CO300: Computer Networks

Course Instructors: Saumya Hegde and Mohit P. Tahiliani

Instructions to carry out the assignment work

1. Each project group must have exactly 3 students.
2. Select three projects (priority wise) from the projects listed below latest by 15.09.2018.
3. 07.09.2018 to 15.09.2018 are reserved as buffer to address project specific queries.
4. The project must be hosted on github, with all the students listed as contributors.
5. Course Instructor must be added by every group to their respective github repositories.
6. Course Instructor's github handle: mohittahiliani
7. The project duration is for 2 months (starting from 16.09.2018 to 15.11.2018).
8. Your progress will be tracked by looking at your github commits i.e., how frequently you commit, and the complexity of each commit. Your interactions with respective mentors would play a vital role in the evaluation of your assignments.

Assignment type: Direct Code Execution (DCE)

Direct Code Execution (DCE) is a framework over ns-3 that provides a feature to run simulations using real Linux network stack. Besides, it provides a feature to use real network applications (such as iperf) inside ns-3 environment without changing their code. This feature helps to leverage the functionalities of existing network tools available for Linux inside the ns-3 environment. The main goal of the assignments listed below is to use DCE and validate the implementation of different TCPs in ns-3. All the projects listed are of direct interest to the ns-3 community.

Mentors for these assignments:
Apoorva Bhargava, Shefali Gupta, Sourabh Jain and Mohit P. Tahiliani

DCE-01: Validate the ns-3 implementation of CUBIC TCP

Brief: CUBIC is the default TCP used in the Linux kernel and recently, a patch has been made available for CUBIC implementation in ns-3. CUBIC is an optimized congestion control for high bandwidth high latency networks (also known as Long Fat Networks). In this project, the aim is to validate ns-3 CUBIC implementation by comparing the results obtained from it to those obtained by simulating Linux CUBIC.
Required experience: C and C++
Bonus experience: Knowledge of CUBIC and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_cubic.c)
- CUBIC Paper (Link: https://dl.acm.org/citation.cfm?id=1400105)

- ns-3 code for CUBIC (Link: https://github.com/natale-p/ns-3-dev-git/tree/tcp-versions-updated/src/internet/model)

DCE-02: Validate the ns-3 implementation of BIC TCP

Brief: Binary Increase Congestion control (BIC) is a precursor to CUBIC and is targeted for Long Fat Networks. It uses a binary search algorithm to find the optimal value of congestion window (*cwnd*). In this project, the aim is to validate ns-3 BIC implementation by comparing the results obtained from it to those obtained by simulating Linux BIC.
Required experience: C and C++
Bonus experience: Knowledge of BIC and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:
- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_bic.c)
- BIC Paper (Link: https://ieeexplore.ieee.org/abstract/document/1354672/)
- ns-3 code for BIC (Path: ns-3.xx/src/internet/model/tcp-bic.{h, cc})

DCE-03: Validate the ns-3 implementation of YeAH TCP

Brief: Yet another High Speed (YeAH) TCP uses a mixed signal of packet loss and packet delay to calculate the appropriate value of congestion window (*cwnd*). The main design goal of this TCP variant is to be highly efficient while ensuring fairness among TCP flows. In this project, the aim is to validate ns-3 YeAH implementation by comparing the results obtained from it to those obtained by simulating Linux YeAH.
Required experience: C and C++
Bonus experience: Knowledge of YeAH and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:
- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_yeah.c)
- YeAh Paper (Link: http://infocom.uniroma1.it/~vacirca/yeah/yeah.pdf)
- ns-3 code for YeAH (Path: ns-3.xx/src/internet/model/tcp-yeah.{h, cc})

DCE-04: Validate the ns-3 implementation of TCP HighSpeed

Brief: HighSpeed TCP (HSTCP) is a congestion control algorithm proposed in RFC 3649. It aims to improve the bandwidth utilization in networks with high Bandwidth Delay Product (BDP). HSTCP achieves this by making minor modifications to the traditional TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm. In this project, the aim is to validate ns-3 HSTCP implementation by comparing the results obtained from it to those obtained by simulating Linux HSTCP.
Required experience: C and C++
Bonus experience: Knowledge of HSTCP and TCP implementation in ns-3

Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_highspeed.c)
- RFC 3649 (Link: https://www.rfc-editor.org/info/rfc3649)
- ns-3 code for HSTCP (Path: ns-3.xx/src/internet/model/tcp-highspeed.{h, cc})

DCE-05: Validate the ns-3 implementation of Scalable TCP

Brief: Scalable TCP is a congestion control algorithm which does not reduce the congestion window (cwnd) by ½ like the traditional TCP. Instead, it reduces the cwnd by ⅞ with an aim to allow the sender to reach its previous cwnd value sooner. This algorithm ensures that the bandwidth is sufficiently utilized in the presence of transient congestion. In this project, the aim is to validate ns-3 Scalable TCP implementation by comparing the results obtained from it to those obtained by simulating Linux Scalable TCP.
Required experience: C and C++
Bonus experience: Knowledge of Scalable TCP and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_scalable.c)
- Scalable Tcp Paper (Link: https://ieeexplore.ieee.org/abstract/document/832487/)
- ns-3 code for Scalable TCP (Path: ns-3.xx/src/internet/model/tcp-scalable.{h, cc})

DCE-06: Validate the ns-3 implementation of TCP Illinois

Brief: TCP Illinois uses a mixed signal of packet loss (primary) and queuing delay (secondary) to calculate the appropriate value of congestion window (cwnd). It uses Additive Increase and Multiplicative Decrease (AIMD) like traditional TCP to adjust the value of its *cwnd*, but the additive and multiplicative factors are not fixed, instead they are a function of average queuing delay. In this project, the aim is to validate ns-3 TCP Illinois implementation by comparing the results obtained from it to those obtained by simulating Linux TCP Illinois.
Required experience: C and C++
Bonus experience: Knowledge of TCP Illinois and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_illinois.c)
- TCP Illinois Paper (Link: https://www.sciencedirect.com/science/article/pii/S0166531607001307)
- ns-3 code for TCP Illinois (Path: ns-3.xx/src/internet/model/tcp-illinois.{h, cc})

DCE-07: Validate the ns-3 implementation of H-TCP

Brief: Hamilton TCP (H-TCP) is a loss based congestion algorithm which also uses AIMD to adjust its cwnd. H-TCP is also targeted to improve the performance of TCP in high BDP networks. It increases the aggressiveness based on the time elapsed after the last packet was lost. In this project, the aim is to validate ns-3 H-TCP implementation by comparing the results obtained from it to those obtained by simulating Linux H-TCP.
Required experience: C and C++
Bonus experience: Knowledge of H-TCP and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_htcp.c)
- H-TCP Paper (Link: https://www.scss.tcd.ie/Doug.Leith/pubs/htcp3.pdf)
- ns-3 code for H-TCP (Path: ns-3.xx/src/internet/model/tcp-htcp.{h, cc})

DCE-08: Validate the ns-3 implementation of TCP Vegas

Brief: TCP Vegas is a delay based congestion control algorithm which measures per packet RTT to infer the state of congestion in the network. It uses Additive Increase Additive Decrease (AIAD) algorithm to adjust its *cwnd*. In this project, the aim is to validate ns-3 TCP Vegas implementation by comparing the results obtained from it to those obtained by simulating Linux TCP Vegas.
Required experience: C and C++
Bonus experience: Knowledge of TCP Vegas and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_vegas.c)
- Tcp Vegas (Link: https://ieeexplore.ieee.org/abstract/document/464716/)
- ns-3 code for TCP Vegas (Path: ns-3.xx/src/internet/model/tcp-vegas.{h, cc})

DCE-09: Validate the ns-3 implementation of TCP Westwood

Brief: TCP Westwood is designed to improve the performance of TCP in wireless networks. It estimates the available bandwidth in the network to adjust its *cwnd*. In this project, the aim is to validate ns-3 TCP Westwood implementation by comparing the results obtained from it to those obtained by simulating Linux TCP Westwood.
Required experience: C and C++
Bonus experience: Knowledge of TCP Westwood and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)

- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_westwood.c)
- Tcp Westwood Paper (Link: https://dl.acm.org/citation.cfm?id=381704)
- ns-3 code for TCP Westwood (Path: ns-3.xx/src/internet/model/tcp-westwood.{h, cc})

DCE-10: Validate the ns-3 implementation of TCP Veno

Brief: TCP Veno is also targeted to improve the performance of TCP in wireless environments. It is a combination Vegas and Reno, and hence, named as Veno. In this project, the aim is to validate ns-3 TCP Veno implementation by comparing the results obtained from it to those obtained by simulating Linux TCP Veno.
Required experience: C and C++
Bonus experience: Knowledge of TCP Veno and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_veno.c)
- TCP Veno Paper (Link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1177186)
- ns-3 code for TCP Veno (Path: ns-3.xx/src/internet/model/tcp-veno.{h, cc})

DCE-11: Validate the ns-3 implementation of TCP Hybla

Brief: TCP Hybla is targeted to improve the performance TCP in Satellite networks where RTT is usually large. In this project, the aim is to validate ns-3 TCP Hybla implementation by comparing the results obtained from it to those obtained by simulating Linux TCP Hybla.
Required experience: C and C++
Bonus experience: Knowledge of TCP Hybla and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_hybla.c)
- ns-3 code for TCP Hybla (Path: ns-3.xx/src/internet/model/tcp-hybla.{h, cc})

DCE-12: Validate the ns-3 implementation of TCP Low Priority

Brief: TCP Low Priority (TCP-LP) is a Less than Best Effort TCP. This category of TCPs are used to ensure that bandwidth hungry large flows do not saturate the bandwidth and starve small delay sensitive flows. In this project, the aim is to validate ns-3 TCP-LP implementation by comparing the results obtained from it to those obtained by simulating Linux TCP-LP.
Required experience: C and C++
Bonus experience: Knowledge of TCP-LP and TCP implementation in ns-3
Difficulty: Moderate

Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_lp.c)
- TCP LP Paper (Link: https://ieeexplore.ieee.org/abstract/document/1209192/)
- ns-3 code for TCP-LP (Path: ns-3.xx/src/internet/model/tcp-lp.{h, cc})

DCE-13: Validate the ns-3 implementation of TCP LEDBAT

Brief: TCP Low Extra Delay Background Transport (LEDBAT) is also Less than Best Effort TCP defined in RFC 6817. It is a defacto TCP used by BitTorrent flows. Apple servers also use LEDBAT for iOS and Macintosh updates. In this project, the aim is to validate ns-3 TCP LEDBAT implementation by comparing the results obtained from it to those obtained by simulating Linux TCP LEDBAT. The challenging part of this project is that TCP LEDBAT is not merged in the mainline of Linux yet. So testing the Linux LEDBAT using DCE is a tricky thing.
Required experience: C and C++
Bonus experience: Knowledge of TCP LEDBAT and TCP implementation in ns-3
Difficulty: High
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (will be provided on request)
- RFC 6817 (Link: https://www.rfc-editor.org/info/rfc6817)
- ns-3 code for TCP LEDBAT (Path: ns-3.xx/src/internet/model/tcp-ledbat.{h, cc})

DCE-14: Validate the ns-3 implementation of Data Center TCP

Brief: Data Center TCP (DCTCP) is the most widely used TCP in Data Center Networks. It works by estimating the amount of congestion in the network and adjust its *cwnd* accordingly. In this project, the aim is to validate ns-3 DCTCP implementation by comparing the results obtained from it to those obtained by simulating Linux DCTCP.
Required experience: C and C++
Bonus experience: Knowledge of DCTCP and TCP implementation in ns-3
Difficulty: Moderate
Recommended Reading:

- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Linux kernel code (Link: https://elixir.bootlin.com/linux/v4.4/source/net/ipv4/tcp_dctcp.c)
- DCTCP Paper (Link: https://people.csail.mit.edu/alizadeh/papers/dctcp-sigcomm10.pdf)
- RFC 8257 (Link: https://tools.ietf.org/html/rfc8257)
- ns-3 code for DCTCP (Link: https://github.com/Vivek-anand-jain/ns-3-dev-git/tree/dctcp-dev)

Assignment type: ns-3 and Linux kernel

The main goal of the assignments listed below is to design, develop and test new models in ns-3 and/or Linux kernel. The tasks involve: design and development of the algorithm, testing, documenting the model and providing an example for the user to work with the algorithm. All the projects listed below are of direct interest to the Internet community.

Mentors for these assignments:
Shikha Bakshi, Jendaipou Palmei, Viyom Mittal, Vivek Jain, Sachin Patil and Mohit P. Tahiliani

SIM-01: Implementation of CAIA Delay Gradient (CDG) in ns-3

Brief: CAIA Delay-Gradient (CDG) is a TCP congestion control originating from Swinburne University's Centre for Advanced Internet Architectures (CAIA). Its development was part of CAIA's efforts to implement a new congestion control framework in FreeBSD. CDG is implemented in the mainline of Linux kernel. This project aims to implement a model of CDG in ns-3.
Required experience: C and C++
Bonus experience: Knowledge of RTT and RTO calculations in TCP
Difficulty: High
Recommended Reading:

- Implementing CAIA Delay Gradient in Linux
  (Link: http://folk.uio.no/kennetkl/jonassen_thesis.pdf)
- Revisiting TCP Congestion Control using Delay Gradients
  (Link: https://link.springer.com/content/pdf/10.1007/978-3-642-20798-3_25.pdf)

SIM-02: Implementation of TCP-NV (New Vegas) in ns-3

Brief: TCP-NV is a major update to TCP Vegas. TCP Vegas is implemented in ns-3. This project aims to implement TCP-NV in ns-3. Linux implementation of TCP-NV is available for reference.
Required experience: C and C++
Bonus experience: Knowledge of TCP Vegas
Difficulty: Moderate to High
Recommended Reading:

- TCP-NV (Link: http://www.brakmo.org/networking/tcp-nv/)
- TCP Vegas code in ns-3 (Path: ns-3.xx/src/internet/model/tcp-vegas{.h, .cc})

SIM-03: Implementation of Rate Control Protocol (RCP) in ns-3

Brief: RCP is a congestion control protocol designed by researchers from Stanford University for fast download times. It's main aim is to reduce the flow completion times. An out-of-tree implementation is available for RCP in ns-2. This project aims to implement RCP in ns-3.
Required experience: C and C++
Bonus experience: Knowledge of RCP
Difficulty: High

Recommended Reading:

- RCP Webpage (Link: http://yuba.stanford.edu/rcp/)
- RCP patch for ns-2.35 (Link: http://mohittahiliani.blogspot.in/2012/08/rate-control-protocol-rcp-patch-for-ns.html)

SIM-04: Implementation of HULL algorithm in ns-3

Brief: High bandwidth Ultra Low Latency (HULL) extends the advantages of Data Center TCP (DCTCP) and aims to provide lower latencies than DCTCP. DCTCP has been implemented in ns-3 recently as a part of GSoC 2017. In this project, it is expected to extend the DCTCP model in ns-3 to support HULL.
Required experience: C and C++
Bonus experience: Knowledge of ns-3 is a plus.
Difficulty: High
Recommended Reading:

- Less Is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center (https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/alizadeh)
- Reproducing Network Research Wordpress Blog (Link: https://reproducingnetworkresearch.wordpress.com/2012/06/08/hull-high-bandwidth-ultra-low-latency/)

SIM-05: Implementation of ECN+ algorithm in ns-3

Brief: Explicit Congestion Notification (ECN) is a signaling mechanism used to inform the TCP sender about congestion. It was proposed in RFC 3168 and has been implemented in the mainline of ns-3. ECN+ is a minor extension of ECN wherein SYN/ACK packets also carry the ECN marks. This project intends to implement ECN+ in ns-3.
Required experience: C++
Bonus experience: Knowledge of ECN.
Difficulty: Low to Moderate
Recommended Reading:

- The Power of Explicit Congestion Notification (Link: http://www.cs.northwestern.edu/~akuzma/doc/ecn.pdf)
- ECN support for TCP in ns-3 (Link: https://www.nsnam.org/docs/models/html/tcp.html#support-for-explicit-congestion-notification-ecn)

SIM-06: Implementation of ECN+/Wait algorithm in ns-3

Brief: ECN+/Wait is a further extension of ECN+ (See SIM-05). This algorithm is described in RFC 5562. This project intends to implement ECN+/Wait in ns-3.
Required experience: C and C++
Bonus experience: Knowledge of ECN
Difficulty: Low to Moderate
Recommended Reading:

- RFC 5562 (Link: https://tools.ietf.org/html/rfc5562)

- ECN support for TCP in ns-3 (Link: https://www.nsnam.org/docs/models/html/tcp.html#support-for-explicit-congestion-notification-ecn)

SIM-07: Implementation of ECN+/TryOnce algorithm in ns-3

Brief: ECN+/TryOnce is also an extension of ECN+ (See SIM-05). This algorithm is described in RFC 5562. This project intends to implement ECN+/TryOnce in ns-3.
Required experience: C and C++
Bonus experience: Knowledge of ECN
Difficulty: Low to Moderate
Recommended Reading:

- RFC 5562 (Link: https://tools.ietf.org/html/rfc5562)
- ECN support for TCP in ns-3 (Link: https://www.nsnam.org/docs/models/html/tcp.html#support-for-explicit-congestion-notification-ecn)

SIM-08: Add the feature of Alternative Backoff with ECN for TCP in ns-3

Brief: Alternative Backoff with ECN (ABE) is a newly proposed feature to enhance the performance of TCP when ECN is deployed. The main idea of ABE is to make the TCP sender respond differently to an ECN signal than it does for a packet loss. This project intends to implement this feature in ns-3.
Required experience: C++
Bonus experience: Knowledge of ECN.
Difficulty: Moderate
Recommended Reading:

- Alternative Backoff: Achieving Low Latency and High Throughput by using ECN and AQM (Link: http://heim.ifi.uio.no/~naeemk/research/ABE/Networking2017ABE.pdf)
- ECN support for TCP in ns-3 (Link: https://www.nsnam.org/docs/models/html/tcp.html#support-for-explicit-congestion-notification-ecn)

SIM-09: Implementation of Double-Threshold Data Center TCP (DT-DCTCP) in Linux kernel

Brief: Data Center TCP (DCTCP) is a very popular TCP extension tailored to perform well in Data Center Networks. It is already implemented in the Linux kernel. DT-DCTCP is an enhancement of DCTCP and aims to improve the queue stability. This project aims to extend the DCTCP implementation in Linux kernel to support DT-DCTCP.
Required experience: C
Bonus experience: Knowledge of DCTCP
Difficulty: Moderate
Recommended Reading:

- Ease the Queue Oscillation: Analysis and Enhancement of DCTCP (Link: http://ieeexplore.ieee.org/document/6681614/)
- Data Center TCP (Link: https://dl.acm.org/citation.cfm?id=1851192)
- RFC 8257 (Link: https://tools.ietf.org/html/rfc8257)

- DCTCP Code in Linux kernel (Link: http://elixir.free-electrons.com/linux/v4.15/source/net/ipv4/tcp_dctcp.c)

SIM-10: Implementation of ICTCP in ns-3

Brief: Incast Congestion control for TCP (ICTCP) is a TCP extension tailored to perform well in Data Center Networks, like DCTCP. ICTCP uses the receiver window adjustments to control congestion in Data Center Networks. This project aims to provide an implementation of ICTCP in ns-3.
Required experience: C and C++
Bonus experience: Knowledge of TCP Incast
Difficulty: High
Recommended Reading:

- ICTCP: Incast Congestion control for TCP in Data Center Networks (Link: http://ieeexplore.ieee.org/document/6203387/)
- TCP models in ns-3 (https://www.nsnam.org/docs/models/html/tcp.html#writing-a-new-congestion-control-algorithm)

SIM-11: Implementation of FQ-PIE algorithm in ns-3

Brief: FQ-PIE is FlowQueue combined with Proportional Integral controller Enhanced (PIE) algorithm to address the problem of bufferbloat in the network. In this project, the main aim is to implement FQ-PIE in ns-3. The current implementation of FQ-CoDel can be used as a reference to start the project.
Required experience: Knowledge of ns-3
Bonus experience: Knowledge of FQ and PIE
Difficulty: High
Recommended Reading:

- (RFC 8033) Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem (Link: https://tools.ietf.org/html/rfc8033)
- PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem (Link: http://ieeexplore.ieee.org/document/6602305/)
- ns-3 code for PIE (Path: ns-3.xx/src/traffic-control/model/pie-queue-disc.{h, cc})

SIM-12: Analysis of PIE with ECN by using Flent

Brief: PIE algorithm is defined in RFC 8033 where guidelines are provided to use ECN with PIE. However, a thorough analysis of PIE with ECN is not done yet and would be interesting to do. This project aims to understand the performance gain obtained by using ECN with PIE.
Required experience: Knowledge of PIE and ECN
Bonus experience: Knowledge of Flexible Network Tester (Flent) is a plus.
Difficulty: Low to Moderate
Recommended Reading:

- (RFC 8033) Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem (Link: https://tools.ietf.org/html/rfc8033)
- PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem (Link:

http://ieeexplore.ieee.org/document/6602305/)
- Flent (Link: http://flent.org/)

SIM-13: Analysis of PIE with ECN by using ns-3

Brief: PIE algorithm is defined in RFC 8033 where guidelines are provided to use ECN with PIE. However, a thorough analysis of PIE with ECN is not done yet and would be interesting to do. This project aims to understand the performance gain obtained by using ECN with PIE.
Required experience: Knowledge of PIE and ECN
Bonus experience: Knowledge of ns-3 is a plus.
Difficulty: Low to Moderate
Recommended Reading:
- (RFC 8033) Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem (Link: https://tools.ietf.org/html/rfc8033)
- PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem (Link: http://ieeexplore.ieee.org/document/6602305/)
- ns-3 code for PIE (Path: ns-3.xx/src/traffic-control/model/pie-queue-disc.{h, cc})

SIM-14: Implementation of HTTP/2 model in ns-3

Brief: HTTP/2 is a widely deployed protocol and offers significant performance gains than its previous versions. It is described in RFC 7540. In this project, the aim is to implement a model of HTTP/2 in ns-3. Unfortunately, ns-3 supports only a specific implementation of HTTP/1.1
Required experience: Fundamentals of ns-3
Bonus experience: Knowledge of HTTP/2 is a plus.
Difficulty: High
Recommended Reading:
- RFC 7540 (Link: https://tools.ietf.org/html/rfc7540)
- 3GPP HTTP model in ns-3 (Link: http://code.nsnam.org/ns-3-dev/rev/e542b127ae50)

SIM-15: Implementation of MPEG-DASH models for ns-3

Brief: Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is an adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. In this project, the aim is to implement a model of MPEG-DASH in ns-3.
Required experience: Fundamentals of ns-3
Bonus experience: Knowledge of MPEG-DASH is a plus.
Difficulty: High
Recommended Reading:
- Overview of MPEG-DASH (Link: https://bitmovin.com/mpeg-dash-vs-apple-hls-vs-microsoft-smooth-streaming-vs-adobe-hds/)

SIM-16: Implementation of QUIC model in ns-3

Brief: Quick UDP Internet Connections (QUIC) is an experimental protocol developed by

Google and currently used in Google Chrome browsers. The main idea of this protocol is to leverage the simplicity of UDP to reduce Internet transport latency. This project aims to implement QUIC protocol in ns-3. There already exists an implementation of QUIC in ns-3, but it is not yet validated and tested.
Required experience: Fundamentals of ns-3
Bonus experience: Knowledge of QUIC is a plus.
Difficulty: High
Recommended Reading:

- Overview of QUIC (Link: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit)

SIM-17: Rebase the implementation of COBALT on latest ns-3-dev

Brief: COBALT is a queue discipline which a result of combining CoDel and BLUE algorithms. It is recently published, but has been already implemented by NITK team in ns-3. This project aims to rebase the implementation of COBALT algorithm on latest ns-3-dev
Required experience: C and C++
Bonus experience: Knowledge of CoDel, BLUE and their implementations in ns-3
Difficulty: Moderate
Recommended Reading:

- Discussions on COBALT in Bufferbloat mailing list
  Link: (https://lists.bufferbloat.net/pipermail/cake/2016-May/001925.html)
- Linux implementation of COBALT
  Link: (https://github.com/dtaht/sch_cake/blob/master/cobalt.c)
- ns-3 implementation of COBALT (will be provided on request).

SIM-18: Integrate Crypto++ library in the latest ns-3 version

Brief: Crypto++ (or cryptopp) is a popular C++ library that contains the implementation of all well known cryptography algorithms. ns-3 is also developed using C++ and permits the user to link other C++ libraries to it. In this project, the main aim is to link Crypto++ library in ns-3 and showcase its use-cases by leveraging the implementation of Cryptographic algorithms in Crypto++.
Required experience: Fundamentals of ns-3
Bonus experience: Knowledge of Crypto++ is a plus.
Difficulty: Low to Moderate
Recommended Reading:

- Crypto++ library (Link: https://www.cryptopp.com/)
- Crypto++ Library wiki (Link: https://www.cryptopp.com/wiki)
- Linking Crypto++ with ns-3 (Link: http://www.mehic.info/2016/04/installing-and-crypto-libcryptopp-with-ns3/)

SIM-19: Implement Quick Start in ns-3

Brief: Quick-Start is designed to allow connections to use higher sending rates when there is significant unused bandwidth along the path, and the sender and all of the routers along the

path approve the Quick-Start Request. If a TCP sender would like to use Quick-Start, it puts the requested sending rate in the Quick-Start Option in the IP header of the TCP packet, called the Quick-Start Request packet. The TCP receiver returns the Quick-Start Response option in the TCP header in the responding SYN/ACK packet, called the Quick-Start Response packet, informing the sender of the results of its request.
Required experience: C and C++
Bonus experience: Fundamentals of TCP Slow Start
Difficulty: High
Recommended Reading:

- S. Floyd, M. Allman, A. Jain and P. Sarolahti, "Quick-Start for TCP and IP," RFC 4782, 2007.(http://www.rfc-editor.org/info/rfc4782)
- ns-2 code: (../ns-2.35/tcp/tcp.cc)

SIM-20: Re-implement TIMELY algorithm in ns-3

Brief: TIMELY is a delay-based congestion control protocol for data centers that uses hardware timestamps to make precise RTT measurements. It uses the gradient of RTT over time to adjust congestion window size before congestion happens. This algorithm has been already implemented in ns-3, however, with many corrections required. This project aims to re-implement TIMELY in ns-3 and send it to ns-3 developers for review.
Required experience: C and C++
Bonus experience: Prior experience with ns-3 and Fundamentals of Congestion Control
Difficulty: Moderate
Recommended Reading:

- TIMELY: RTT-based Congestion Control for the Datacenter
  Link: http://web.stanford.edu/class/cs244/papers/timely-sigcomm2015.pdf
- Existing ns-3 source code: https://github.com/andrewfang/244_timely

SIM-21: Implementing Hoe's Modification to Slow Start

Brief: It proposes changes to improve the startup behavior of the congestion control scheme. Changes proposed include using an estimated value instead of default value for the slow start threshold at start-up, and modifying the Fast Retransmit algorithm.
Required experience: C and C++
Bonus experience: Fundamentals of TCP Slow Start
Difficulty: Moderate
Recommended Reading:

- J. C. Hoe, "Improving the Start up Behaviour of a Congestion Control Scheme for TCP" Proceedings of ACM SIGCOMM. (http://dl.acm.org/citation.cfm?id=248180)

Assignment type: Internet of Things

The main goal of the assignments listed below is to analyze the performance of IoT protocols such as Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT) and others. These assignments do not require intensive programming. Most of the work in these assignments will be done by using COOJA simulator and C libraries such as libcoap. All the projects listed below are of direct interest to the IoT community.

Mentors for these assignments:
Vishal Rathod and Mohit P. Tahiliani

IoT-01: Implementation of RTT estimation in libcoap library

Brief: libcoap is a library that contains the implementation of Constrained Application Protocol (CoAP) in C. RTT estimation is an important feature which is missing in libcoap. This project aims to implement the RTT estimation feature in libcoap.
Required experience: C
Bonus experience: Knowledge of RTT and RTO calculations in TCP
Difficulty: Moderate
Recommended Reading:

- Libcoap implementation (Link: https://github.com/obgm/libcoap)
- A Comparison of RTT Estimation Algorithms (Link: https://users.soe.ucsc.edu/~slukin/surfit_report_10.pdf)

IoT-02: Integration and testing of libcoap library in ns-3 DCE

Brief: This project is similar to SIM-18 except that libcoap is to be integrated with ns-3 DCE. This will enable us to evaluate the performance of real time CoAP implementation in a virtual environment provided by ns-3 DCE.
Required experience: C and C++
Bonus experience: Knowledge of libcoap and DCE
Difficulty: Moderate
Recommended Reading:

- Libcoap implementation (Link: https://github.com/obgm/libcoap)
- Direct Code Execution (Link: https://www.nsnam.org/overview/projects/direct-code-execution/)
- Running libcoap in ns-3 DCE-1.9 (Link: https://groups.google.com/forum/#!topic/ns-3-users/t4dH9teYiIQ)

IoT-03: Analysis of CoAP Congestion Control with realistic IoT traffic

Brief: Constrained Application Protocol (CoAP) is an application protocol for Internet of Things (IoT) and uses UDP for communication. This project aims to evaluate the performance of CoAP in terms of number of transmission per second and RTO values with realistic IoT traffic.
Required experience: C, C++ and Fundamentals of COOJA simulator
Bonus experience: Knowledge of CoAP
Difficulty: Moderate
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)

IoT-04: Analysis of CoAP Congestion Control with different IoT MAC protocols

Brief: Constrained Application Protocol (CoAP) is an application protocol for Internet of Things (IoT) and uses UDP for communication. This project aims to evaluate the performance of CoAP in the presence of different Radio Duty Cycle (RDC) and MAC protocols.
Required experience: C, C++ and Fundamentals of COOJA simulator
Bonus experience: Knowledge of CoAP
Difficulty: Moderate
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)

IoT-05: Set up an experimental testbed for IoT

Brief: The main goal of this project is to test the performance of CoAP in a real testbed. CoAP sends confirmable messages and waits for an acknowledgement. We plan to calculate the RTT per packet. For this project, we can use Raspberry PI, Arduino Yun sensors and Sky or Zolerita mote (hardware would be provided by the Course Instructor).
Required experience: Real time testbed setup experience (Beginner Level)
Bonus experience: Knowledge of real time setup of IoT hardware
Difficulty: Moderate to High
Recommended Reading:

- Raspberry PI Board (Link: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/)
- Zolerita (Link: https://github.com/Zolertia/Resources/wiki/RE-Mote)
- Arduino (Link: https://www.arduino.cc/en/Guide/ArduinoYun)

IoT-06: Analysis of CoAP using FIT/IoT Lab

Brief: Constrained Application Protocol (CoAP) is an application protocol for Internet of Things (IoT) and uses UDP for communication. This project aims to evaluate the performance of CoAP by using FIT/IoT Lab.
Required experience: Fundamentals of SSH and CLI
Bonus experience: Knowledge of CoAP
Difficulty: Moderate
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)

IoT-07: Analysis of CoAP using FlockLab

Brief: Constrained Application Protocol (CoAP) is an application protocol for Internet of Things (IoT) and uses UDP for communication. This project aims to evaluate the performance of CoAP by using FlockLab.
Required experience: Fundamentals of SSH and CLI
Bonus experience: Knowledge of CoAP
Difficulty: Moderate
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)

IoT-08: Integration of Explicit Congestion Notification (ECN) with CoAP in COOJA

Brief: This is a core research project and the idea is to combine the advantages of ECN in CoAP. This idea has not been attempted in the past, but if done properly, would have a significant positive impact on the performance of IoT applications.
Required experience: Fundamentals of COOJA
Bonus experience: Knowledge of ECN and CoAP
Difficulty: High
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)
- RFC 3168 (ECN) (Link: https://tools.ietf.org/html/rfc3168)

IoT-09: Design and Implementation of Adaptive CoCoA++

Brief: CoCoA++ is an extension of CoAP algorithm and has been designed at NITK. However, the newly designed algorithm has some fixed knobs which should be made adaptive. This is also a research project and if done properly, would have a great impact on the performance of CoAP.
Required experience: Fundamentals of COOJA
Bonus experience: Knowledge of CoAP
Difficulty: High
Recommended Reading:

- CoAP (RFC 7252) (Link: https://tools.ietf.org/html/rfc7252)
- Comparison of CoAP and CoCoA+ Congestion Control Mechanisms for Different IoT Application Scenarios (Link: https://ieeexplore.ieee.org/abstract/document/8024686/)
- CoCoA+: An advanced congestion control mechanism for CoAP (Link: https://www.sciencedirect.com/science/article/pii/S1570870515000888)
- CoCoA++ details (will be provided on request)

IoT-10: Implementation of CDG in libcoap

CAIA Delay-Gradient (CDG) is a TCP congestion control originating from Swinburne University's Centre for Advanced Internet Architectures (CAIA). Its development was part of CAIA's efforts to implement a new congestion control framework in FreeBSD. CDG is implemented in the mainline of Linux kernel. This project aims to implement a model of CDG in libcoap.
Required experience: C and C++
Bonus experience: Knowledge of RTT and RTO calculations in TCP and libcoap
Difficulty: High
Recommended Reading:

- Implementing CAIA Delay Gradient in Linux
  (Link: http://folk.uio.no/kennetkl/jonassen_thesis.pdf)
- Revisiting TCP Congestion Control using Delay Gradients
  (Link: https://link.springer.com/content/pdf/10.1007/978-3-642-20798-3_25.pdf)
- Libcoap implementation (Link: https://github.com/obgm/libcoap)

Assignment type: FreeBSD

The main goal of the assignments listed below is to implement small functionalities in FreeBSD operating system and provide an understanding of its codebase to the students. The assignments listed below are related to the well known problem of Bufferbloat in the Internet.

Mentors for these assignments:
Sachin Patil, Leslie Monis and Mohit P. Tahiliani

BSD-01: Implementation of Nonlinear RED in FreeBSD

Brief: Nonlinear RED NLRED) is an AQM algorithm. It is a minor extension of RED algorithm.
Required experience: C and C++
Bonus experience: Knowledge of RED.
Difficulty: Low
Recommended Reading:

- Nonlinear RED: A simple yet efficient active queue management scheme (Link: http://www.sciencedirect.com/science/article/pii/S1389128606000879)
- ns-3 code for NLRED (Path: ns-3.xx/src/traffic-control/model/red-queue-disc{.h, .cc})

BSD-02: Implementation of Modified CoDel in FreeBSD

Brief: Controlled Delay (CoDel) is a well studied Active Queue Management (AQM) algorithm and deployed in the Linux kernel. However, it fails to control the queue delay when large number of UDP flows coexist with TCP flows. To overcome this problem, one of our Ph.D students has developed a modified CoDel algorithm and tested its performance by using ns-2. In this project, the main goal is to implement the modified CoDel algorithm in FreeBSD.

Required experience: C, C++ and CoDel
Bonus experience: Knowledge of FreeBSD
Difficulty: Moderate
Recommended Reading:

- Sachin D. Patil, Mohit P. Tahiliani (2016). *On the robustness of AQM mechanisms against non-responsive traffic* (https://ieeexplore.ieee.org/abstract/document/7947857/)
- Controlled Delay AQM (RFC 8289: https://tools.ietf.org/html/rfc8289).
- CoDel code in Linux kernel (Link: https://elixir.bootlin.com/linux/latest/source/net/sched/sch_codel.c)