormann proposed CoCoA which improve RTO calculation mechanism of default Coap.
- Betzler, A did an evaluation of advance congestion control technique on unreliable CoAP communication.
- Bacco, M. studied the impact of NSTART for 7 values between 1 and 100 (1, 2, 3, 4, 5, 10, and 100) in satellite-based M2M communications. [1]
- Toward Adaptive Range for Parallel Connections in CoAP [2] suggested an algorithm to dynamically set NSTART value.It uses congestion ratios to decide NSTART value.

Approach

1. When do we increase the value of NSTART?
2. When do we decrease the value of NSTART?
3. When do we not change the NSTART?

Proposed approach

- RTO (Retransmission Timeout) plays a major role in overall throughput of the network.
- As congestion in the network increases the RTO value increases.
- Our main idea was to not let RTO value get too big.
- So we adjusted the NSTART value accordingly.

Flow chart

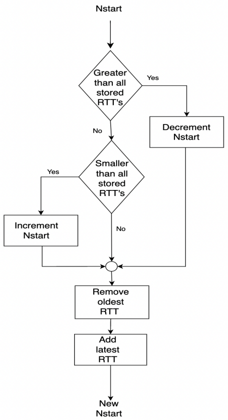


Figure: Flow chart of the proposed algorithm

Algorithm

Algorithm 1 An algorithm for dynamic NSTART value

Ensure: $NSTART = 4$ ▷ Intial NSTART value is default 4

```
if len(RTT_Q) ≠ Q_LEN then
    push RTT → RTT_Q
    return
end if
if RTT > max(RTT_Q) & NSTART > MIN_NSTART then
    NSTART ← NSTART − Df
end if
if RTT < min(RTT_Q) & NSTART < MAX_NSTART then
    NSTART ← NSTART + If
end if
pop → RTT_Q
push RTT → RTT_Q
```

Figure: Proposed algorithm

Data structures and variables

- RTT_Q : Queue to maintain last few RTT's
- Q_LEN : Number of RTT's to be maintained in the queue
- MAX_NSTART : Maximum NSTART value
- MIN_NSTART : Minimum NSTART value
- If : Increment factor
- Df : Decrement factor

Test Setup

1. Test network setup
2. Server: Raspberry Pi Model B+ V1.2. The specification of the server is 700 MHz CPU, 512 MB RAM, Ethernet Port, and 8 GB Micro SD card with Raspbian Linux OS.
3. Client: The PC has 8 GB RAM 256 GB SSD and 1TB HDD.
4. Server and Client configured using Californium
5. Evaluation parameters used: Average Throughput/Turnaround time

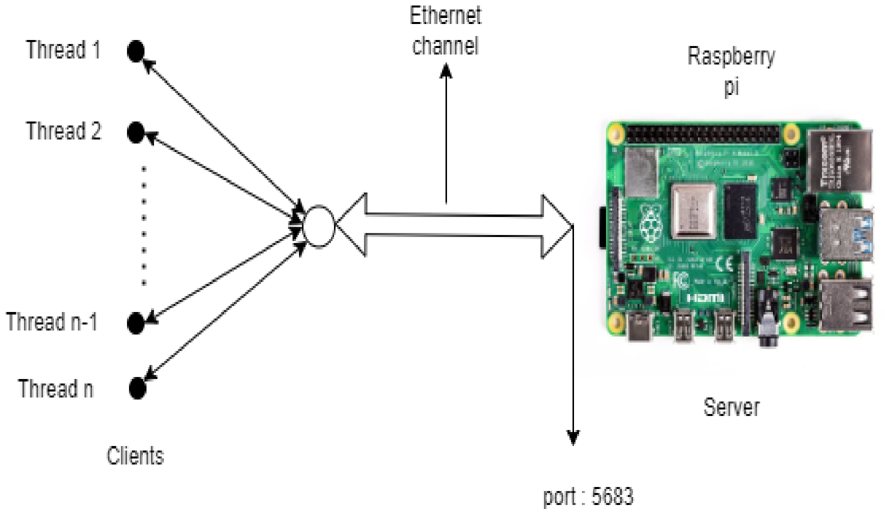


Figure: Testing Setup

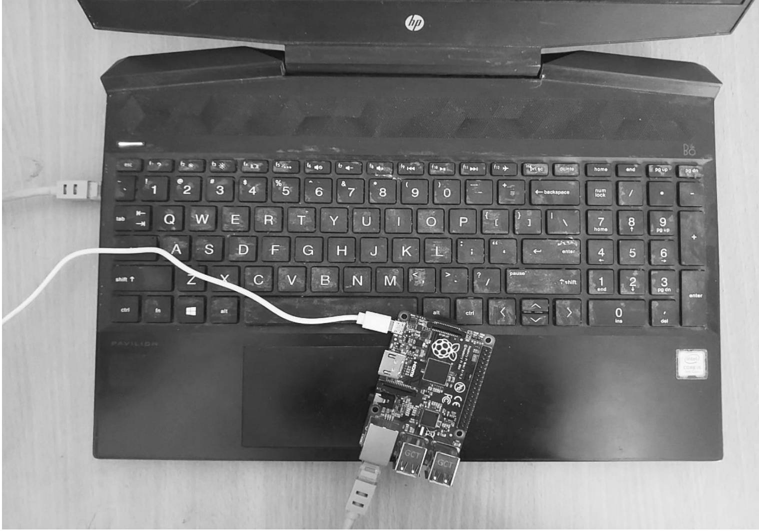


Figure: Test setup

Performance Evaluation

1. Relation between throughput and turnaround time.
2. The proposed algorithm does not offer much benefits for the earlier clients i.e. those whose packets are least affected by congestion but as far as the later clients those who are more affected by congestion are concerned these are the one who benefit from our approach.
3. The proposed algorithm performs best for later clients as it varies NSTART such as to prevent more congestion and at the same time maximizing channel utilization thereby preventing RTO values from increasing too much.

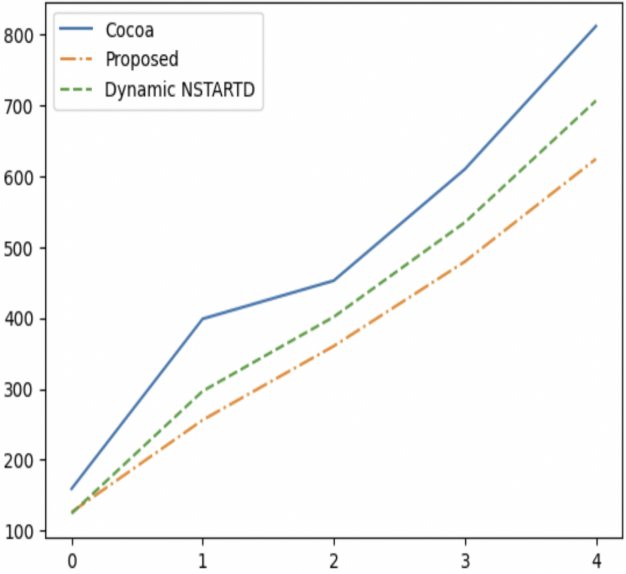


Figure: Average turnaround time vs Number of parallel clients

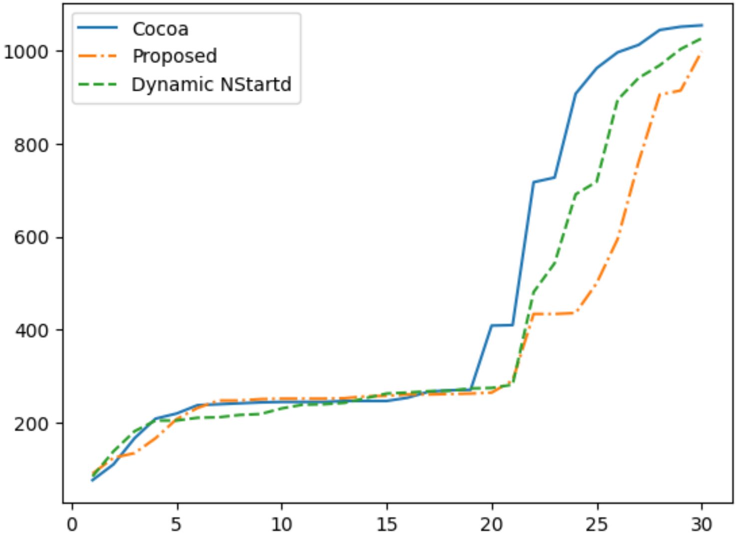
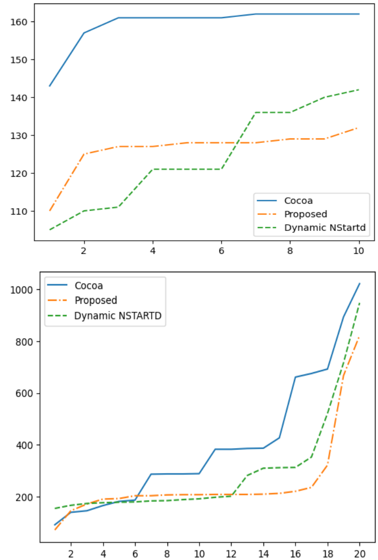
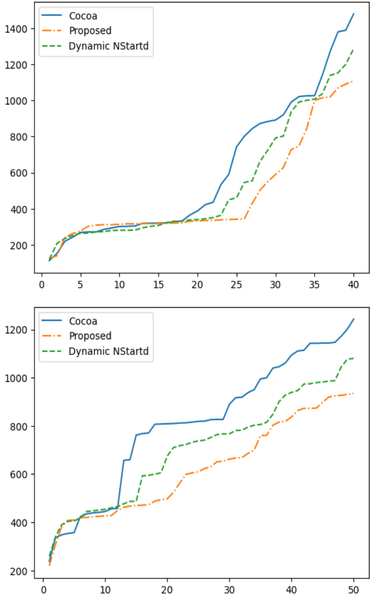


Figure: 30 clients



Turnaround time graph a. n=10 b. n=20



Turnaround time graph a. n=40 b. n=50

Conclusion and Future Scope

1. The main aim of this project was to find a way to change NSTART dynamically so as to improve the throughput of the network and our algorithm was able to do so on the network we tested.
2. There was better average throughput for the clients. Especially for the congested clients who had more turnaround time.
3. The future work of this project include further testing of the algorithm in different network scenarios.

References I

A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, “Evaluation of advanced congestion control mechanisms for unreliable coap communications,” *Ad Hoc Networks*, vol. 33, pp. 126–139, 2015.

V. K. Jain, A. P. Mazumdar, and M. C. Govil, “Toward adaptive range for parallel connections in CoAP,” *Arabian Journal for Science and Engineering*, vol. 46, pp. 3595–3611, Apr. 2021.

Thank You