REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 02/28/2020	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* 12/01/2016 - 11/30/2019

4. TITLE AND SUBTITLE Interdisciplinary Expeditionary Cyber Research	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER N000014-17-1-2046
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Noubir, Guevara; Melodia, Tommaso; Schirner, Gunar; Chowdhury, Kaushik; Barabasi, Albert-Laszlo	**5d. PROJECT NUMBER** 100006059
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Northeastern University 360 Huntington Ave. Boston, MA 02115	**8. PERFORMING ORGANIZATION REPORT NUMBER** Final
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 875 N. Randolph Street Suite 1425 Arlington, VA 22203-1995	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; distribution is Unlimited".

13. SUPPLEMENTARY NOTES
None

14. ABSTRACT
These projects bridge cyber space and physical space. Indeed, this is a hallmark of Expeditionary Cyber, the interlinking of these two spaces, and is juxtaposed to the "Cloud", in which physical location is largely irrelevant by design. Projects address research into the physical ocalization of cyber assets, the use of visible light networks, sensing, computation and sensing, computation and communication within networks of unmanned aerial system, elastic and software-defined networks and power efficient hardware, a key challenge for success in the Expeditionary Cyber realm.

15. SUBJECT TERMS
Expeditionary Cyber (EC) Research that covers topics that are core enabling issues areas for EC in networks, sensing and detecting, computation, communications, and improving hardware efficiency.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			David Luzzi
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	51	617-373-5600

NORTHEASTERN UNIVERSITY

AWARD NO. N00014-17-1-2046

Interdisciplinary Expeditionary Cyber Research

ISSUED BY:
OFFICE OF NAVAL RESEARCH
875 N. RANDOLPH STREET SUITE 1425
ARLINGTON, VA 22203-1995

INSTRUMENT TYPE: GRANT AWARD
AUTHORITY: 10 USC 2358 AND 31 USC 6304, AS AMENDED
CFDA: 12.300
DUNS NUMBER: 001423631
AWARD NUMBER: N00014-17-1-2046
MODIFICATION NO: A00002
MODIFICATION TYPE: ADM
PR NO: 4720002829
ACTIVITY/AGENCY PROPOSAL NO: GRANT12237019

PROPOSAL DATE: 08/26/2016

ISSUED TO:
NORTHEASTERN UNIVERSITY
360 HUNTINGTON AVE
BOSTON MA 02115-5005
UNITED STATES OF AMERICA

NAME OF RESPONSIBLE PERSON: DAVID LUZZI

PERIOD OF PERFORMANCE: 12/01/2016 THROUGH 11/30/2019

TABLE OF CONTENTS

All Technical Leads reported the results of their research to the **Principal Investigator, David Luzzi**, who possessed the overall responsibility for the management of this program, coordination of reporting, and submission of final reports to the Office of Naval Research.

Program Overview

The challenge of cybersecurity where the nation's Marines and Army Soldiers are in close proximity and contact with adversaries, is a growing and important threat and opportunity. Research and development in cybersecurity against these threats, and in systems and software with which to exploit and disrupt adversaries' command, control, communications and computer (C4) systems, is necessary if the DoD is to address the growing cyber challenge to our expeditionary forces. This cyber domain at the tactical edge is termed Expeditionary Cyber; as a newly considered domain, the scope and definition of Expeditionary Cyber will evolve, yet it will remain the cyber domain with the closest direct connection to saving American lives, improving mission outcomes, and minimizing collateral damage including innocent civilian casualties.

This three-year research program is the start to address the Expeditionary Cyber challenge. Seven projects will be executed on research into important enabling science and engineering that could yield valuable technological capabilities for Expeditionary Cyber. These projects bridge cyber space and physical space. Indeed, this is a hallmark of Expeditionary Cyber, the interlinking of these two spaces, and is juxtaposed to the "Cloud", in which physical location is largely irrelevant by design. Projects address research into the physical localization of cyber assets, the use of visible light networks, sensing, computation and communication within networks of unmanned aerial system, elastic and software-defined networks and power efficient hardware, a key challenge for success in the Expeditionary Cyber realm.

Research Project Titles and Technical Leads

1. **Elastic Networks for Resilient and Secure Multi-Radio Mobile Systems**
Technical Lead - Guevara Noubir
2. **Software-Defined Infrastructure-less Wireless Networking with Distributed Control**
Technical Lead - Tommaso Melodia
3. **Domain-specific Power-efficient Processing for Expeditionary Cyber Missions**
Technical Lead - Gunar Schirner
4. **Robust Localization and Time Synchronization**
Technical Lead - Guevara Noubir
5. **LANET: Visible-Light Infrastructure-less Wireless Networks for Expeditionary Cyber**
Missions Technical Lead - Tommaso Melodia
6. **Sensing, Computation and Communication on the Fly: Connected UAV Mesh Networks**
Technical Lead - Kaushik Chowdhury
7. **Understanding the Representation Power of Graph Neural Networks in Learning Graph**
Topology Technical Lead – Albert-Lazlo Barabasi

ONR N00014-17-1-2046 - PROJECT 1 - Elastic Networks for Resilient and Secure Multi-Radio Mobile Systems
Technical Lead: Guevara Noubir

Abstract. A key requirement of expeditionary networks is uninterrupted communication capability over intermittent links and highly dynamic infrastructure, even in the presence of intentional interference. However, wireless networks are notorious for not degrading gracefully and are highly sensitive to cross-layer attack. This is significantly due to a *heavy control plane* that consumes disproportionate spectrum resources in harsh conditions, such as recovery from natural disasters where a significant part of the infrastructure is damaged. The effects of current networks' poor elasticity and slow control-loop, amplify at higher layers, and results in brittle applications performance.

Elastic wireless networks address these challenges using communication techniques that have *ultra-thin control traffic* overhead and are *opportunistic* in exploiting the channel and network topology dynamics at the edge. The team made progress on several fronts towards an agile and elastic physical/MAC layer that can withstand attacks and degrades gracefully in the presence of adversaries. The research activities cover both theoretical and systems aspects. (1) agile multi-carrier physical/link layer, (2) analyzing super-position coded communication under jamming, (3) a game-theoretic framework for reasoning about cross-layer attacks, the existence of Nash-Equilibrium (NE) for communications in adversarial settings, as well as explicit analytical mixed-strategies for multi-carrier rate adaptation (modulation/coding) against a power adaptive smart-jammer.

Agile Multi-Carrier Physical/MAC Layer

We investigated the agility, potential, and limitations of three types of physical layers that can provide agility and elasticity, yet provide high spectrum efficiency. The first physical layer we considered is an Orthogonal Frequency Domain Multiplexing (OFDM) physical layer, based on our SDR implementation of Wi-Fi. While this physical layer can provide compatibility with commercial systems (both Wi-Fi & LTE), we demonstrated that OFDM does not degrade gracefully when some sub-carriers are interfered with even if such sub-carriers are not currently used by the transmitter. This is a fundamental limitation of OFDM (as it requires orthogonality from emissions across the whole band. It makes it highly sensitive to smart and power-efficient smart-jamming.

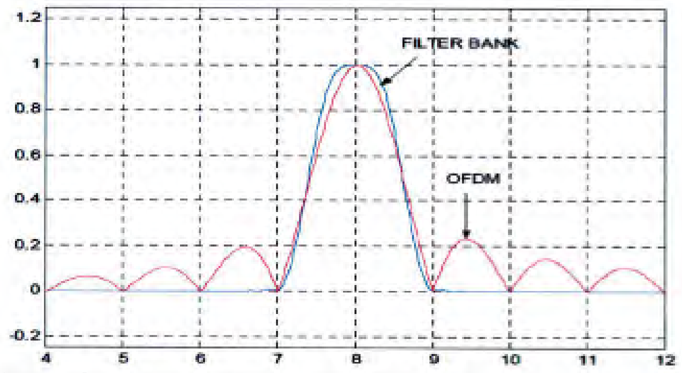


Figure 1. OFDM sub-carriers can be jammed beyond their effective spectrum usage.

Due to OFDM limitations, we investigated alternative approaches for agile multi-carrier systems, first a basic polyphase filters technique, then a Filter Bank Multi Carrier (FBMC). These physical layers provide good elasticity with graceful degradation. The transmitter uses sub-carriers independently, therefore mitigating the impact of narrowband jamming with zero-control traffic. The zero-control traffic is a key advantage as most cross-layer attacks target such channels (the Achilles' heel of wireless systems). The transmitter also senses the spectrum and re-allocates power and traffic to sub-carriers.

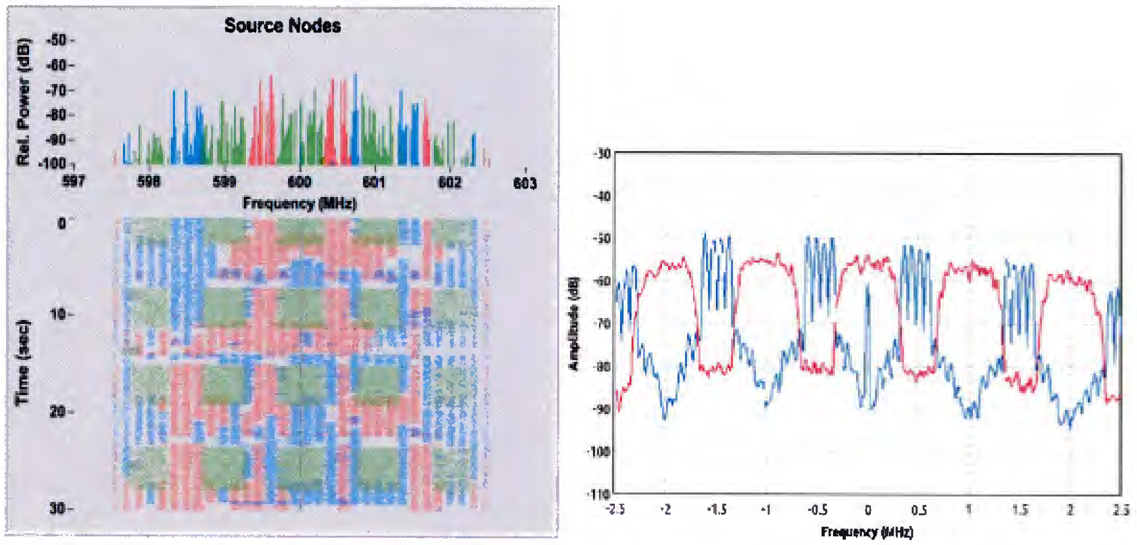


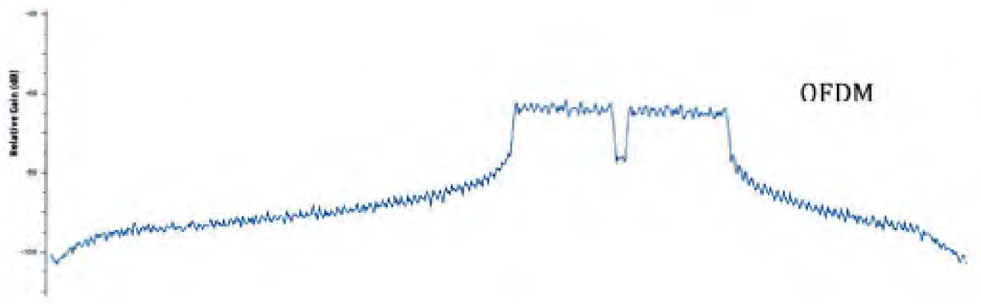
Figure 2. (Left) Illustration of three interfering polyphase filter-based multi-carrier systems.(Right) two transmitter with different numbers of sub-carriers sharing a 5MHz spectrum.

Despite its anti-jamming and elasticity potential, the basic polyphase-filters multi-carrier approach however comes at the expense of losing spectral efficiency due to the non-overlapping sub-carriers. This led us to investigate the design of a Filter Bank Multi-Carrier system that provides agility, elasticity and retains the spectrum efficiency of OFDM systems. The goal is to provide agility through the use a variable number of ultra-narrow sub-carriers, elasticity through the scheduling of traffic independently over sub-carriers and use of error correction codes with cryptographic-interleavers across sub-carriers, and finally high spectrum efficiency through the intrinsic characteristics of FBMC.

	OFDM	FBMC
Spectrum efficiency	Cyclic prefix needed	Cyclic prefix not needed ✔
1-to-N (downlink) sync	Time domain ✔	Frequency domain
N-to-1 (uplink) sync	Complex (e.g., coordination required)	Simple ✔
Inter-Channel Interference/Jamming	Large	Negligible ✔
Dynamic access	Complex	Easy ✔
Estimation and Equalization	Full spectrum: easy Shared spectrum: impractical	Can be robust, practical, but computation intensive ✔
Implementation complexity	Low ✔	High
Computation complexity	Low ✔	High

Table 1. OFDM vs. FBMC.

We prototyped the FBMC system demonstrating high rejection of interference from out-of-band emissions (See Figure 3 for comparison with OFDM), as well as agility capability to operate over sub-carriers without requiring an explicitly control channel (See Figure 4). The prototype system supports Multicarrier transmission techniques (FBMC), modulations: BPSK, QPSK, 16-QAM, 64-QAM, error correction codes: convolutional codes with coding rate 1/2, 2/3, 3/4, 5/6, simultaneous transmission and reception on adjacent bands, and multicarrier sensing.



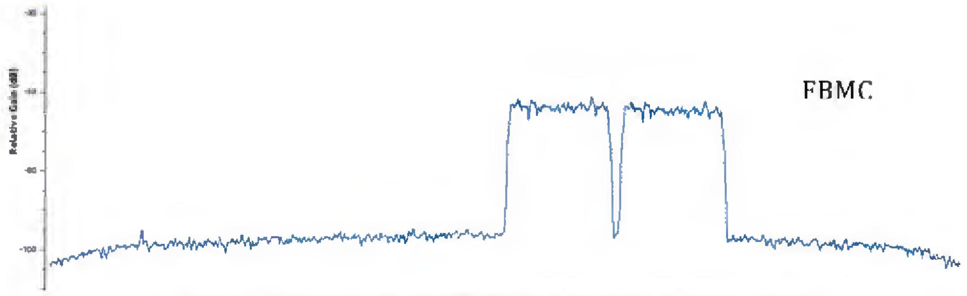


Figure 3. FBMC provides over 20dB out-of band emissions rejection.

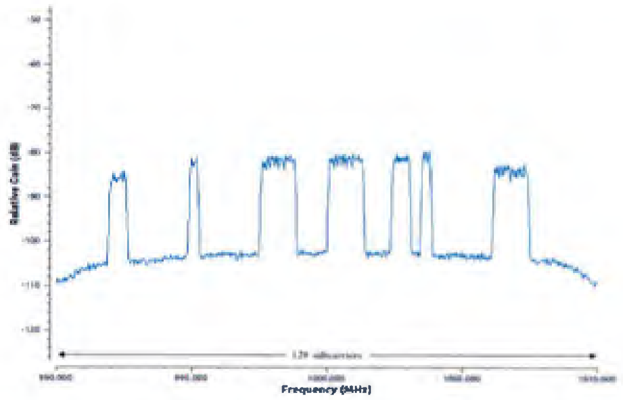


Figure 4. Agility in operating over a subset of sub-carriers with out-of-band emissions rejection and without the explicit need for negotiation over a control channel.

A theoretical framework for reasoning about randomization and game-theoretic interaction between communicating nodes and adversary

We developed a theoretical framework and techniques for elasticity to mitigate cross-layer attacks. The framework provides a systematic way for generalizing and reasoning about randomization (mechanisms hopping) as a defense against smart adversaries. The framework is rooted in game theory and aims at deriving strategies for a variety of utility functions including deception (both as a defense and attack approach).

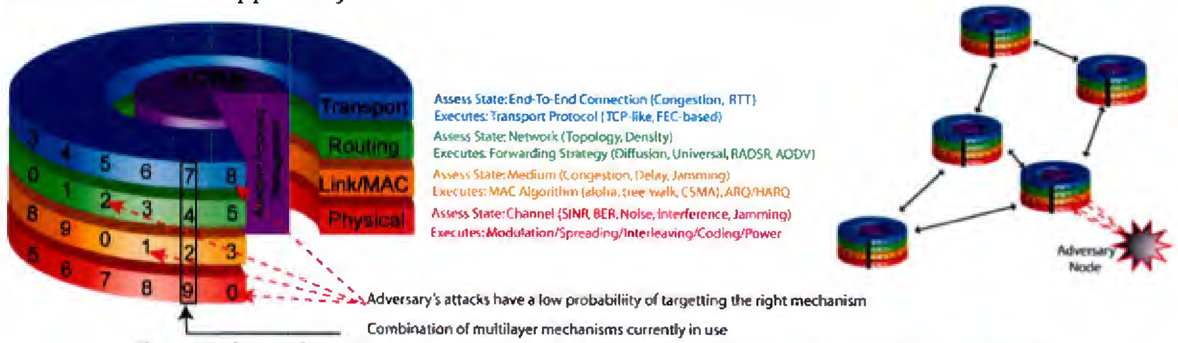

CORE
Transport
Routing
Link/MAC
Physical
Assess State: End-To-End Connection (Congestion, RTT)
Executes: Transport Protocol (TCP-like, FEC-based)
Assess State: Network (Topology, Density)
Executes: Forwarding Strategy (Diffusion, Universal, RADSR, AODV)
Assess State: Medium (Congestion, Delay, Jamming)
Executes: MAC Algorithm (aloha, tree walk, CSMA), ARQ/HARQ
Assess State: Channel (SINR, BER, Noise, Interference, Jamming)
Executes: Modulation/Spreading/Interleaving/Coding/Power
Adversary's attacks have a low probability of targetting the right mechanism
Combination of multilayer mechanisms currently in use
Adversary Node

Figure 5. Game Theory for strategic randomization to defend against smart cross-layer attacks.

Within this framework, we investigated how to defend against a power-adaptive adversary. While we can vary our rates to be more (lower rates) or less (higher rates) resilient to a given jamming power, lower rates reduce throughput. To prevent an adversary from forcing the communicating nodes to operate at lower rates, the communicating nodes can randomize their rates making it difficult for the adversary to select the jamming power that can successfully interfere with a packet.

In this work, we analytically derived Nash-Equilibrium strategies for randomizing the rates to mitigate jamming. We consider both randomization of power as well as randomization of power across multiple sub-carriers/bands. Some of our findings include, the analytical derivation of optimal mixed strategies for the fixed power rate adaptive transmitter, bounded average/peak power jammer, these strategies can be approximated by semi-uniform distribution and integrated within a multi-carrier communications system. We are currently investigating how to design practical and optimal RAA in a networked setup.

Analysis of Super-Position Coded Communications

We developed a theoretical framework for reasoning about jamming super-position coded communications. In this scheme, the transmitter can superpose multiple streams of data. The lowest layers are more robust to interference while the highest layers are more fragile. This scheme also enables elasticity and graceful degradation. In the absence of interference all layers are decodable, while in the presence of a jammer only the lower layers are recoverable. This elasticity does not require any feedback loop or control traffic. Within this context, we derived a lower and upper-bound on the achievable rates for various allocations of power to the coded layers. We are working on the integration within the game-theoretic framework.

Testbed

We built a small testbed of 4 x USRP X310 (that can operate up to 6GHz Rx/Tx on a 160 MHz bandwidth with a 2x2 MIMO capability). This testbed is used for some of the evaluation of our system performance and is shared with the secure localization and synchronization project.

Future Activities

We plan to continue our research activities towards extending the theoretical framework to analyze the connections between Elasticity & Game Theory, analyzing combinations of cross-layer techniques for elasticity, in particular the integration with higher layers such as back-pressure, MultiPath TCP. Another promising avenue of research that we already started focusses on the benefits of RF-centric machine learning models towards developing RF situational awareness (classifying emissions, their nature, and patterns).

ONR N00014-17-1-2046 - PROJECT 2 - Software-Defined Infrastructure-less Wireless Networking With Distributed Control Technical Lead: Tommaso Melodia

Institute for the Wireless Internet of Things
Northeastern University, Boston, MA 02115
Email: melodia@ece.neu.edu

1 Research Goal

The goal of this project is to study new techniques for software-defined, cross-layer controlled, infrastructure-less wireless networks (SoDiNet). SoDiNet will provide the network designer with abstractions hiding the low-level details of the network operations through a network virtualization plane. SoDiNet will virtualize the details of the distributed implementation of the network control operations, and provide the network designer with a centralized view abstracting the network functionalities at a high level. The SoDiNet control plane will take network control programs written on a centralized, high-level view of the network and auto-matically generate distributed cross-layer control programs based on distributed optimization theory that are executed at the network edge by each individual network element on an abstract, common representation of the radio hardware.

2 Accomplishments

The following outlines the main accomplishments towards this research goal.

SoDiNet Architecture and Virtualization Principle Design. At a high level, SoDiNet comprises three key components: Network Virtualization, Automated Problem Decomposition, and Programmable Protocol Stack (PPS). Network virtualization is the interface through which the network designers define their network control problems and hence to control the networks to achieve certain application-specific objectives. The user-defined centralized network control problems are decomposed into a set of distributed sub-problems, each of which characterizes the local behavior of a single session or single node. The resulting distributed algorithms are then used to define control actions online based on an abstract representation of the radio hardware and of the protocol stack. We have studied the design principle of SoDiNet virtualization and then proposed a new abstraction approach based on the design principle.

Testbed Development. A newly-designed general purpose testbed has been developed based on software-defined radio devices (USRP N210) to verify effectiveness of the SoDiNet design p rinciple. The testbed supports multi-hop end-to-end application data streaming and features a brand new programmable protocol stack that covers physical layer, link layer, network layer, transport layer and application layer. The testbed features a scalable out-of-band (OOB) control channel (UDP socket based), which supports multi-layer ac-knowledgments, signaling, and control information among others. This infrastructure will be instrumental in testing the effectiveness of the designed architecture by demonstrating joint distributed control of transport-layer transmission rate and power adaptation at physical layer in multi-hop multi-session ad hoc networks. SoDiNet has been implemented following a hierarchical architecture with three tiers, i.e., SoDiNet control host, SDR control host and SDR front-end.

At the top tier of the hierarchical architecture is the SoDiNet control host, based on which one can specify the network control objective using the provided network abstract framework WiNAR. The output of this tier is a set of automatically generated distributed solution algorithms, which will be sent to each of the SDR control hosts. At the second tier, the programmable protocol stack (PPS) is installed on each of the SDR control hosts. The distributed optimization algorithms received from the SoDiNet control host are stored at the decision plane of the PPS. At run time, the PPS will be compiled to generate operational code to control the SDR front-ends of the third tier. Finally, each of the SDR front-ends (i.e., USRP) receives the baseband samples from its control host via Gigabit Ethernet (GigE) interface and then sends them over the air with transmission parameters dynamically specified in the control commands from the SDR control hosts.

The primary benefit of prototyping SoDiNet based on an hierarchical architecture is to enable scalable network deployment. Specifically, the tier-1 SoDiNet control host is connected to all tier-2 SDR control hosts via wireless interfaces (which is Wi-Fi in current prototype), through which the generated distributed algorithms can be automatically *pushed* to and installed at each of the SDR control hosts. Hence, one needs to create a single piece of code only in order to control all the 21 USRPs.

On the SoDiNet control host, which is a Dell OPTIPLEX 9020 desktop running Ubuntu 16.04, four key SoDiNet functions have been implemented using a combination of Python 3.0 and CogApp 2.5.1, including the wireless network abstraction framework WiNAR, disciplined instantiation, automated decomposition as well as automated numerical solution algorithm generation. We base our development on Python to take advantage of its high programming efficiency and high-level expressiveness and the flexible, open-source programming interfaces to GNU Radio for controlling USRPs. CogApp is an open-source software written in Python for template programming, a programming technique based on which the automated numerical solution algorithm generation has been implemented in the current prototype.

SoDiNet PPS Design. The SoDiNet PPS has been developed in Python on top of GNU Radio to provide seamless controls of USRPs. The PPS covers all the protocol layers. The application layer opens end-to-end sessions for transferring custom data such as files, binary blobs, as well as random generated data, among others. A session can be established between any two network entities and multiple sessions can be established at the same time. Programmable parameters include the number of sessions and the number of hops in each session, as well as the desired behavior of each session, e.g., maximum/minimum rate, power budget of the nodes, among others.

The transport layer implements segmentation, flow control, congestion control as well as addressing. This layer supports end-to-end, connection-oriented and reliable data transfer. To accomplish this, a *Go-Back-N* sliding window protocol is implemented for flow control and congestion control, and transport layer acknowledgments are used to estimate the end-to-end Round Trip Time (RTT), which serves as an estimate of network congestion. Programmable parameters at this layer include transmission rate, sliding window size and packet size, among others.

The network layer implements host addressing and identification, as well as packet routing. The network layer is not only agnostic to data structures at the transport layer, but it also does not distinguish between operations of the various transport layer protocols. Routing strategies can be programmed at this layer.

At datalink layer the core functionalities include fragmentation/defragmentation, encapsulation, network to physical address translation, padding, reliable point-to-point frame delivery, Logical Link Control (LLC) and Medium Access Control (MAC) among others. In particular, the reliable frame delivery employs an hybrid LLC's *Stop and Wait* ARQ protocol and Forward Error Correction (FEC) mechanism (*Reed-Solomon* coding), such that frames are padded with FEC code and retransmissions are performed when the link is too noisy. The FEC is dynamic, reprogrammable, and can automatically adapt to the wireless link conditions at fine granularity, by increasing or decreasing the channel coding rate based on the observed packet error rate. Programmable parameters at this layer include channel coding rate, maximum retransmission times, and target residual link-level packet error rate, among others.

Finally, the physical layer features both CDMA and OFDM access schemes, yet with a wide set of modulation schemes supported, including Binary phase-shift keying (BPSK), Quadrature phase-shift keying (QPSK), Gaussian Minimum Shift Keying (GMSK) among others. Programmable parameters at the physical layer include modulation schemes, transmission power, and receiver gain, among others.

Experimental Evaluation. We test SoDiNet on the designed SDR testbed in five different networking scenarios. Scenarios 1-3 deploy six nodes and two traffic sessions; while Scenario 4 considers nine nodes and three traffic s essions, w ith e ach s ession s panning o ver t wo h ops. I n S cenario 5 , t hree s essions are deployed over 21 nodes, with six hops for each session. Six spectrum bands in the ISM bands are shared by the 21 USRPs, with bandwidth of 200 kHz for each spectrum band. At each USRP, the data bits are first modulated using GMSK and then sampled at sampling rate of configured 800 k Hz. Reed-solomon (RS) code is used for forward error coding (FEC) with coding rate ranging from 0.1 to 0.4 at a step of 0.1.

Through the experiments, we seek to demonstrate the following properties: i) *Effectiveness.* Through experiments in Scenarios 1-3, we show that SoDiNet-based network optimization outperforms non-optimal or purely locally optimal (greedy) network control; ii) *Flexibility.* Through experiments in Scenarios 4 and 5, we showcase the flexibility of S oDiNet in modifying the global network behavior by changing control objectives and constraints. iii) *Scalability.* In Scenario 5 we show the scalability of SoDiNet by deploying code over a large-scale network.

Cooperative Anti-jamming for Massive MIMO Expeditionary Cyber Networks. We have developed techniques for software-defined cooperative anti-jamming in expeditionary cyber networks (EC) with massive MIMO-enabled hotspots. Our objective is to develop countermeasures to jamming based on cooperation among hotspots to defend legitimate users against jamming attacks. The proposed techniques jointly determine the pilot sequence allocation and power control to maximize the throughput of the legitimate users. Two scenarios are considered: i) infrastructure-less EC networks with distributed hotspots, and ii) infrastructure-based EC networks with hotspots connected using high-speed links. Distributed anti-jamming strategies have been designed for both cases, and a centralized but globally optimal solution algorithm was also designed to provide a performance benchmark for the distributed algorithms.

Cooperative-beamforming-based Coexistence. In topic, we focused on EC networks where heterogeneous wireless technologies coexist on the same spectrum bands. We considered coexistence of cellular (e.g., LTE/LTE-A) and non-cellular (e.g., Wi-Fi, Bluetooth) technologies and designed coexistence schemes based on cooperative beamforming. With the new coexistence scheme, cellular networks cooperate to exploit spatial multiplexing and finally a chieve s uccessful d ownlink LTE t ransmissions w hile guaranteeing no interference towards other coexisting non-cellular networks. Three contributions have been made, i.e., CoBeam framework design, prototype development, and experimental performance evaluation.

CoBeam Framework Design. We propose for the first t ime C oBeam, a n ew, cognitive-beamforming-based spectrum sharing approach for 5G-and-beyond wireless networks. We discuss the design of the main components of the CoBeam framework, including programmable physical layer driver, cognitive sensing engine, beamforming engine, and scheduling engine.

Prototype Development. To demonstrate the effectiveness of the proposed framework, we present a prototype of CoBeam by considering a specific problem in 5G wireless networks, i.e., spectrum sharing between coexisting Wi-Fi and LTE in the same unlicensed spectrum bands.

Experimental Performance Evaluation. We extensively evaluate the performance of CoBeam on a large-scale office-space i ndoor t estbed b ased o n s oftware-defined ra dios. Th rough ex tensive ex periments, we show that an average of 169% throughput gain can be achieved for the resulting coexisting Wi-Fi/U-LTE networks with guaranteed cross-technology fairness.

Distributed Wireless Network Slicing. We considered the problem of network slicing in wireless scenarios, where the wireless infrastructure is composed of multiple Remote Radio Heads (RRHs) owned by one (or possibly more) entities. Mobile Virtual Network commanders (MVNs) can create their own virtual radio access network (RAN) through network slicing. Accordingly, the network owner leases slices of the RAN to the MVNs. We showed that the network slicing problem can be modeled as an atomic weighted congestion game with splittable flows, and then proposed a distributed iterative algorithm which provably converges to the unique Nash Equilibrium and does not require disclosure of privacy sensitive parameters from the MVNOs.

3 Publications

1. Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, "WNOS: An Optimization-based Wireless Network Operating System," in *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Los Angeles, USA, June 2018.

2. Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, "Demo Abstract: WNOS: Software-defined Generation of Distributed Optimal Control Programs for Wireless Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Honolulu, HI, April 2018.

3. Z. Guan, L. Bertizzolo, E. Demirors, T. Melodia, "WNOS: Enabling Principled Software-Defined Wireless Networking," *IEEE/ACM Transactions on Networking*, under revision, 2019.

4. Z. Guan, T. Melodia, "The Value of Cooperation: Minimizing User Costs in Multi-broker Mobile Cloud Computing Networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 780-791, Oct.-Dec. 2017.

5. L. Bertizzolo, E. Demirors, Z. Guan, T. Melodia, "CoBeam: Beamforming-based Spectrum Sharing With Zero Cross-Technology Signaling for 5G Wireless Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Beijing, China, April 2020.

6. Z. Guan, Nan Cen, T. Melodia, Scott Pudlewski, "Self-Organizing Flying Drones with Massive MIMO Networking," in *Proc. of Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Capri, Italy, June 2018.

7. Z. Guan, N. Cen, T. Melodia, S. Pudlewski, "Joint Power, Association and Flight Control for Massive-MIMO Self-Organizing Flying Drones," *IEEE/ACM Transactions on Networking*, under revision, 2019.

8. S. D'Oro, F. Restuccia, A. Talamonti, T. Melodia, "The Slice Is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Paris, France, April 2019.

9. S. D'Oro, F. Restuccia, T. Melodia, and S. Palazzo, "Low-Complexity Distributed 5G Network Slicing: Analysis, Algorithms, and Experimental Results," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2815-2828, Dec. 2018.

10. L. Zhang, F. Restuccia, T. Melodia and S.M. Pudlewski, "Jam Sessions: Analysis and Experimental Evaluation of Advanced Jamming Attacks in MIMO Networks," in *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc)*, Catania, Italy, July 2019.

ONR N00014-17-1-2046 - PROJECT 3 - Domain-specific Power-efficient Processing for Expeditionary Cyber Missions
Technical Leads: Gunar Schirner & Hamed Tabkhi

Abstract—**As human perception is often insufficient to rapidly assess any situation, enhanced cyber cognitive abilities are needed to automatically evaluate the vast sensory influx (video, radar, RF analysis, laser-based, and a myriad of scalar sensors). For example, machine vision is used for understanding scene structure, object classification and analysis for stationary (e.g. sensor network) or autonomous (on vehicle, robot) operations. Deep Convolution Neural Networks (DCNNs) are one of the most demanding advanced algorithms, offering unsupervised data analysis and pattern detection and making them ideal for vision, data mining and aggregation. To tame the overall computation for real-time scene understanding and not overwhelm the often scarce communication bandwidth, DCNNs processing close to the sensor becomes paramount. The challenge is that DCNNs are very computationally intense, with tens of Giga Operations per Second (GOPS) needing multi-Gigabytes (GB)/s of data traffic, resulting in 100s of watts of power usage on desktops. Deployment in expeditionary cyber missions, in contrast, demands reliable, real-time performance under extremely scarce resources which is far beyond the capabilities of state-of-the-art platforms (even on high-performance Digital Signal Processors (DSPs) and Graphics Processing Units (GPUs)). The myriad of desired sensors (and sensing modalities), each with individual analysis, shockingly amplify the challenges. This requires novel, high-performance, power-efficient processing to provide nimble, transparent computing capabilities for real-time DCNNs processing close to the sensors.**

To address the need this project lays the foundation for a modular implementation of streaming processors focusing on a deep neural network application. In addition, this project investigated into a domain specific language for composing data-centric applications.

Index Terms—**Design Space Exploration, Platform Allocation, ACC-Rich Platform, Streaming Applications.**

I. Overview

Addressing the vast challenges set out for domain-specific processing requires a holistic view across the entire stack from design, user application, system software, operating system extensions, hardware/software interfaces down to micro architectural innovations. This project moves toward providing solutions by addressing these challenges from two different angles. In a top-down view, we are looking at how design automation can help reasoning about domain processors and aim to decide which components to place into the processors, as well as how to efficiently scale a highly heterogeneous system composed of a dense set of accelerators. Conversely, at a bottom-up perspective, this project has investigated into micro-architectural approaches with a focus on Deep Neural Network processing.

G.Schirner is with the Department of Electrical and Computer Engineering, Northeastern University, MA, 02115, USA.

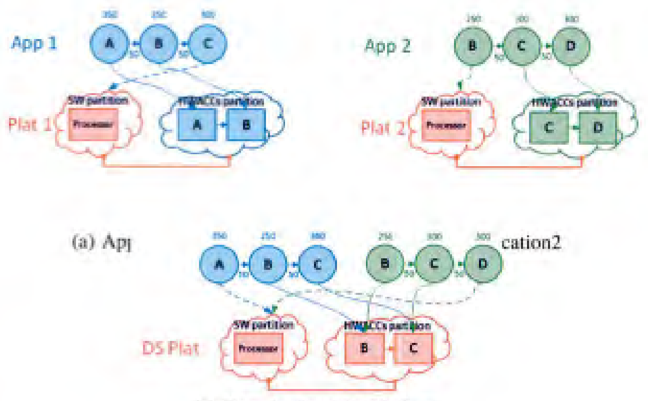
H. Tabkhi is with the Department of Electrical and Computer Engineering, University of North Carolina Charlotte (UNCC), NC, 28223, USA.

II. Top-Down Design Automation

A. *DS-DSE: Domain-Specific Design Space Exploration for Streaming Applications [1]*

Domain-specific computing is promising for high-performance low-power execution of applications with similar functionality. In particular, streaming applications with significant functional and structural similarities can tremendously benefit. However, current Design Space Exploration (DSE) focuses on individual applications in isolation. Hence, much of the domain optimization opportunities are missed. DSE methodologies need to broaden the scope from individual applications in isolation to optimizing across applications within a domain.

We introduce a novel Domain-Specific DSE (DS-DSE) approach for domain-specific computing with a focus on streaming applications. Our key contributions are: (1) a formalized method to extract the functional and structural similarities of domain applications, (2) a novel algorithm for hardware/software partitioning of a domain-specific platform to maximize the throughput across domain applications (under certain constraints) and (3) a methodology to evaluate a domain platform.



(c) Domain-Specific Platform

Fig. 1: Penalty of Application Scope

To gain an intuitive view of the domain concept, Fig. 1 illustrates the lost opportunities due to the limited DSE scope for a domain with two apps. Fig. 1a and 1b select ACCs with application scope (A,B and C,D respectively) yielding efficient execution on the own platform. However, executing an app on the foreign platform (i.e. $app1$ on $plat2$, $app2$ on $plat1$) results in significant penalties as either ACC A or D is not used. In result, the overall domain performance (all apps execute on same platform) is low.

Domain DSE can dramatically improve domain performance. Its aim is to detect and exploit common used kernels (e.g. function B and C) and composition (e.g. B-C in Fig. 1) across apps. When analyzing both applications, the common domain architecture (Fig. 1c) executes both applications efficiently.

Our work lays the foundations by defining a domain and quantifiable features (metrics) that can guide exploration. The features take both behavioral (processing) and structural (communication) aspects into account, as well as consider the distribution over the domain. Based on these definitions, the paper has introduced the Dynamic Score Selection (DSS) algorithm for domain exploration. The DSE maximizes throughput across the whole domain and creates a domain-specific architecture that has more flexibility to execute domain applications. Our results on 4 domains (OpenVX and 3 synthetic domains) demonstrate a significant performance improvement (36.8%-50.7%) executing on the respective domain architecture compared to application-specific architectures.

B. Alleviating Scalability Limitation of Accelerator-based Platforms [2]

Accelerator-based Chip Multi-Processors (ACMPs), which combine application-specific HW ACCelerators (ACCs) with host processor core(s), are promising architectures for high-performance and power-efficient computing. However, ACMPs with many ACCs have scalability limitations. The ACCs' performance benefits can be overshadowed by bottlenecks on shared resources of processor core(s), communication fabric/DMA, and on-chip memory. Primarily, this is rooted in the ACCs' data access and the orchestration dependency. Due to very loosely defined ACC communication semantics, and relying on general architectures, the resources bottlenecks hamper performance.

Current ACMP architectures have a processor-centric view as they were built upon the assumption of sparse integration of ACCs. ACCs are slaves devices requiring many shared resources (communication fabric, DMA, shared memory, and processor for coordination) for their transactions. However, power and performance efficient computing calls for more ACCs on a chip, which is not effectively supported by processor-centric architectures. Processor-centric architectures suffer from scalability limitations that restrict ACCs benefits. With integrating more ACCs, the burden on shared resources increases dramatically even though some ACCs logically communicate directly with each other. Therefore, there is a need for scalable architectures giving ACCs more autonomy.

To lay the foundations for improving the efficiency of ACC integration, we first identified the semantics of ACC communication. Then, we analyzed the impact of semantics aspects realization on ACCs' benefits to efficiently design our Transparent Self-Synchronizing Accelerators (TSS) architecture that reduces the load of ACC communication on shared resources. TSS gives autonomy to ACCs to self-synchronize and self-orchestrate each other independent of the processor, thereby enabling finest data granularity to reduce the pressure on the shared memory. TSS also exploits a local and reconfigurable interconnect for direct data transfer among ACCs without occupying DMA and communication fabric.

Fig. 2 shows a high-level overview of our architecture and its integration to the host processor through the shared memory and communication fabric. it contains a set of accelerators that communicate and synchronize directly with each other without the need for the processor intervention. The TSS is practically only visible through its gateway interface. The gateway receives the configuration information once from the processor(s), through the control bus, and sets up the interconnect at the beginning. During the application execution, the gateway reads large input data from shared memory, breaks it into smaller jobs to feed the chains. Conversely, it collects small resultant data from the chains, and writes a larger resultant data to shared memory.

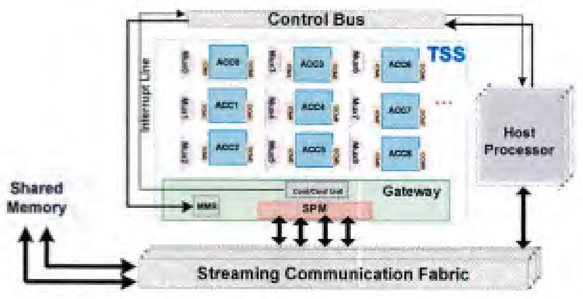


Fig. 2: TSS system integration

We used automatically generated virtual platforms to compare TSS and processor-centric architectures when running five streaming applications. With the same set of ACCs and same mapping, TSS improves throughput up to 1.6x at 20% ACC computation coverage and up to 130x at 100% ACC computation coverage. These benefits are achieved by natively realizing ACC-to-ACC communication in TSS, which reduces the load on shared resources by 6.57x and 328x, respectively. With ACC computation coverage and number of ACC-to-ACC connections increasing, the TSS benefits become more pronounced.

Overall, self-synchronizing architectures, such as the TSS are very promising to allow heterogeneous scaling and the proliferation of dense acceleration. Instead of exposing every single accelerator to the host CPU, only each concurrent stream (potentially being processed by many many accelerators) is exposed to the processor. This dramatically lowers the orchestration overhead and allows to create more efficient, powerful systems.

III. Bottom-Up Micro-Architecture

A. Background

To give a background on our motivation for specialized CNN implementation, we briefly overview data access types in CNN, and the differences between General Matrix Multiplication (GEMM) and direct convolution. We conclude with the motivation to focus on the first two layers of the CNN.

1) Data Access Types: Convolutions Neural Networks (CNN) are both memory and compute-intensive applications, often reusing intermediate data and while consistently doing millions of parallel operations. Furthermore, the inherent memory intensive aspects of the algorithm are further exaggerated due to complex multi-dimensional data accesses. In this regard, we consider two major types of data when performing CNN.

1) 2D Weight: The first type is 2D weight matrices. These weight matrices each correspond to a single channel and these channels weight matrices group together to construct the entire kernel. Multiple kernels form a layer, and multiple layers create a network topology.
2) Frame Pixels: The streaming pixels which are the input to the CNN processing. Just like the weight matrices these are 2D matrices, with multiple channels. This is the data that flows through the network topology.

2) GEMM vs Direct Convolution: Direct convolution is the point-wise Multiply and Accumulation (MAC) operation across the 2D weight Matrices and frame pixels. In direct convolution, similar to the algorithmic level definition, the weight Matrices are used to perform multiple multiply and then accumulation operations directly on the 2D window of input pixels. The direct convolution performs in a sliding window fashion with respect to a stride parameter that varies layer to layer in network topologies. Fig. 3 exemplifies direct convolution operation, for a 3 by 3 convolution window over a frame with 5 by 5 pixels.

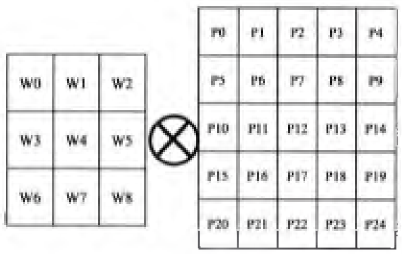


Fig. 3: Direct convolution

Traditionally, GPUs have seen much success in the cloud by using a linear algebra transformation called General Matrix Multiplication (GEMM) to lower the dimensions of convolution to regular matrix multiplication. GEMM transforms all the temporal parallelism into spatial parallelism. This helps GPUs to achieve a high throughput assuming the large data batches are available. However, this comes at a significant memory cost. The transformation is done by rearrangement with redundant copies of input image pixels. Our estimation reveals that the rearrangement results in 11X data duplication only for the first layer of any CNN network. This translates to significant power and energy overhead for accessing the redundant pixel data throughout memory hierarchy. Fig. 4 exemplifies GEMM operation, for the same example illustrated in Fig. 3. As we observe, redundancy of frame pixels is required to transfer the convolution operation to a large matrix multiplication. For this example, the pixels will be 9 by 9 compared to original frame size which is 5 by 5.

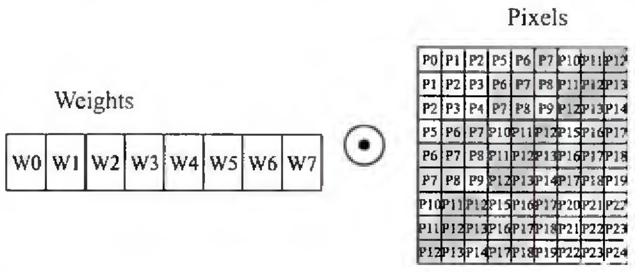

Pixels
Weights
W0 W1 W2 W3 W4 W5 W6 W7
P0 P1 P2 P5 P6 P7 P10 P11 P12
P1 P2 P3 P6 P7 P8 P11 P12 P13
P2 P3 P4 P7 P8 P9 P12 P13 P14
P5 P6 P7 P10 P11 P12 P15 P16 P17
P6 P7 P8 P11 P12 P13 P16 P17 P18
P7 P8 P9 P12 P13 P14 P17 P18 P19
P10 P11 P12 P15 P16 P17 P20 P21 P22
P11 P12 P13 P16 P17 P18 P21 P22 P23
P12 P13 P14 P17 P18 P19 P22 P23 P24

Fig. 4: General Matrix Multiplication (GEMM)

3) CNN Execution Bottlenecks: An initial focus for acceleration can be the first two layers of CNN as the major execution bottlenecks. We specifically target SqueezeNet, a DCNN design with memory effciency in mind. To motivate our argument, we have estimated the computation demands across the CNN layers for the example of . Fig. 5 shows the computation distribution across the SqueezeNet layers. Overall, SqueezeNet contains 10 layers. The first and last layers are traditional convolution layers (conv0 and conv9). The intermediate layers are squeeze (s) and expand (e) convolutional layers.

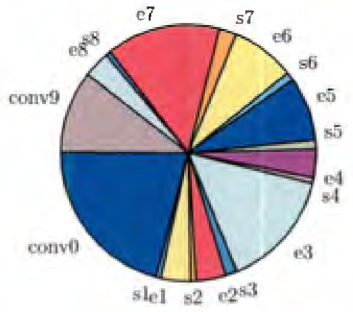


Fig. 5: Computation distribution across the SqueezeNet layers

The squeeze layers combine the feature maps to make the network more efficient and expand layers expand the feature map. As we observe, the first layer (conv0) has the highest computation demand with 21% contribution to overall computation demand. The first layer also generates the largest size of feature map across all layers which can lead to significant communication and memory traffic. Fig. 6 presents the contribution of layers on feature map. To minimize the memory access and communication demand, it would be beneficial to accelerate the second layer (s1, e1) along with the first layer. In this way, much smaller feature maps will be transferred to the edge server for processing of the remaining layers.

Direct convolutions have clear benefits, albeit they are less explored in current heterogeneous implementations. The next two subsections focus on two acceleration approaches for 1D and 2D.

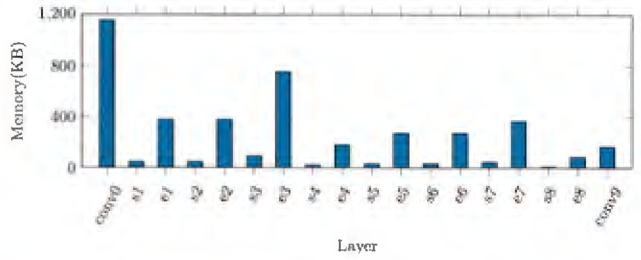


Fig. 6: Feature map distribution

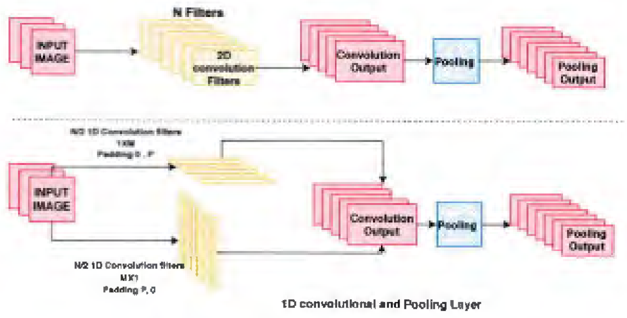


Fig. 7: Full first layer architecture

B. A Novel 1D-Convolution Accelerator for Low-Power Real-time CNN processing on the Edge [3]

With the rise of deep learning, the demand for real-time edge intelligence is greater than ever. Current algorithm and hardware realizations often focus on the cloud paradigm and maintain the assumption that the entire frames data is available in large batches. As a result, obtaining real-time AI inference at the edge has been a tough goal due to tight-latency awareness as well as streaming nature of the data. There is an inherent need for novel architectures that can realize latency-aware agile deep learning algorithms at the edge.

In this part of our work, we introduce a novel joint algorithm architecture approach to enable real-time low-power Convolutional Neural Network (CNN) processing on edge devices. The core of the proposed approach is utilizing 1D dimensional convolution with an architecture that can truly benefit from the algorithm optimization. On the algorithm side, we present a novel training and inference based on 1D convolution. On the architecture side, we present a novel data flow architecture with the capability of performing on-the-fly 1D convolution over the pixel stream.

1) 1D Convolution Algorithm Optimization: To dramatically reduce the computation and memory demand, we modify the first layer of convolution (which is the most compute-intensive layer) to utilize 1D convolution kernels, and consequently retrain these 1D kernels at the training stage. We used Caffe to train our 1D convolutions on an Nvidia Tesla P100. We simply changed the first layer to utilize 1D kernels and trained the rest of the network in conjunction with the new 1D layer. Once we got the new 1D-Squeeznet topology to converge, we then made an architecture for the inference stage that took advantage of our optimization and would run the trained weights.

Fig. 7 presents our 1D convolution optimization in detail. We propose separate filters for the X and Y dimensions. We then combine the results by concatenating the feature maps produced by vertical and horizontal convolutions (X and Y convolutions). As a result, there are M/2 parallel NX1 and 1XN filters. This is different from other works doing separable convolution, in which they are transforming a single 2D kernel into two equivalent 1D kernels. In this work we completely replace every 2D kernel with a singular 1D kernel. While we do expect accuracy loss (even with the network being trained with this optimization), this technique offers memory and computation reductions essential for real time edge execution.

2) 1D Convolution Architecture: We designed an algorithm-aware streaming architecture. We configure our architecture based on the natural parallelism of CNNs, with this work focusing on Kernel parallelism. Finally, this architecture can be reconfigured to map to any network topology. These novel contributions of our architecture, allow us to handle the first layers high FM data size and intensive computation, in an efficient manner. The proposed hardware accelerator designed to do inference analytics at the edge has three main parts: (1) Convolutional Processing Element (CPE) to perform convolution on weights and image pixels. (2) Aggregation Processing Element (APE) to sum outputs of convolution of Red/Green/Blue channels and convert negative values to zeros. (3) Pooling Processing Element (PPE) that performs max-pooling on the output of APE. In this subsection we will discuss these parts in the context of 1D convolution.

To implement the first layer with our architecture for SqueezeNet, there should be one CPE per each input channel i.e. 3 CPEs in total. According to SqueezeNet topology there are 96 kernels in the first layer, which means each CPE should have 96 MACs each one with its own output(48 for horizontal 48 for vertical convolution). This is necessary to implement kernel parallelism (where all the kernels are used at the same time). Each of these CPEs outputs are added together in APEs. There should be one APE per kernel which makes 96 APEs in total. Outputs of these 96 APEs go to 96 independent PPEs, which are located right after APEs. Fig. 8 shows a detailed illustration of the proposed architecture for the first layer of SqueezeNet topology (lacking the extra information provided in the vertical and horizontal component figures).

Overall, one dimensional-training and inference can lead to significant theoretical reductions in the memory and computation demand of CNN layers. In the case of SqueezeNet, 96 convolutions of 7X7 filters are replaced with 48 sets of 7X1 and 48 sets of 1X7 filters, which finally output 96 concatenated feature maps. This reduces the number of parameters in the first layer by 7x. The algorithm optimization translates to a 7 times smaller kernel memory (2016 bytes). The total memory we need to store pixels before convolution is 1253 bytes, reducing the buffer size by almost half, and the operation frequency that drives the computation of our architecture is 7 times smaller due to the reduced number of operations per convolution.

We have implemented a hardware/software co-design solution on a Xilinx Zynq-7000 FPGA. Our results demonstrate that when the 1D algorithm optimization is adapted to our algorithm-aware architecture we only consume 16mW of

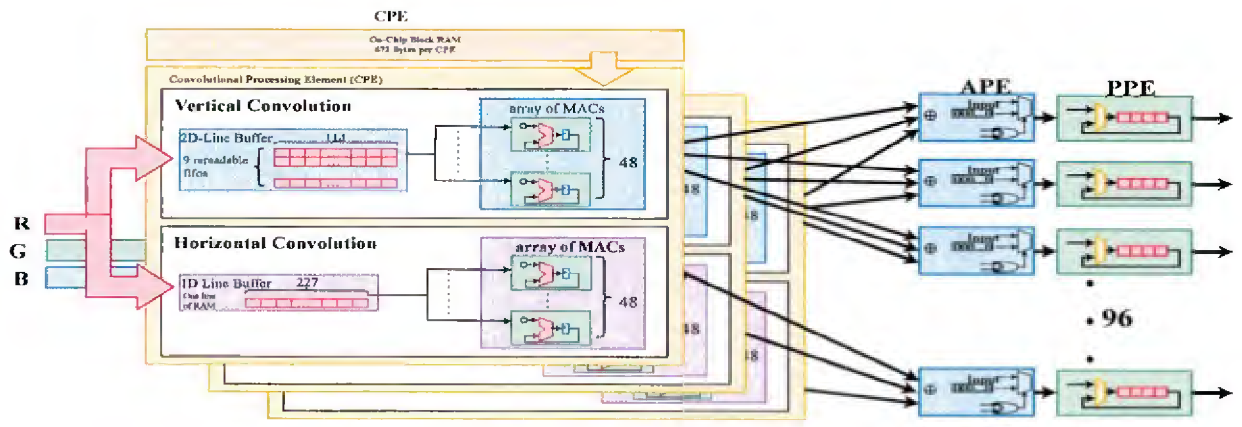


Fig. 8: Full first layer architecture using 1D convolutions

dynamic power (custom logic alone, not the entire system), a 7.3X power reduction compared to the original 2D design. When comparing the full system implementation to a GPU Implementation we achieve a low power consumption of only 1.73W, resulting 4.3x the power savings. Finally in both designs our architecture is able to support 60 FPS, while the Mobile GPU is unable to do so.

C. A Reconfigurable Streaming Processor for Real-Time Low-Power Execution of Convolutional Neural Networks at the Edge [4]

With the recent advances in machine learning and the deep learning paradigm, there is a huge demand to push the data analytics and cognitive inference to the edge of the network near the data producers and sensors. Edge analytics are essential for real-time video analytics and situational awareness; which is required for the wide range of cyber-physical applications such as smart transportation, smart cities, and smart health. To this end, novel architectures and platforms are required to enable real-time low-power deep learning execution at the edge. This paper introduces a novel reconfigurable architecture for real-time execution of deep learning and in particular convolutional Neural Networks (CNNs) at the edge of the network, close to the video camera. The proposed architecture offers a set of coarse-grain function blocks required for realizing CNN algorithms. The macro-pipelined datapath is created by chaining the function blocks with respect to the topology of the target network. The function blocks operate over the streaming pixels (directly fed from the camera interface) in a producer/consumer fashion. At the same time, function blocks offer enough flexibility to adjust the processing with respect to area, power, and performance requirements. This paper primarily focuses on the two first layers of CNNs as the two most compute-intensive layers of CNN network.

This section introduces our proposed architecture template, for real-time execution of CNN inference on the edge. The proposed template targets FPGA devices, as they offer both efficient execution and sufficient reconfigurability to cope with continuously growing CNN topologies. Furthermore, by targeting the FPGAs, we are able to generate a customized datapath per each CNN network as such to best fit the processing requirements. The major premise of our proposed architecture is to remove the gap between the algorithm execution semantic and architecture realization. Therefore, our proposed architecture is primarily a data flow machine working on streaming data based on direct convolution. It consists of three main function blocks for realizing the wide range of CNN inference topological structure. The blocks are Convolutional Processing Element (CPE), Aggregation Processing Element (APE), and Pooling Processing Element (PPE). The blocks will be configured and connected with respect to target network topology, creating a macro-pipeline datapath. Fig. 9 presents overall architecture realization from logical domain (algorithm) to physical domain (architecture).

Our architecture is designed based on the natural dataflow of CNNs. It is able to exploit both spatial parallelism across the convolutions within the same layer, as well as temporal parallelism between the blocks across the layers. The blocks are configurable with respect to network parameters such as size of convolution and stride. This gives us the possibility of easily adapting the architecture to any desired network topology.

While our proposed architecture template is extensible and can support the entire CNN topology, the primary limitation is available hardware resources on FPGAs of the edge devices. At this moment, we are targeting smaller FPGAs, e.g. Xilinx Zynq, with small reconfigurable fabric. However, by accelerating the first two layers on the edge node, we will able to relax the computation demands on the edge server. The edge node will perform the heavy computation of the first layer. Furthermore, it runs the second layer to significantly shrink the feature map. Then it sends the feature maps to the edge server for the remaining layers to do the processing.

The proposed novel architecture template for real-time low-power execution of Convolutional Neural Networks at the edge is primarily targeted for FPGAs, and is able to offer configurable macro-pipeline datapath for scalable direct convolutions over streaming pixels. The proposed architecture is an example of a hybrid solution across edge nodes and edge servers for realizing compute-intensive deep learning applications. It is

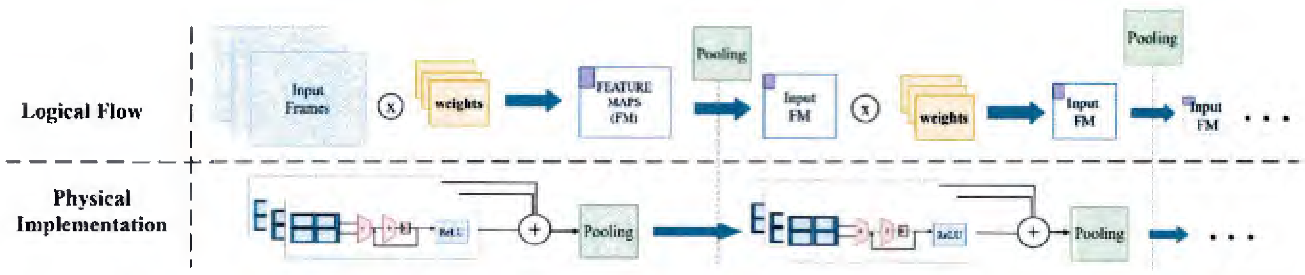


Fig. 9: From algorithm composition to architecture realization

also able to reduce the network traffic and execution time of the overall application. At the same time, it maintains the flexibility to map to any standard CNN network topology. Future work includes supporting full network topology acceleration on edge and supporting nonstandard CNN, as well as a workflow for mapping them efficiently to different FPGAs.

The implementation has extremely promising results. When mapped to a Xilinx Zynq FPGAs, for the first two layers of the SqueezeNet Network, shows 315mW power consumption when designed at 30 fps, with only a 0.24 ms one-time-latency. In contrast, the Nvidia Tegra TX2 GPU is limited to perform at 32.2 fps due to the 31.4ms delay, with a much higher power consumption (7.5 W).

IV. Conclusion

The work performed on this project shows extremely promising results for domain-specific computing. Our key insight to increase efficiency is to minimize data movements at any architecture hierarchy level, exploit parallelism (spatial and temporal) across convolutions, and remove the overhead of instruction-level programmability while maintaining enough flexibility. We have approached this challenge from a top-down design automation perspective, as well as from a bottom-up architecture view.

In a broader perspective, our work introduces a new class of processors with efficiency comparable to custom hardware accelerators and sufficient flexibility to perform various applications/configurations within a domain of applications. This research opens a path to perform real-time complex stream processing near the sensors offering human-like and beyond human cyber cognitive abilities.

The PIs would like to thank ONR for their geneours support of our work.

References

[1] J. Zhang, H. Tabkhi, and G. Schirner, "Ds-dse: Domain-specific design space exploration for streaming applications," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 165–170.

[2] N. Teimouri, H. Tabkhi, and G. Schirner, "Alleviating scalability limitation of accelerator-based platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 7, pp. 1317–1330, July 2019.

[3] J. Sanchez, N. Soltani, R. Chamarthi, A. Sawant, and H. Tabkhi, "A novel 1d-convolution accelerator for low-power real-time cnn processing on the edge," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, Sep. 2018, pp. 1–8.

[4] J. Sanchez, N. Soltani, P. Kulkarni, R. Chamarthi, and H. Tabkhi, *A Reconfigurable Streaming Processor for Real-Time Low-Power Execution of Convolutional Neural Networks at the Edge*, 06 2018, pp. 49–64.

Gunar Schirner (S'04–M'08) holds PhD (2008) and MS (2005) degrees in electrical and computer engineering from the University of California, Irvine. He is currently an Associate Professor in Electrical and Computer Engineering at Northeastern University. His research interests include the modelling and design automation principles for domain platforms, real-time cyber-physical systems and the algorithm / architecture co-design of high-performance efficient edge compute systems.

Hamed Tabkhi received his PhD degree in electrical and computer engineering from Northeastern University, Boston, USA, 2014. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, William States Lee College of Engineering, the University of North Carolina Charlotte (UNCC). His research interests span a wide range of areas including high-performance low-power architectures, embedded vision computing, system-level design and modeling for market-oriented MPSoCs. He currently focuses on transformative computer architecture and system solutions to bring the recent advances in AI and data analytics to enhance the safety, security and overall well-being of our community.

ONR N00014-17-1-2046 - PROJECT 4 - Robust Localization and Time Synchronization
Technical Lead: Guevara Noubir

Abstract. Localization and time synchronization are key elements of a variety of communication, networking, and situational awareness tasks. We consider five objectives, self-localization, localizing a friendly user (e.g., search and rescue), localizing an adversary (e.g., jammer or IED operator), localizing an event (e.g., gunshot/sniper), and localizing a user/event in the presence of an adversary. In this context, the adversary might be trying to hide its presence or location, and/or disturb the localization/time synchronization of legitimate users. There is today no solution that satisfactorily addresses this problem by algorithmically and robustly integrating signals and information at the *edge* of the network from and by a distributed set of sources. For instance, it is today easy today to make a device believe that it is located at a distance location by spoofing radio signals.

In the highly dynamic and unpredictable environment envisioned for Expeditionary Cyberspace, it is critical to opportunistically exploit every single source of information (e.g., GPS, Cellular, Wi-Fi, RBDS, sound, crowd-sourced measurements including radio/audio) to improve accuracy, speed, and resiliency to injection of malicious information. We have made progress on several fronts: (1) LTE sniffer for cellular devices localization and machine learning techniques for devices classification, (2) Wi-Fi devices fingerprinting, (3) anti-jamming/spoofing for friendly devices localization, and (4) time synchronization using carrier frequency offset estimation.

LTE Sniffer and Cellular Devices Localization

We developed a software defined radio sniffer of LTE radio communications (for both the downlink and uplink). The sniffer analyzes eNodeB downlink transmissions, extracting unencrypted meta-data such as radio block allocation as well as other device specific information. The sniffer can then infer the existence of LTE devices as well as other information unique to these devices. This is the basis for our LTE devices localization using multiple multi-antenna sniffers. For instance, Figure 5 illustrates an LTE Downlink Frame (10ms). Subframe 5 contains information about the downlink transmission using Radio Network Temporary Identifier (RNTI) 65534 indicating paging of devices (it also indicates modulation and coding scheme 0 (QPSK) using 8 (out of 75) resource blocks for a total of 104 bits). The paging message provides information about LTE devices. This is a lists of TMSI (Temporary Mobile Subscriber Identity) which rarely changes, or IMSI (International Mobile Subscriber Identity) in case of unallocated TMSI or network failure).

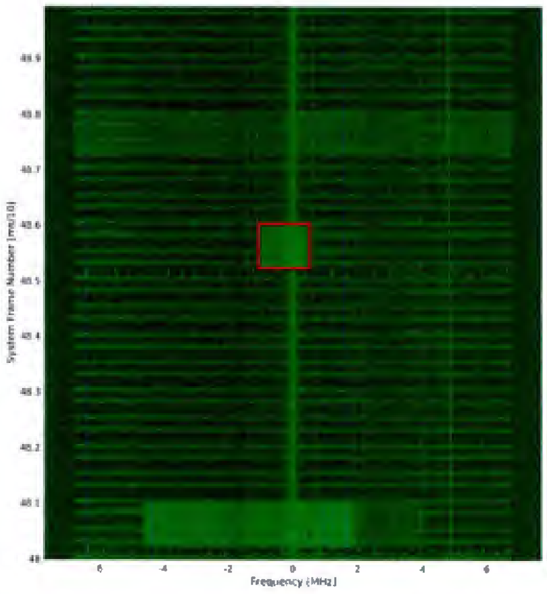


Figure 1. LTE Downlink Frame (10ms) captured using our sniffing tool. Subframe 5 contains the paging information that can identify LTE devices.

Further analysis of the LTE Frame enables the continuous tracking of traffic of LTE devices (See Table 2). The amount of traffic (both uplink and downlink) enables the fingerprinting of LTE/Mobile devices at the application level. For instance, it becomes possible to create a signature for a device based on the set of mobile apps and their usage patterns.

Time		RNTI	Info	MCS	RBn	TBS
120	3	6681	0	11	1	144
120	5	65535	1	0	8	208
120	6	6681	0	12	4	776
120	9	10451	1	12	8	1608
120	9	10451	0	22	5	2344
121	0	10451	1	15	63	18336
121	1	10451	1	14	51	12960
121	1	8094	1	11	4	1904
121	2	10451	1	15	75	21384
121	3	10451	1	15	56	15840
121	4	10451	1	15	72	20616
121	7	10451	1	15	72	20616
121	8	10451	1	15	63	18336
121	9	10451	1	14	51	12960
122	0	10451	1	15	40	11448
122	1	10451	1	15	75	21384
122	2	10451	1	15	56	15840
122	2	6681	1	0	4	88
122	3	10451	1	15	72	20616

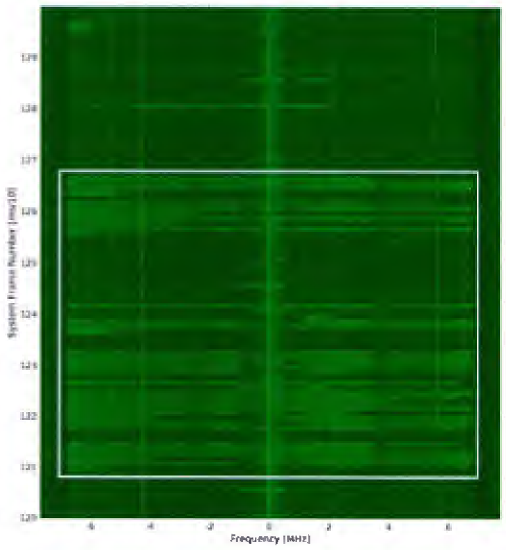


Table 1. Analysis of an LTE Frame enables the identification of traffic of devices and therefore application level fingerprinting.

We developed a set of techniques that can fingerprint a mobile device based on the applications running on the phone. This fingerprinting is entirely based on the passively sniffed radio traffic. We initially focused on the passive activity of the apps on the phone as such patterns are easier to classify. Figure 7 illustrates the amount of traffic between an LTE eNodeB (base station) and UE (mobile) measured on our private testbed.

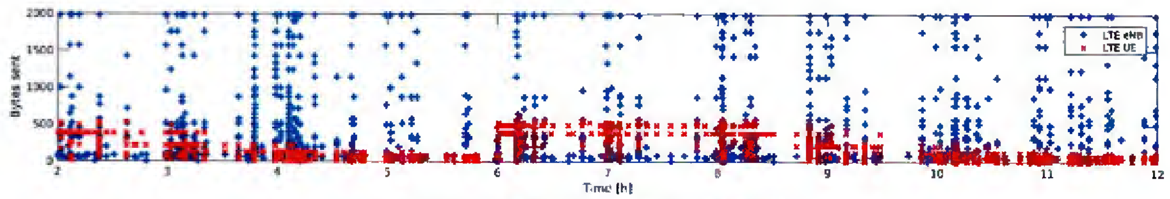
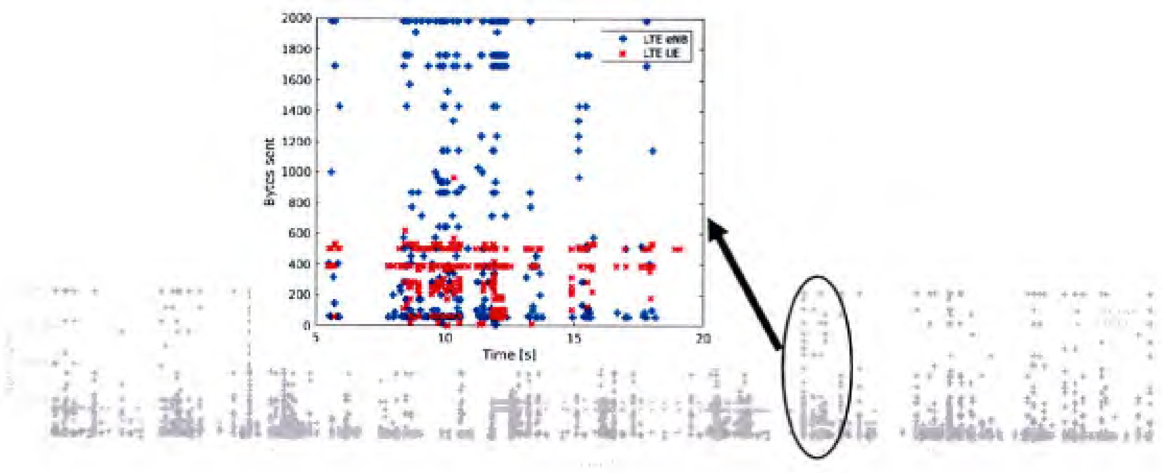


Figure 2. Sniffed traffic (uplink & downlink).

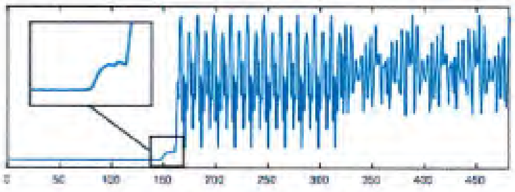
It then becomes possible to use the RF trace to isolate communication patterns that are specific to a phone. The developed techniques include clustering traffic, and processing (e.g., PCA) the sniffed traffic unencrypted meta-information to facilitate the training and learning on aggregate traffic.



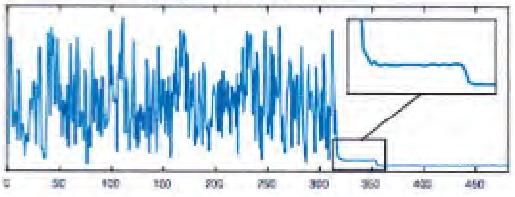
Our initial results indicate the potential for achieving high accuracy in identifying a mobile device by the set of installed applications relying only of passive sniffing of radio emissions.

Wi-Fi Fingerprinting

We developed techniques for fingerprinting Wi-Fi radio interfaces which include carrier frequency offset (CFO), sampling frequency offset (SFO), Wi-Fi scrambling seed, and RF front end ramp-up/down signatures. The project goal is to first develop radio interface unique fingerprinting/localization techniques and then combine cellular and Wi-Fi fingerprinting and localization to robustly locate wireless devices.



(a) Transient at frame start



(b) Transient at frame stop

Figure 3. RF Front End transients are fairly unique and allow the fingerprinting of a radios.

The developed techniques are a combination of machine learning, clustering based on Kullbak-Liebler divergence between distributions of CFO/SFO. Our evaluation in the wild of 100 Wi-Fi interfaces resulted in 65% to uniquely identify a device (out of 93 devices) and 93% to distinguish one of 5 interfaces.

Robust friendly localization and time synchronization

On the theoretical side, we are integrating cryptographic techniques to mitigate timing and localization spoofing (TESLA) and mitigate jamming (keyless spread spectrum) in the protocols design of new time synchronization and localization techniques designed from the ground-up. The proposed technique recursively refines timing and location inference combining both the keyless spread spectrum with TESLA authenticated broadcast technique.

We developed a synchronization, localization, and mobility (velocity) estimation technique based on carrier frequency offset between devices. The technique was implemented within the USRP X310 FPGA. Preliminary evaluation indicates that the developed technique has the potential to achieve a time synchronization accuracy of few nanoseconds.

Future Activities

We plan to extend the synchronization and localization to a network of devices to develop an efficient secure broadcast localization scheme for both cooperating devices and RF sources. We also plan to integrate the TESLA broadcast authentication techniques to defend against spoofing from insider threats (compromised devices).

ONR N00014-17-1-2046 - PROJECT 5 - LANET: Visible-Light Infrastructure-less Wireless Networks for Expeditionary Cyber Missions
Technical Lead: Tommaso Melodia

Dept. of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115

1 Research Goal

The LANET project aims at investigating fundamental principles underlying the design of infrastructure-less ad hoc networking based on Visible Light Communications (VLC). LANET will provide the network researchers with a real-time software-defined programmable protocol stack. LANET will adapt to diverse networking environments (e.g., air/ground/underwater) by seamlessly switching among different front-end transceivers. To achieve this objective, the project will (1) study and develop new resource allocation algorithms and protocols for visible-light ad hoc networking; (2) design the LANET algorithmic and architectural framework encompassing all layers of the networking protocol stack and (3) develop a wavelength-agile software-defined experimental prototype, including an intelligent control system that can leverage the hybrid visible light, ultraviolet and RF transceivers to maximize system reliability and security.

2 Accomplishments

Toward this goal, our team has so far the following accomplishments.

LANET Architecture Design. At a high level, the LANET framework consists of two main components: the LANET hardware and a LANET programmable protocol stack. The LANET hardware is designed to enable wavelength-agile (RF/VLC/UV) signal processing and is built off of high performance software-defined radios (i.e., USRP X310) that includes a Large customizable Xilinx Kintex-7 FPGA and supports two-channel processing concurrently. The LANET software framework implements newly-designed LANET networking functionalities across multiple layers of the protocol stack in a software-defined fashion to enable real-time intelligent adaptation to the operational environment. The protocol stack has a modular structure, where different functional blocks, such as timing functionalities, medium access control, routing, and application adaption (air/underwater/ground), among others, can be designed and upgraded independently and in a modular fashion.

VL-MAC: Channel-aware VLC MAC design. Because of the unique characteristics of visible light wireless links, i.e., directionality (different LEDs have different fields of view), low penetration (which results in more frequent blockage), as well as strict Tx-Rx alignment requirements, networking protocol stacks for visible light communications cannot be blindly drawn from their RF counterparts. To address these challenges, we designed what is, to the best of our knowledge, the first medium access control (MAC) protocol uniquely designed for LANETs. The proposed utility-based MAC is based on a new sector-based neighbor discovery mechanism suitable for LANETs and relies on synchronization and opportunistic link establishment. The proposed synchronization algorithm enables nodes to know when an idle neighbor will become available, thereby overcoming the problem of deafness. A utility based opportunistic three-way handshake is employed to efficiently negotiate medium access. Specifically, a node chooses the optimal transmission sector to maximize the probability of establishing a link when a fraction of the neighbors are affected by blockage or deafness. Simulation results have demonstrated up to 48% increase in throughput compared to state-of-the-art techniques.

VL-ROUTE: A Cross-Layer Routing Protocol for Visible Light Ad Hoc Network. To overcome the unique challenges like blockage and deafness that render routes in LANETs highly unstable, we propose a cross-layer optimized routing protocol (VL-ROUTE) that interacts closely with the MAC layer to maximize the throughput of the network by taking into account the reliability of routes. To accomplish this in a distributed manner, we carefully formulate a Route Reliability Score (RRS) that can be computed by each node in the network using just the information gathered from its immediate neighbors. Each node computes an RRS for every known sink in the network. RRS of a given node can be considered as an estimate of the probability of reaching a given sink via that node. The RRS value is then integrated to the utility based three-way handshake process used by the MAC protocol (VL-MAC) to mitigate the effects of deafness, blockage, hidden node, and maximize the probability of establishing full-duplex links. All these factors contribute towards maximizing the network throughput. Simulation of VL-ROUTE shows 124% improvement in network throughput over a network that uses Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) along with shortest path routing. Additionaly, VL-ROUTE also showed up to 21% improvement in throughput over the network that uses VL-MAC along with a geographic routing.

LiBeam: Cooperative Beamforming for Visible Light Networks. Current research has revealed that Tx-Rx dis-alignment in highly directional VLC links can result in up to 10 dB of signal-to-noise-ratio (SNR) variation, which in turn has a significant impact on the link reliability in terms of packet error rate (PER). Therefore, how to ensure robust and reliable communication between the VLC transmitter and receiver under visible light link dynamics (e.g., intermittent blockage, mobility, device misalignment) becomes critical. We proposed LiBeam, a new cooperative beamforming scheme to maximize the system throughput, based on forming multiple LED clusters. Each cluster then collaboratively serve the same set of users by jointly determining the user-LED association strategies and the beamforming vectors of the LEDs thus reducing the interference among adjacent users hence enhancing the quality of the visible light links. We first propose a mathematical model of the cooperative beamforming problem, presented as maximizing the sum throughput of all VLC users. Then, we solve the resulting mixed integer nonlinear nonconvex programming problem by designing a globally optimal solution algorithm based on a combination of branch and bound framework as well as convex relaxation techniques. Performance evaluation results show that over 95% utility gain can be achieved compared to suboptimal network control strategies.

Testbed Development. A newly designed general purpose LANET testbed has beend developed based on high performance software-defined radio devices (USRP X 310). The testbed supports multihop end-to-end dual-band RF/VLC data transmission and features a brand new programmable protocol stack that covers physical layers, link layer, network layer, transport layer and application layer. The testbed features a scalable out-of-band control channel (UDP socket based), which provides multi-layer acknowledgements, signaling, and control information (e.g., neighbor discovery, routing table setup and exchanges), among others. So far, the following functionalities have been implemented on the LANET testbed.

Automatic Routing Set-up: Routing at the network layer plays a significant role on the performance of LANET and have a major influence on the overall network throughput. The multi-hop routing design for visible-light ad-hoc networking is challenging due to the intermittent links. We first implemented a simple reactive routing protocols in 2-hop LANET testbed to investigate the link characteristics of visible-light ad hoc networks. The filed of views of the transceivers are limited which slows down the progress of neighbor discovery and routing set-up. An omni-directional transceivers have been proposed to overcome the above-mentioned issue.

Intelligent Switching in Heterogeneous Radio and Visible-Light Ad Hoc Networks: Visible light communication can provide very high aggregate capacity; however, VLC is challenged to accommodate highly dynamic environments. Specifically, the VLC channel is susceptible to blockage and the smaller coverage region implies that devices with high mobility will change connections frequently. In order to mitigate the impact of these limitations, heterogeneous networks (RF/VLC/UV) has been proposed and developed in our LANET project, which combines the aggregate capacity gains of VLC/UV with the coverage and reliability of RF. We developed an intelligent switching algorithm based on three-way handshake process when no routes available in the VLC domain after a predefined time window, a switching request to RF will be initiated, or otherwise.

Cooperative Transmission: To test the proposed LiBeam on LANET testbed, we implemented the cooperative transmission functionality among multi VLC users on LANET. The number of cooperative transmitters can be configured through program in a real-time fashion, where the OctoClock CDA-2990 is used to synchronize a system of USRP X310s for coherent operation, like cooperative beamforming in LiBeam. The functionality will be useful for many other applications such as diversity combining, or VLC MIMO transceiver design.

The testbed has been used to validate the effectiveness of the proposed cooperative beamforming scheme, LiBeam. It will also allow us to test the effectiveness of the designed LANET architecture by demonstrating VL-MAC, VL-ROUTE, self-routing and intelligent switching functionalities in multihop ad hoc networks. Currently, our testbed can support transmission distance upto 4 m with data rate 1 Mbps.

3 Publications

1. N. Cen, J. Jithin, M. Simone, Z. Guan, T. Melodia, "LANET: Visible Light Ad Hoc Networks," Ad Hoc Networks (Elsevier), 2019.
2. J. Jagannath, T. Melodia, "VL-ROUTE: A Cross-Layer Routing Protocol for Visible Light Ad Hoc Network," in Proc. of IEEE Symp. on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM), Washington D.C., USA, June 2019.
3. N. Cen, N. Dave, E. Demirors, Z. Guan, T. Melodia, "LiBeam: Throughput-Optimal Cooperative Beamforming for Indoor Visible Light Networks," in Proc. of IEEE Computer Communications (INFOCOM), Paris, France, April 2019.
4. J. Jithin, T. Melodia, "An Opportunistic Medium Access Control Protocol for Visible Light Ad Hoc Networks," in Proc. of International Conference on Computing, Networking and Communications (ICNC),, Maui, Hawaii, USA, March 2018.

ONR N00014-17-1-2046 - PROJECT 6 - Sensing, Computation and Communication on the Fly: Connected UAV Mesh Networks Technical Lead: Kaushik R. Chowdhury

Electrical and Computer Engineering Department, Northeastern University, Boston, MA, USA

Abstract

Unmanned aerial systems (UASs) allow easy deployment, three-dimensional maneuverability and high reconfigurability, as they sustain communication network in the absence of pre-installed infrastructure. The proposed FOg Computing in UAS Software-defined mesh network (FOCUS) paradigm aims to realize an implementable network design that considers practical issues of aerial connectivity and computation. It allocates UASs to the tasks of data forwarding and in-network fog computing while maximizing number of ground-users in UAS coverage. FOCUS improves efficient utilization of network resources by introducing on-board computation and innovates on top of software-defined networking stack by integrating the capabilities of network and ground controllers to enable simultaneous orchestration of both UASs and communication flows. There are three main contributions through this project: First, an SDN-based architecture is designed enabling autonomous configuration of computation and communication as well as managing multi-hop aerial links. Second, a global optimization problem to achieve optimal forwarding and computational allocation is formulated using Open Jackson Network model and solved via a heuristic approach with well defined complexity. Third, FOCUS framework is implemented on a small-scale testbed of Intel® Aero UASs performing image analysis with a full software stack. Experiments reveal at least 32% latency improvement in computation service time compared to traditional centralized computation at the end-server or greedy task allocation schemes within the network.

I. Introduction

Unmanned aerial system (UAS) applications have grown exponentially over the last five years, presently driving business close to 1 billion USD already within the USA, with an upwards growth targeted to reach 46 billion USD by 2025 [1]. Thus, it is foreseen that they will be one of the key enablers toward smart cities with their applications range from construction to communication and to surveillance. However, most of the existing deployments consider UAS as a mobile wireless sensor, with data processing offloaded to a computational cloud. At the same time, the enhancements in UAS control and communication hardware and mass production at economical price-points are paving the way towards Unmanned Aerial Networks (UANs), composed of swarms of UASs connected to the Internet, capable of limited on-board computation, but also being integrated with the cloud [1].

Different from traditional ad hoc and mobile networks, the design of a UAN poses unique challenges, such as highly dynamic topology, 3D mobility and high energy consumption per unit time [2]. Piloting commercial off the shelf (COTS) UASs requires manual skills, but UAN applications that have rigid objectives and performance constraints amplify the above challenges when they operate in groups. In this project, we envision the UAN as a mixed sensing, information relaying and computing platform, taking advantage of the entirety of its on-board capabilities. Towards this goal, we wish to adopt the flexibility and structured approach of classical software defined networking (SDN), building on the OpenDayLight [3] architecture that has proven to be successful in the wired networking domain. Thus, each UAS within the larger UAN becomes a network switch that directs data traffic towards the remote cloud as well as towards peer-nodes for in-network processing. To our best knowledge, while many works have pointed towards the trend in COTS UAN to gain increasingly greater computational power and ability to support popular operating systems and processing packages [4], a transformative design that allows the UAN to become a fully capable, aerial SDN has not been implemented in actual systems.

A. Problem and Solution Overview

We consider a scenario where ground sources may generate rich sensing data (e.g. videos, terrain maps, RF spectrum surveys) that needs to be transmitted to the cloud for purposes such as storage, aggregation and analytics as seen in Fig. 1. This generic environment covers several real-world use-cases related to post-disaster recovery, rural broadband access and military operations. In many of these cases, the cloud is not reachable via a direct link, and hence, instead of a single UAS, we envision that a UAN is deployed for data forwarding to the cloud server. We note that several distributed mobility-aware routing and communication-aware mobility control schemes for UASs have been proposed in the literature that address only a subset of factors affecting the UAN deployment [2] [5]. In this project, we focus on an architecture that leverages SDN, given that it effectively decouples control and user data plane, and has shown great potential in management of wired/wireless networks.

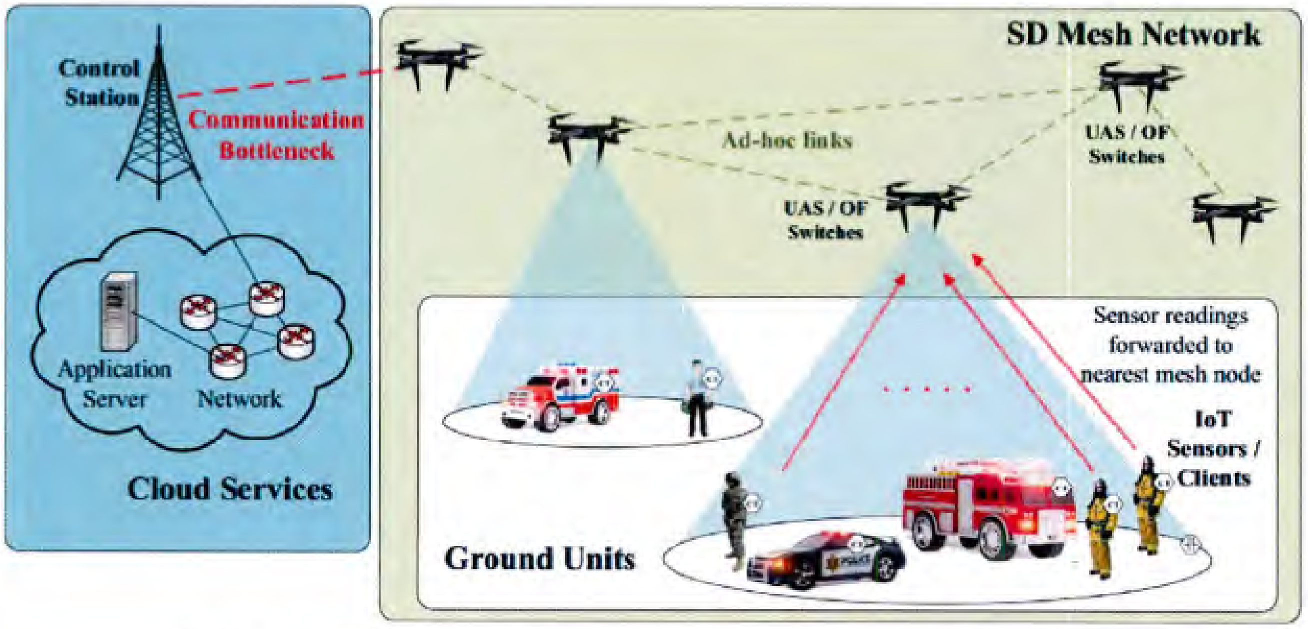


Fig. 1: Network architecture of a UAN connected to computational cloud and ground units.

A conceptual view of SDN controllers to compute the UAN topology and routing based on actual link load estimations have been proposed in [6] [7]. In addition, given the highly dynamic network topologies in such networks, there is likelihood of a temporal drift between the actual network state and the virtual state information available at the controller. In such situations, shifting the entire burden of the scenario analysis to the cloud becomes risky; few incorrect or outdated decisions taken by the controller can result in major end-to-end latencies in the mission [8]. Our approach addresses these challenges by considering, in a single theoretical formulation, the optimized assignment of the individual UASs to function as forwarding nodes that relay data towards the sink or to alternate intermediate-UASs that may perform computational tasks closer to the source. This results in significantly lowering the volume of data to be transmitted through the UAN. We look at the problem not only from a theoretical viewpoint but also from that of practical implementation, by (*i*) implementing a robust mesh network formation and routing protocol, (*ii*) defining and architecting the SDN control interfaces, and (*iii*) showing the benefits of intelligently assigning computation tasks to selected UASs in the network.

B. Main Contributions

We make three main contributions in this project:

- We present the architecture and design of FOCUS, a framework for the deployment of a practical UAN. It autonomously sets up the UAN configuration in terms of network topology (i.e. routing table), and computation (i.e. task allocation over cloud or intra-UAN nodes, also called as *fog* nodes), based on QoS requirements. This is performed via an SDN architecture that handles both computation and communication in multi-hop aerial environments.
- We formulate the problem of joint routing and computation assignment over UAN via an analytical model. Being NP-hard, we decompose it into two sequential tasks, with heuristic solutions for each. We also provide rigorous bounds on the computational complexity.
- We implement the system on a limited testbed of Intel Aero UASs performing data processing of image-data. We also validate FOCUS experimentally and through simulation, starting from the data generation on the ground to the final logging of processed results in a database.

The rest of this report is structured as follows. In Section II, we review the existing literature on the application of SDN and cloud computing on UAN. Preliminary results supporting the motivation of our work are provided in III. The joint routing-computation problem and algorithms are formally described in IV. FOCUS architecture and implementation details are explained in V. Experimental and simulation results are reported in Section VI. Finally, we conclude in Section VII.

II. Related Work

Recent works have attempted to address UAS requirements of dynamic topology and variable network load [2]. Moving some of these important problems from the physical device plane to the controller of an SDN has gained traction [9], where link status and flight statistics are collected from each UAS and used to compute the routing tables. The core functionality of packet routing is enhanced in [10] by incorporating a centralized energy- and load-aware routing scheme. [6] maps these approaches in context of video-surveillance, while [7] exploits the closely related Network Function Virtualization (NFV) functionalities for telemetry monitoring and anomaly detection. [8] uses SDN controller for motion and location prediction, by utilizing knowledge of the current physical position and trajectory of each UAS to envision how the network may evolve ahead.

Different from these works, FOCUS SDN-enabled architecture jointly performs network routing to both the end cloud as well as in-network computational resources while taking the network topology and the network load requests into consideration. The latter concept, commonly known as *fog computing* is still in a nascent stage. Several papers have investigated the dual problem, i.e., how to offload the computation from the UASs to an edge-server or to a remote server in the cloud [11]. [12] and [13] describe video-surveillance applications, where data are gathered by UASs, and processed on edge nodes through computational expensive vision algorithms. Instead, in FOCUS, we envision UAS themselves as computationally capable devices. The Intel Aero drones we use have Intel Atom® x7-Z8750 processor, 4 GB LPDDR3-1600 RAM, Intel® Dual Band Wireless-AC 8260, 32 GB eMMC, Altera® Max® 10 FPGA and Ubuntu 16.04 LTS.

	Study	Network Aspect	Computational Aspect	Target Domain	Test-bed Imp.
UAS app. with on-board processing	Motlagh et al. [12]	Direct link to LTE infrastructure for communication	Crowd survailence with on-board processing with a single drone	IoT Platform	✓
	Wang et al. [13]	Adaptive relaying mechanism with full-duplex radios on UAS	Real-time video streaming via UAS to a ground server without pre-processing on-board	Multi-UAV network	✓
Vehicular Cloud	Meneguette et al. [14]	Poisson arrival model for vehicles entering to a cloud without stressing network or comms. challenges	Formulating semi-markov decision process to allocate resources based-on availability	Vehicular Networks	✗
	Hui et al. [15]	Probabilistic V2I and V2V link model with two-layer relay selection scheme	Content dissemination framework among edge computation nodes to increase availability	Content Delivery Networks / Vehicular Networks	✗
	Florin et al. [16]	N/A Network challenges are not addressed	Approximation of job completion times on distributed systems	Distributed Systems / Vehicular Networks	✗
UAS in Edge	Jeong et al. [17]	Single UAV	Moving cloudlet with the goal of energy optimization	Cellular networks where mobile users demand computation	✗
	Narang et al. [18]	Single or multiple UAVs without interconnection among each other	N/A Comms. challenges are addressed	Challenged Networks (CN)	✗
	Kattepur et al [19]	Single hop links to robots/drones	Timing-estimation based deployment of OpenCV applications	Multi-robot environment	✓
	Messous et al [20]	Challenges of aerial networks are not addressed	Game-theory based offloading scheme	Multi-UAV system	✗
FOCUS		Joint formulation and optimization of network and computation resources with two-layer Open Jackson Networks.		Multi-UAV system / Aerial Mesh Networks	✓

TABLE I: Comparison with the literature

Similar to the notion of delegating computational workload to UASs, following studies proposed to utilize vehicles as computational nodes or relays in vehicular ad-hoc networks to improve overall performance of cloud systems [14]–[16]. [14] proposed semi-markov decision process based resource allocation scheme for vehicular clouds, where vehicles in the network increases cloud resource pool via sharing their own computational resources. In [15], the authors developed content dissemination framework by integrating edge computing with vehicular networks where vehicles act as relays to deliver contents efficiently. In addition, an approximation scheme for job completion time in vehicular cloud is proposed in [16]. However, these studies neither considered UAS-specific challenges such as limited on-board resources, 3D mobility and unreliable ad-hoc links while devising their frameworks, nor provided any real-world implementation.

We describe next the works that come closest towards joint consideration of UAS capabilities, including computation. [19] estimates the execution time of different tasks when executed locally on ground robot units and remotely in the fog/cloud servers, and defines offloading strategies to optimize the service time. [20] approaches the same problem through game theory. General architectures employing UASs as fog nodes are proposed in [18] and [17]. In the former case, the UASs serve as mobile base stations providing connectivity to the ground units. In the second case, the UASs offer computation offloading opportunities to mobile ground units. However, the goal of this work is on enhancing the uplink/downlink communications and the path planning of the UASs. Table I showcases the differences of FOCUS with the existing studies in the literature where these studies are grouped under three sub-categories based on their scope and are investigated by four different aspects such as network, computation, target domain and test-bed implementation. Moreover, the work that we present in this paper does not explore the limited energy problem of the UASs that is indeed quite important when analysing aerial vehicles. However, several work can be found in literature that proposed solutions for this problem [21] [22] that can be used to obtain continuous operability. FOCUS pushes the envelope further by a joint analytical and systems approach, while opening up for the community the software tools and code to build and deploy aerial SDNs.

III. Preliminary Study on Computation on UASs

We conduct two different sets of experiments motivating the use of a UAN as an aerial computing platform: the first one stressing the communication/computing tradeoff on a simple linear topology, the second one showcasing the impact of processing on the UAS battery/flight time. To better understand the advantages of data processing on fog nodes within an UAN as opposed to centralized cloud, we set up a mesh UAN in a linear topology consisting of 3 Intel Aero Ready-To-Fly UASs, as seen in Fig. 2a. Two ground laptops are placed at the two ends of the mesh, one acting as a ground control station (GCS) and application server, and the other as the source. The GCS containing 16GB DDR3 RAM, 7th generation Intel processor and NVIDIA® GTX 1060 graphic card doubles as the centralized cloud entity along with being the controller. The source generates network data as UDP packets and static images for processing. The target location for the processing may vary over time via *computational requests*. Since, we use same workload generation and image processing algorithms during evaluation, further details are explained in Section VI.

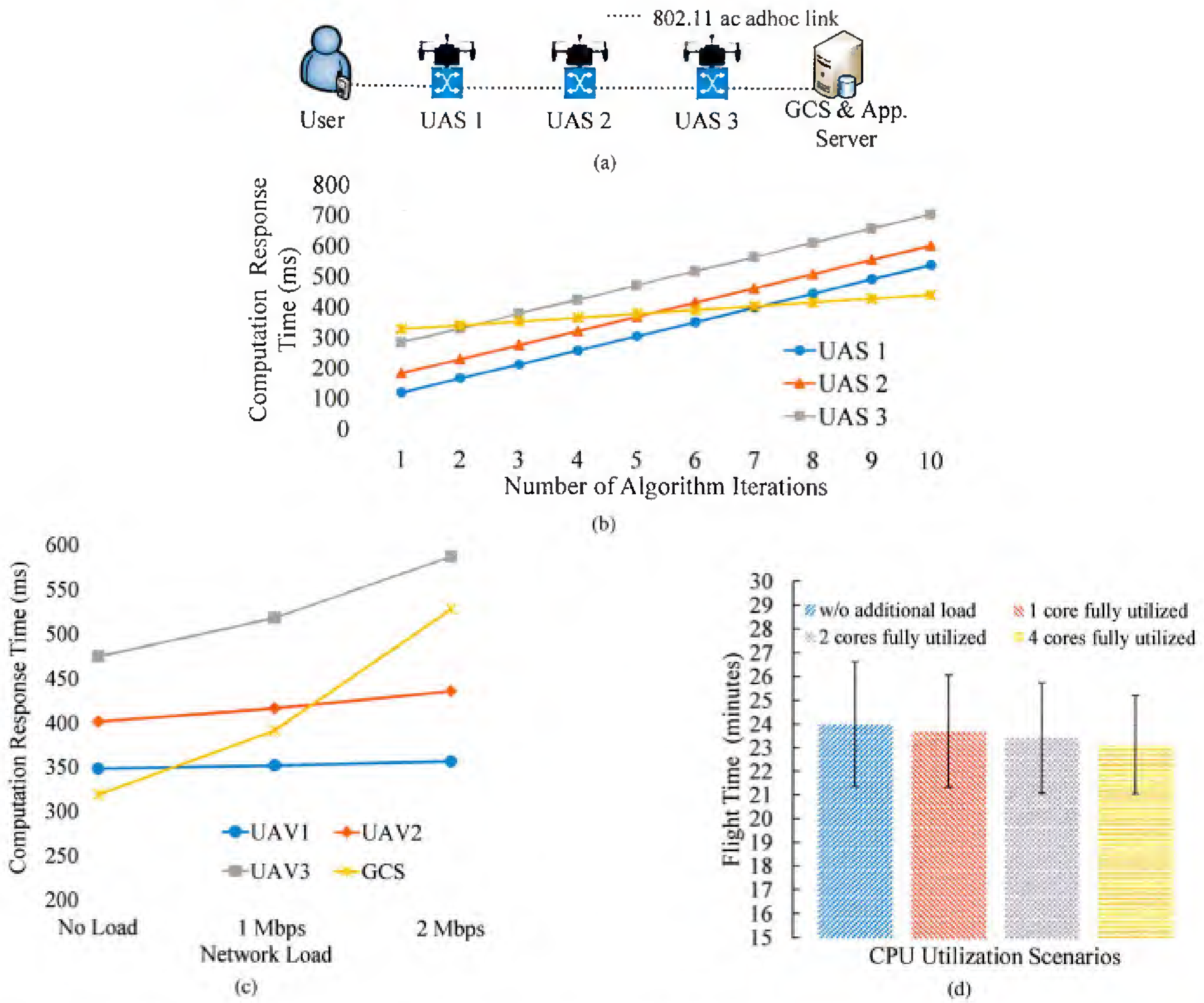


Fig. 2: (a) Linear UAN topology; computation response time w.r.t. (b) algorithm iteration number, (c) network load and; (d) UAS flight time w.r.t. processing complexity

Consider four scenarios where: computational requests are handled by UAS in inter-mediate hops, i.e., UAS1, UAS2, UAS3 and when the images reach the GCS/application server. The network load is gradually increased from 0 to 1Mbps and finally to the upper limit of 2Mbps. Under non-loaded network conditions, understandably, it is beneficial to execute the data processing

Symbol	Description
N	Number of UASs.
M	Number of ground units.
$\mathbf{E}$	Adjacency matrix for UAS network.
$\mathbf{A}$	Air-to-Ground connection matrix.
$\mu_i^{(t)}$	Service capacity for the traffic type t[1]at UASi.
$\gamma_i^{(t)}$	Bandwidth request of ground unit i belonging the traffic type t.
$\Omega_i^{(t)}$	Sum of arrival rates from ground units assigned to UASi.
$\lambda_i^{(t)}$	Arrival rate for the traffic type t at UASi.
$\mathcal{L}_i^{(t)}$	Expected number of packets in the queue at UASi.
$\mathcal{W}_i^{(t)}$	Average waiting time of type t packet in the queue at UASi.
$\gamma_j^{(t)}$	Bandwidth request of ground unit j.
$\mathbf{R}^{(t)}$	Routing matrix.
$r_{ij}^{(t)}$	Element of $\mathbf{R}^{(t)}$ defining the ratio of the packets routed from UASi to UASj.
$\mathbf{F}, \mathbf{C}$	Auxiliary matrices used in the heuristics to hold pointers to father (**F**) and child (**C**) nodes on network flows.
$\mathcal{B}^R, \mathcal{B}^P$	Bipartite graphs holding computational requests (R) and available slots (P)

TABLE II: List of important notations

[1] $t \in \{n, c\}$, where n states non-computation/network and c states computation traffic. If a symbol does not contain a superscript, it means there is not any traffic type distinction for that parameter.

at the remote GCS, as seen in Fig. 2c. However, when the network load begins to increase, the amount of time spent on forwarding information in the network indicates the benefit of processing the image on a fog entity, i.e., the UAS as opposed to the GCS, despite having weaker computational power. As seen for the cases of 1 and 2 Mbps network traffic, processing images on the UAS improves the response time.

As algorithms may vary in processing requirements (for e.g., genetic algorithms and particle swarm optimization run multiple iterations), we define an iteration variable that increases the effort involved in completing the image processing task. Each iteration re-analyzes the image, with the goal being to identify the break-even points for switching the processing from fog nodes to the GCS. As seen in Fig. 2b for a constant network load of 1Mbps, algorithms with fewer iterations to completion are more suited for the fog nodes, though every additional forwarding hop impacts the difference from the ground server to a greater extent. These studies indicate practical *handover* points that we use in our optimization approach.

Computation and data relaying both incur energy costs. To study their inter-dependence, we flew a UAS equipped with the standard 4S, 4500mAh battery, under different CPU utilization scenarios, as shown in Fig. 2d, recording the flight time for each setting. These scenarios are defined by the number of CPU cores fully utilized during the entire flight starting with 0 and continue with 1, 2 and 4 cores. During the flights, we use a simple workload generator [23] to fully utilize individual CPU cores on UASs. Increasing CPU utilization drains negligible battery power when compared to the power drawn by the UAS propeller motors. This further motivates us to fully leverage fog nodes for processing within the network. Fig. 2d shows high variations in the upper and lower bounds in the flight time due to uncontrollable flight dynamics (altitude deviation, pitch, yaw, roll) caused by wind, which again impacts the UAS battery consumption rate more than internal data processing. We run the experiment ten times for each scenario with a fully charged battery, and record the flight time until its remaining battery capacity hits a critical level (which is 15%).

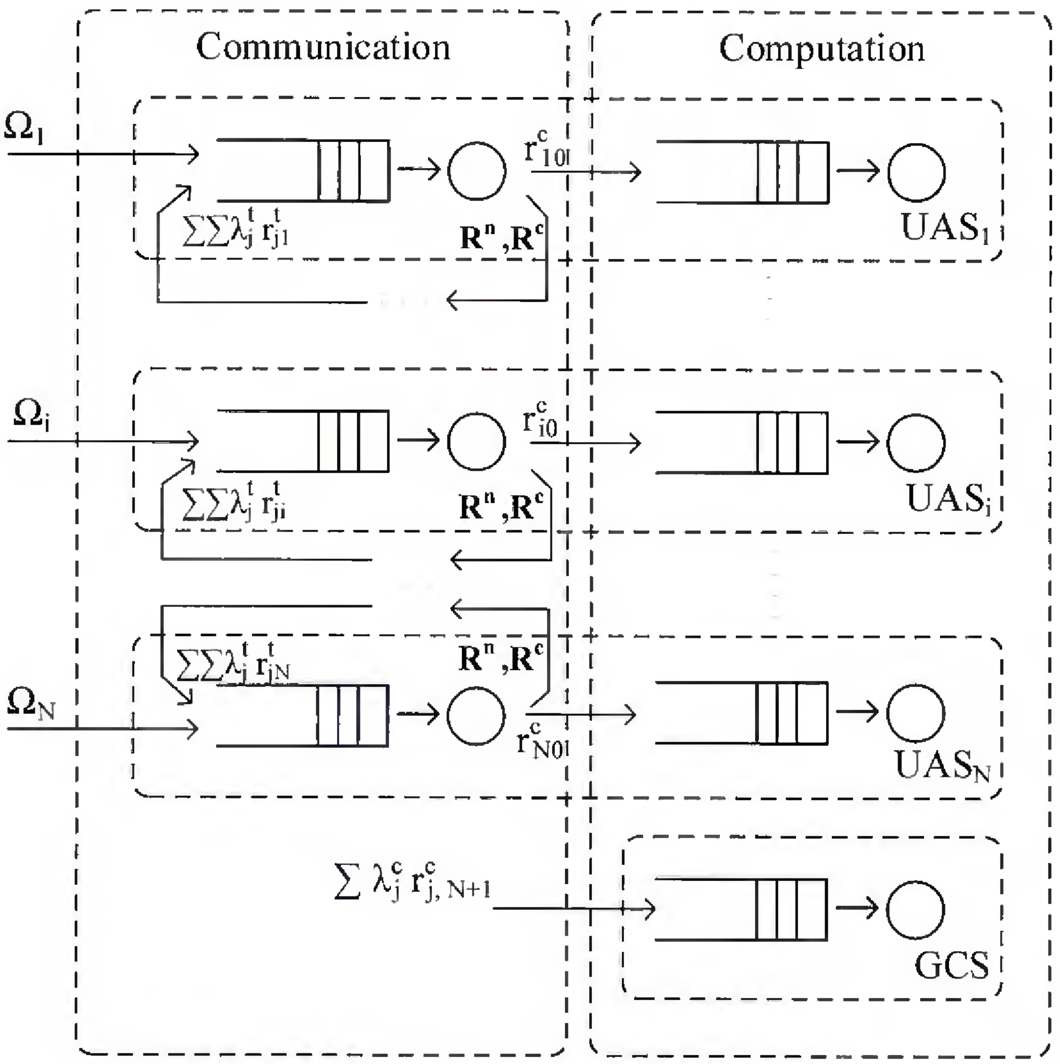
IV. Problem Formulation and Optimization Framework

In the following, we formulate the problem of joint assignment of network flows and computational tasks by modeling UAN as a network of queues. Then we describe two-phase solution to this problem, where network flow and computational flow optimizations are solved consecutively. Table II defines the variables and symbols used in the process.

A. System Model

We assume a scenario with N flying UASs, M ground units and one ground control server (*GCS*). The ground units produce two classes of network traffic: (*i*) generic network traffic (denoted by the n apex), i.e. sensor data that must be delivered to the GCS, and (*ii*) computational traffic (denoted by the c apex), i.e. data that must be processed on the cloud or on fog nodes. We introduce the following variables:

- $U = \{u_1, u_2, \ldots, u_N\}$ is the set of available UASs;
- $G = \{g_1, g_2, \ldots, g_M\}$ is the set of the ground units;
- $\Upsilon^{(c)} = \{\mu_1^{(c)}, \mu_2^{(c)}, \ldots, \mu_N^{(c)}, \mu_{N+1}^{(c)}\}$ is the computational capacity (in Kbps) for each UAS $u_i \in U$ and for the *GCS* (defined by $\mu_{N+1}^{(c)}$);



Communication
Computation
Ω_1
Ω_i
Ω_N
r^c_{10}
r^c_{i0}
r^c_{N0}
$\sum\sum\lambda^t_j r^t_{j1}$
$\sum\sum\lambda^t_j r^t_{ji}$
$\sum\sum\lambda^t_j r^t_{jN}$
$\mathbf{R}^n,\mathbf{R}^c$
$\sum \lambda^c_j r^c_{j,N+1}$
UAS$_1$
UAS$_i$
UAS$_N$
GCS

Fig. 3: Two-layer Open Jackson Network

- $\Gamma = \gamma_1, \gamma_2, \dots, \gamma_M$ is the set of bandwidth requests (in Kbps) for each ground unit $g_j \in G$, where $\gamma_i = \gamma_i^{(n)} + \gamma_i^{(c)}$, i.e. it includes both generic and computational data;
- $\mathbf{E}_{N\times(N+1)}$ is UAN adjacency matrix, where $e_{i,j} \rightarrow \{0,1\}$ indicates whether there is an active link between the UASs $u_i, u_j \in U$, where also index $N+1$ represents the GCS;
- $\mathbf{A}_{N\times M}$ is Air-to-Ground (A2G) connection matrix, where $a_{i,k} \rightarrow \{0,1\}$ indicates whether there is an active connection between UAS $u_i \in U$ and ground unit $g_k \in G$.

$\mathbf{E}_{N\times(N+1)}$ and $\mathbf{A}_{N\times M}$ matrices are assumed to be pre-computed based on the position of the ground units. The UAN is modeled as an *Open Jackson Network* [24], and each UAS node is represented with two consecutive M/M/1 queues as shown in Fig. 3. On the left, we depict the network queues used for both traffic types (n and c). On the right, we depict the process queue corresponding to the computational traffic (c), for both the UASs and the GCS. The output of the queues are determined by the routing policies. These latter are formally modeled via the matrices $\mathbf{R}^{(n)}_{N\times(N+2)}$ and $\mathbf{R}^{(c)}_{N\times(N+2)}$, respectively for the n, c traffic classes; $r^{(t)}_{i,j}$ indicates the ratio of traffic of u_i routed to u_j for traffic type $t \in \{n, c\}$. In addition, $r^{(n)}_{i,0}$ and $r^{(c)}_{i,0}$ denotes ratio of the packets left the network and, ratio of packets processed locally at u_i. Similarly, let $r^{(c)}_{i,(N+1)}$ be the ratio of the computational packets that u_i forwards to GCS, to be processed on the cloud. The overall packet arrival rate (λ_i) to u_i is defined as $\lambda_i = \lambda_i^{(n)} + \lambda_i^{(c)}$. Assuming that, the ground units generate requests with the average $1/\gamma$ time difference, we use Markovian queues to obtain closed-form equations for the upper-bounds of the response times in the model.

For each traffic type ($\forall t \in \{c, n\}$), the $\lambda_i^{(t)}$ term can be modeled as $\lambda_i^{(t)} = \Omega_i^{(t)} + \sum_{j=1}^{N} r^{(t)}_{j,i} \cdot \lambda_j^{(t)}$, where $\Omega_i^{(t)} = \sum_{k=1}^{M} a_{i,k} \cdot \gamma_k^{(t)}$ and the variable $\gamma_k^{(t)}$ is the arrival rate of the c and n traffic from the k^{th} ground unit and $\Omega_i^{(t)}$ is the sum of the arrival rates over all ground units assigned to UAS i.

By Little's Law, the average response time for packets belonging to traffic type c ($\mathcal{W}^{(c)}$) is defined as follows:

$$\mathcal{W}^{(c)} = \frac{\sum_{i=1}^{N} \mathcal{L}_i^{(n)} + \sum_{i=1}^{N+1} \mathcal{L}_i^{(c)}}{\sum_{k=1}^{M} \gamma_k^{(c)}} \tag{1}$$

where $\mathcal{L}_i^{(n)}$ and $\mathcal{L}_i^{(c)}$ indicate the expected number of computational packets in the network queue and in the computational queue of UAS i respectively. Let $\Upsilon^{(n)} = \{\mu_1^{(n)}, \mu_2^{(n)}, \dots, \mu_N^{(n)}\}$ be the average service rate for each UAS $u_i \in U$, with $\mu_i^{(n)}$ expressed again in Kb/s. Hence, the average number of computational (c-type) packets at the first queue is $\mathcal{L}_i^{(n)} = \lambda_i^{(c)}/\left(\mu_i^{(n)} - \lambda_i\right)$. On the other hand, the number of the packets in the second queue is defined as follows $\mathcal{L}_i^{(c)} = \lambda_i^{(c)} r_{i,0}^{(c)}/\left(\mu_i^{(c)} - \lambda_i^{(c)} r_{i,0}^{(c)}\right)$. We can hence rewrite $\mathcal{W}^{(c)}$ as follows:

$$\mathcal{W}^{(c)} = \frac{\sum_{i=1}^{N} \frac{\lambda_i^{(c)}}{\mu_i^{(n)} - \lambda_i} + \sum_{i=1}^{N+1} \frac{\lambda_i^{(c)} \cdot r_{i,0}^{(c)}}{\mu_i^{(c)} - \lambda_i^{(c)} \cdot r_{i,0}^{(c)}}}{\sum_{k=1}^{M} \gamma_k^{(c)}} \tag{2}$$

Similarly, we compute the average response time for the generic network traffic as follows:

$$\mathcal{W}^{(n)} = \frac{\sum_{i=1}^{N} \frac{\lambda_i^{(n)}}{\mu_i^{(n)} - \lambda_i}}{\sum_{k=1}^{M} \gamma_k^{(n)}} \tag{3}$$

The proposed system assumes to have the knowledge of the all data load requests Γ and the network links quality $\Upsilon^{(n)}$. Consequently, also the $\mathbf{A}_{N\times M}$ and the $\mathbf{E}_{N\times(N+1)}$ matrices are assumed to be known. These indexes represent the ground user bandwidth requests and the network links quality, respectively. These two indexes can be known in advances for static scenarios and with full knowledge, but this case is very rare. On the other side, for evolving and/or unknown scenarios, these indexes can be dynamically estimated by the system using link quality estimation within the SDN networks [25] in order to estimate $\Upsilon^{(n)}$ and $\mathbf{E}_{N\times(N+1)}$ and continuous monitoring of the ground user requests to estimate Γ and $\mathbf{E}_{N\times(N+1)}$. The effects of the convergence of the index estimations and their change over time will be analysed in future works.

B. Problem Formulation

Based on this system model, we formally define the joint routing and computation assignment (RC) problem as follows:

$$\min_{\mathbf{R}^{(t)}} \mathcal{W}^{(c)} \tag{4}$$

$$\text{subject to:} \sum_{u_i \in U} a_{i,j} = 1 \qquad \forall g_j \in G \tag{5}$$

$$dim\ kernel(Laplacian(\mathbf{E})) = 1 \tag{6}$$

$$\lambda_i < \mu_i^{(n)} \qquad \forall u_i \in U \tag{7}$$

$$\lambda_i^{(c)} \cdot r_{i,0}^{(c)} < \mu_i^{(c)} \qquad \forall u_i \in U \tag{8}$$

$$\sum_{j=0}^{N+1} r_{i,j}^{(t)} = 1 \qquad \forall u_i \in U, t \in \{n, c\} \tag{9}$$

$$r_{i,0}^{(t)} + \sum_{j=1}^{N+1} (r_{i,j}^{(t)} \cdot e_{i,j}) = 1 \qquad \forall u_i \in U, t \in \{n, c\} \tag{10}$$

$$\mathcal{W}^{(n)} \le \rho^{(n)} \tag{11}$$

$$r_{i,j}^{(t)}, \lambda_i^{(t)}, \mu_i^{(t)}, \gamma_i^{(t)} \ge 0 \qquad \forall u_i \in U, t \in \{n, c\}, \; j \in [0, \mathrm{N}+1] \tag{12}$$

The goal of the optimization problem, defined by (4), is to minimize the average response time of computational traffic, by determining the optimal routing matrices $\mathbf{R}^{(t)}$, meeting the following constraints: (5) states that each ground unit must be connected to a single UAS; (6) ensures the aerial mesh is connected by analysing dimension of the kernel of the adjacency matrix's *Laplacian*; (7) and (8) guarantee the validity of Open Jackson Network model; (9) and (10) state that all the packets leaving a network queue will flow to another network queue or to a computation queue, but only using active links; finally, (11) ensures the service time for type n traffic does not exceed a user-defined threshold $\rho^{(n)}$.

RC multi-objective optimization problem is NP-hard since it is a generalization of the well-known splittable flow problem [26]. Therefore, we divide the original problem into two phases: first, we compute the routes for traffic type n toward the GCS. Then, based on such allocation, and the estimated network traffic congestion, we compute the routes for the computational data flows (type c). Fig. 4 shows the modules implementing each phase. The first module, namely *Network Flow Optimization (NFO)*, generates the entries of the $\mathbf{R}^{(n)}$ matrix as output. These values are taken as input by *Computation Flow Optimization (CFO)* module producing as output the final $\mathbf{R}^{(c)}$ matrix.

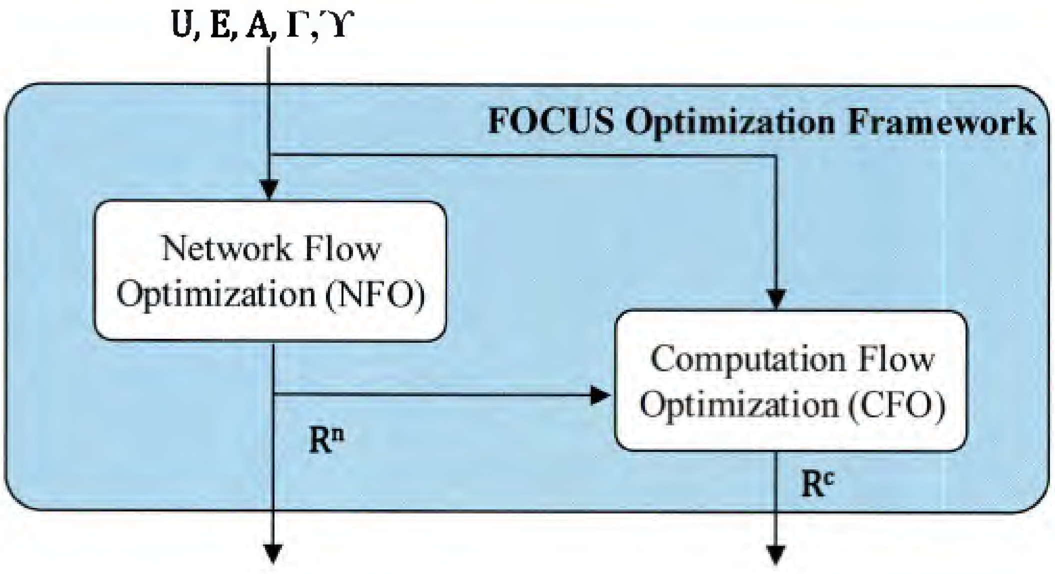

U, E, A, Γ, Υ
FOCUS Optimization Framework
Network Flow Optimization (NFO)
Computation Flow Optimization (CFO)
Rⁿ
Rᶜ

Fig. 4: FOCUS optimization framework

C. Network Flow Optimization (NFO)

NFO algorithm determines the proper values of $r_{i,j}^{(n)}$, with $i, j \leq N$, i.e. the routing entries for the class n traffic directed to the GCS. Given the complexity of the original problem, we relax the constraint of (11), i.e. we determine the routing entries minimizing the average response time $\mathcal{W}^{(n)}$. Formally:

Definition 1. Given the set of available UASs (U), the connection matrix ($\mathbf{E}_{N\times N}$), the air-to-ground active links matrix ($\mathbf{A}_{N\times M}$) and the network requests set (Γ), the goal is to compute the network routing matrix $\mathbf{R}_{N\times(N+2)}^{(n)}$ such that the average response time for class n traffic $\mathcal{W}^{(n)}$ is minimized.

Algorithm 1: NFO algorithm

```
 1: NFO ($U, \mathbf{E}_{N\times N}, \mathbf{A}_{N\times M}, \Gamma, \Upsilon$)
 2: init $\mathbf{F}_{N+1\times N+2}$ with $f_{i,j} = -1$ {$f_{i,j} \geq 0$ is the cost of $j$ being father of $i$}
 3: init $\mathbf{C}_{N+1\times N+2}$ with $c_{i,j} = 0$ {$c_{i,j} = 1$ if $j$ is a child for $i$}
 4: init $v_i = false$, $\lambda_i^{(n)} = \Omega_i^{(n)}$, $r_{i,j}^{(n)} = 0$, $1 \leq i, j \leq N$
 5: set $f_{N+1,N+2} = 0$ {$N+1$ is the GCS and $N+2$ is a dummy node}
 6: while $\exists u_k$ s.t. $(v_k = false) \wedge (\exists j$ s.t. $f_{k,j} \geq 0) \wedge (\arg\min_k \xi^{(n)}(u_k))$ do
 7:     $v_k = true$
 8:     call updateLambdas($u_{N+1}$)
 9:     call updateFathersCost($N+2, u_{N+1}, 0$)
10:     call updateChildren($u_k$)
11: return $R^{(n)}$

12: function updateLambdas ($u_k$)
13: $\lambda_k \leftarrow \Omega_k$
14: for all $u_i$ s.t. $c_{k,i} = 1$ do
15:     if $v_i = true$ then
16:         $\lambda_k \leftarrow \lambda_k + (r_{i,k}^{(n)} \cdot$ updateLambdas($u_i$) )
17: return $\lambda_k$

18: function updateFathersCost ($j, u_k, fullcost$)
19: $f_{k,j} \leftarrow fullcost$
20: for all $u_i$ s.t. $c_{k,i} = 1$ do
21:     updateFathersCost($k, u_i, \xi^{(n)}(u_k)$)

22: function updateChildren ($u_k$)
23: for all $u_i$ such that $e_{k,i} = 1$ with $e_{k,i} \in \mathbf{E}$ do
24:     if $v_i = false$ then
25:         $c_{k,i} = 1$
26: for all $u_i$ s.t. $c_{k,i} = 1$ do
27:     $f_{k,i} \leftarrow \xi^{(n)}(u_k)$
28:     $r_{i,j}^{(n)} \leftarrow \frac{\prod_{(m: f_{i,m} \geq 0, m \neq j)} \xi^{(n)}(u_m)}{\sum_{(m: f_{i,m} \geq 0)} \prod_{(w: f_{i,w} \geq 0, w \neq m)} \xi^{(n)}(u_w)}, \forall j: f_{i,j} \geq 0$
```

We address the problem by using a modified version of the Dijkstra algorithm for the Shortest Path Problem (SPP) over acyclic weighted graphs. To this purpose, we define the cost function for node u_i ($\xi^{(n)}(u_i)$) as a proxy of the average delay toward the GCS, computed as follows:

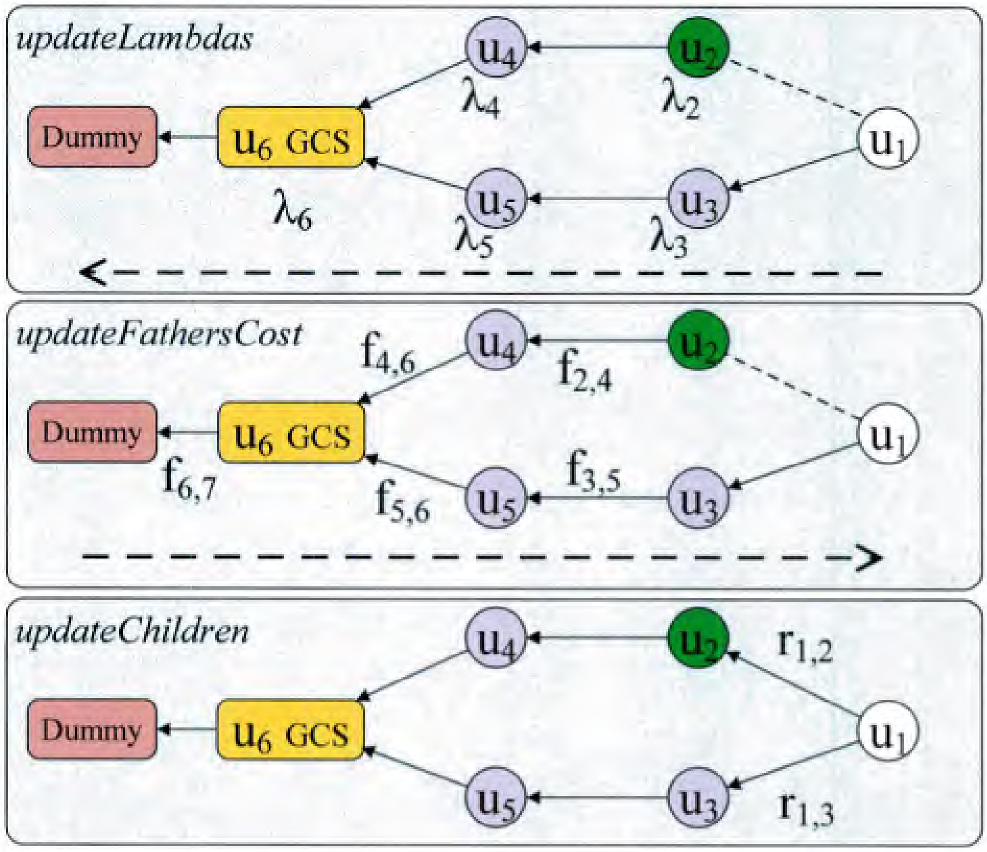


Fig. 5: A schematic view for a single step execution of Alg. 1 (lines 8 - 10). Here N=5 and active node in the step is u_2.

$$\xi^{(n)}(u_i) = \frac{1}{\mu_i^{(n)} - \lambda_i^{(n)}} + \sum_{1 \leq j \leq (N+2)} f_{i,j} \cdot r_{i,j}^{(n)} \tag{13}$$

Here, the first fraction represents the average queuing delay at node u_i, while the second one is the average delay of the path toward the GCS. We recall that the Dijkstra algorithm computes the shortest path between a source (or multiple sources) to a single destination. Unfortunately, the original algorithm does not fit our problem because: (*i*) the weight of the nodes/arcs is not static, since it may change over time as a specific node is included in the routing path of a flow toward the GCS; (*ii*) each node can have multiple paths towards the GCS, and exploit all of them concurrently. The proposed solution is described by Alg.1. As in Dijkstra's algorithm, we start building the shortest paths from the destination (the GCS in our case). In addition, we keep two auxiliary matrices as:

1) **F** matrix keeps track of the network flows costs. It includes the forward pointers from one node to its fathers, i.e. $f_{i,j} \in \mathbf{F}$ is greater then zero if u_j is a father for u_i towards the GCS and defines the path cost from u_j;
2) **C** matrix keeps track of the incoming network flows. It includes the reverse pointers from one node to its subtree, i.e. $c_{i,j} = 1$ if node u_j is a child of node u_i, 0 otherwise.

These two matrices are initialized at the beginning of Alg. 1: line 3 for the **C** matrix and line 2 for the **F** matrix; here, as starting point we defined a father connection from the GCS to a dummy node (having index $N+2$) whose cost is set to zero (line 5).At each iteration, the algorithm performs a greedy selection over the nodes, by adding the one having the minimum cost towards the GCS to the solution set (line 6). Then, it updates the $\lambda_i^{(n)}$ and the matrices **F** (lines 8 and 9) and **C** (lines 26-28). Finally, the algorithm updates $\mathbf{R}^{(n)}$ by balancing the outgoing traffic towards all the paths to minimize the total cost (line 28) equalizing $(r_{i,j}^{(n)} \cdot \xi^{(n)}(u_j))$ values, $\forall j$: $f_{i,j} \geq 0$.For space reason, here we omit the check if $r_{i,j}^{(n)} > 1$ in cases, which cost cannot be balanced over all the fathers.

In Fig. 5, we depicted one step execution of Alg. 1. We visualize the three main functions `updateLambdas`, `updateFathersCost` and `updateChildren` in a case where the active node is u_2 and u_1 is not yet visited. During `updateLambdas` method, the flows goes from the leaves to the root (GCS) and all the λ_i are updated accordingly with the tree connections. Then, `updateFathersCost` function updates the links' cost of each connection starting from *Dummy-GCS* connection (having $f_{N+1,N+2}=0$) and going down towards the leaves. Finally, the `updateChildren` function updates the routing values $r^{(n)}$ for all the children belonging to node u_2. Since each node can have multiple paths towards the destination, the output of the proposed algorithm is a destination oriented directed acyclic graph rooted at the GCS, formally represented by $\mathbf{R}^{(n)}$ matrix.

D. Computation Flow Optimization (CFO)

CFO algorithm allocates tasks to computational resources, represented by the cloud or by UAS fog nodes. Based on the model in Section IV-A, this translates into determining the destination and path for class c packets. We model the problem as a weighted bipartite graph where *(i)* $\mathcal{B}^R = \{b_1^R, b_2^R, \dots\}$ is the set of computational requests (per unit of time) to be executed, with $|\mathcal{B}^R| = \sum_{\gamma_i \in \Gamma} \gamma_i^{(c)}$, and *(ii)* $\mathcal{B}^P = \{b_1^P, b_2^P, \dots\}$ is the set of computational slots (again, per unit of time) available on the fog either on the cloud ($|\mathcal{B}^P| = \sum_{\mu_i^{(c)} \in \Upsilon^{(c)}} \mu_i^{(c)}$). Let $\zeta : \mathcal{B}^R \times \mathcal{B}^P \to \mathbb{R}$ be the weight function, representing the benefit of assigning a request in $\mathcal{B}^R$ to a computational slot in $\mathcal{B}^P$. We consider an asymmetric assignment problem where the computational resources are strictly greater than the requests (satisfying the requirement of Equation 8), i.e. $|\mathcal{B}^R| < |\mathcal{B}^P|$. Let $req(b_k^R) = u_i$ be the mapping function that returns the UAS generating the request b_k^R; similarly, let $pow(b_l^P) = u_j$ be the function which returns to UAS providing computational slot b_l^P. The aim of the assignment problem is to determine an optimal assignment set $S = \{\langle b_i^R, b_j^P \rangle : b_i^R \in \mathcal{B}^R, b_j^P \in \mathcal{B}^P\}$, such that the total benefit $\sum_{\langle b_i^R, b_j^P \rangle \in S} \zeta(b_i^R, b_j^P)$ is maximized, clearly subject to the constraints that each request must be assigned to a single slot and each slot can host at most one computational request.

Algorithm 2: CFO algorithm

```
1: CFO $U, p, \mathbf{E}_{N \times N}, \mathbf{A}_{N \times M}, \Gamma, \Upsilon^{(n)}, \Upsilon^{(c)}, \mathcal{B}^R, \mathcal{B}^P$
2: for all $u_i \in U$ do
3:     if $\Omega_i^{(n)} > 0$ then
4:         calculate Dijkstra($u_i, GCS$)
5:         update $r_{j,k}^{(n)}$, $\forall u_k \in U$ and $\forall u_j$ in the calculated path
6:         update $\lambda_j^{(n)}$, $\forall u_j$ in the calculated path
7: $PL(b_i^R) \leftarrow \{\}, \forall b_i^R \in \mathcal{B}^R$
8: while $S$ doesn't contains all the assignment for $\forall b_i^R \in \mathcal{B}^R$ do
9:     calculate Dijkstra($u_i, u_j$), $\forall u_i, u_j \in U$
10:    update $\zeta(b_i^R, b_j^P), \forall b_i^R \in \mathcal{B}^R, b_j^P \in \mathcal{B}^P$ based on Dijkstra($req(b_i^R), pow(b_j^P)$)
11:    execute one forward/reverse step of the auction algorithm
12:    if $\langle b_i^R, b_j^P \rangle$ is a new assignment then
13:        $PL(b_i^R) \leftarrow$ Dijkstra($req(b_k^R), pow(b_l^P)$)
14:    else if $\langle b_i^R, b_j^P \rangle$ is removed as an assignment then
15:        $PL(b_i^R) \leftarrow \{\}$
16:    update all $\lambda_i^{(c)}$ and $r_{i,j}^{(c)}$ based on the path lists $PL(b_i^R)$, $\forall b_i^R \in \mathcal{B}^R$
```

The proposed solution (given in Alg. 2) enhances the basic forward/reverse auction scheme described in [27] for the case of weights dynamically changing over time. Indeed, each assignment causes the modification of ζ function due to the alteration of $\lambda_i^{(c)}$ values for UASs residing on the chosen path (line 10). Hence, the algorithm implements a sequence of forward/reverse iterations, where at each iteration a request-slot assignment can be added/removed from the final result. Let $PL(b_i^R)$ denote the list (line 7, 13, 15) containing the calculated path for each b_i^R, the benefit function $\zeta(b_i^R, b_j^P)$ of assigning request b_i^R to slot b_j^P is modeled as follows:

$$\zeta(b_i^R, b_j^P) = \frac{1}{1 + cost(b_i^R, b_j^P)} \tag{14}$$

where $cost(b_i^R, b_j^P)$ is a proxy for the delay of traffic class c from UAS $req(b_i^R)$ to $pow(b_j^P)$, and can be calculated by considering the Dijkstra($req(b_i^R), pow(b_j^P)$) shortest path having the edge $(u_i \to u_j)$ weight defined as: $\frac{1}{\mu_i^{(n)} - \lambda_i}$. More precisely, let $PL(b_i^R) = \{req(b_i^R), \dots, pow(b_j^P)\}$ be the path used to reach $pow(b_j^P)$, then:

$$cost(b_i^R, b_j^P) = \left(\sum_{u_k \in PL(b_i^R)} \frac{1}{\mu_k^{(n)} - \lambda_k} \right) + \frac{1}{\mu_{pow(b_j^P)}^{(c)} - \left(\lambda_{pow(b_j^P)}^{(c)} \cdot r_{pow(b_j^P),0}^{(c)} \right)} \tag{15}$$

E. Computational Complexity

We now analyze Computational Complexity (CC) of FO-CUS where *NFO* algorithm is followed by *CFO* algorithm.
NFO algorithm is based on the Dijkstra algorithm whose complexity is $O(N^2)$ in its basic form. We see this complexity in the main *while* loop in line 6 of Algorithm 1 where the loop is executed N times and the *argmin* operator is $O(N)$. Inside the loop, the computation is dominated by the functions *updateLambdas* and *updateFathersCost* that visit the whole graph to update the lambdas and cost variables. In conclusion, the *CC* of the *NFO* algorithm is $O(N^3)$.

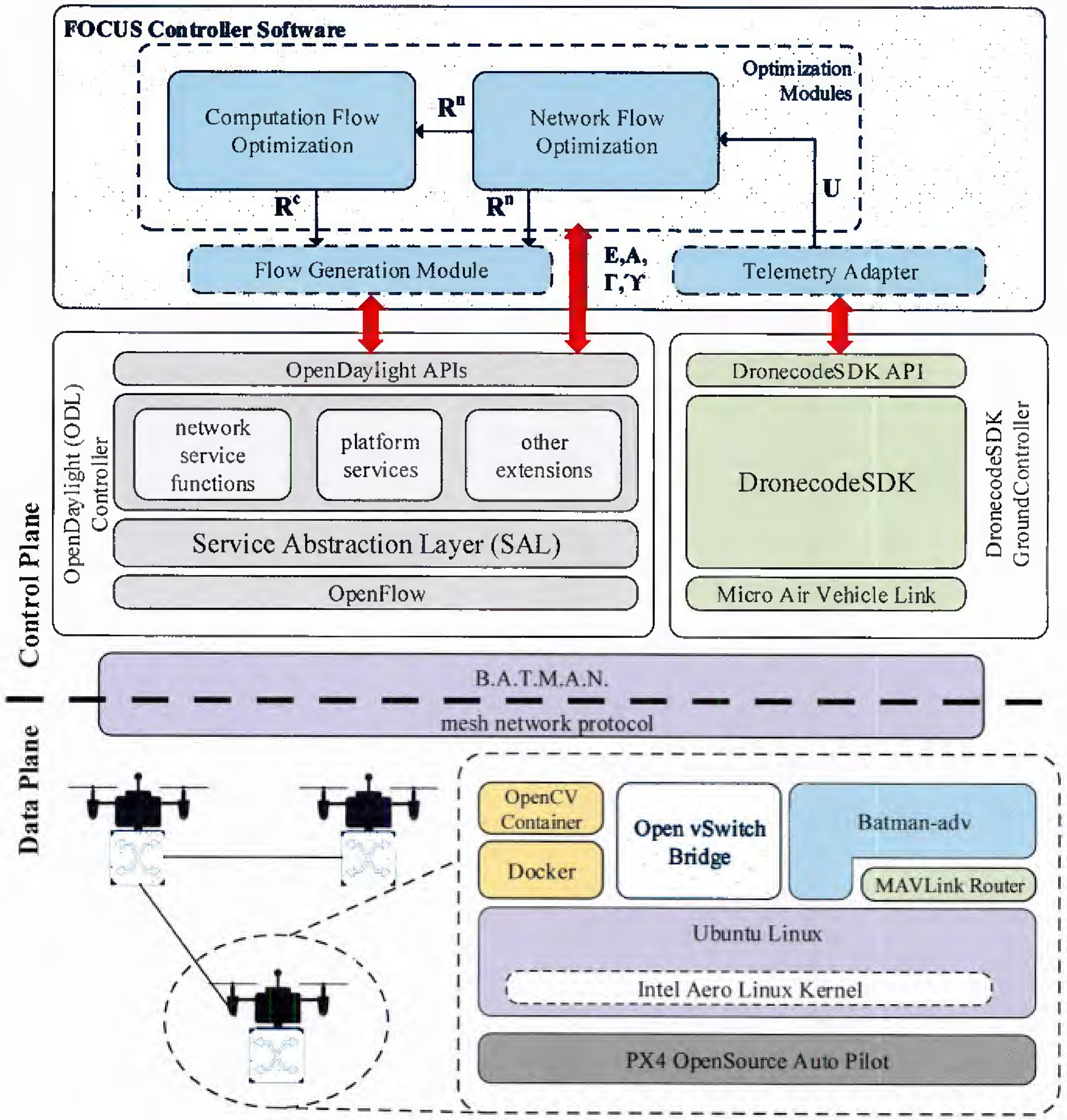


Fig. 6: FOCUS Software Architecture

CFO algorithm copes with the asymmetric bipartite graph problem between two asymmetric sets: the computational requests set having cardinality $|\mathcal{B}^R|$ and the computational slots set having cardinality $|\mathcal{B}^P|$. The auction algorithm solves a generic asymmetric bipartite graph problem in $O(|\mathcal{B}^R||\mathcal{B}^P| \cdot log(n))$, where n is a parametric value [27]. However, in our implementation, we add an extra execution time for updating the cost matrix defined by the function $\zeta(b_i^R, b_j^P)$ (lines 4 and 5 in Alg. 2). Dijkstra's algorithm has a complexity of $O(N^2)$ and the matrix update has complexity $O(|\mathcal{B}^R||\mathcal{B}^P|)$. This brings the total *CC* to $O(|\mathcal{B}^R||\mathcal{B}^P| \cdot log(n) \cdot (|\mathcal{B}^R||\mathcal{B}^P| + N^2))$.

V. Focus Systems-level Implementation

One of the main contributions within FOCUS is the development of the middleware platform transforming the classical UAN into a joint sensing, forwarding and fog computing architecture. This middleware interacts with existing software blocks,

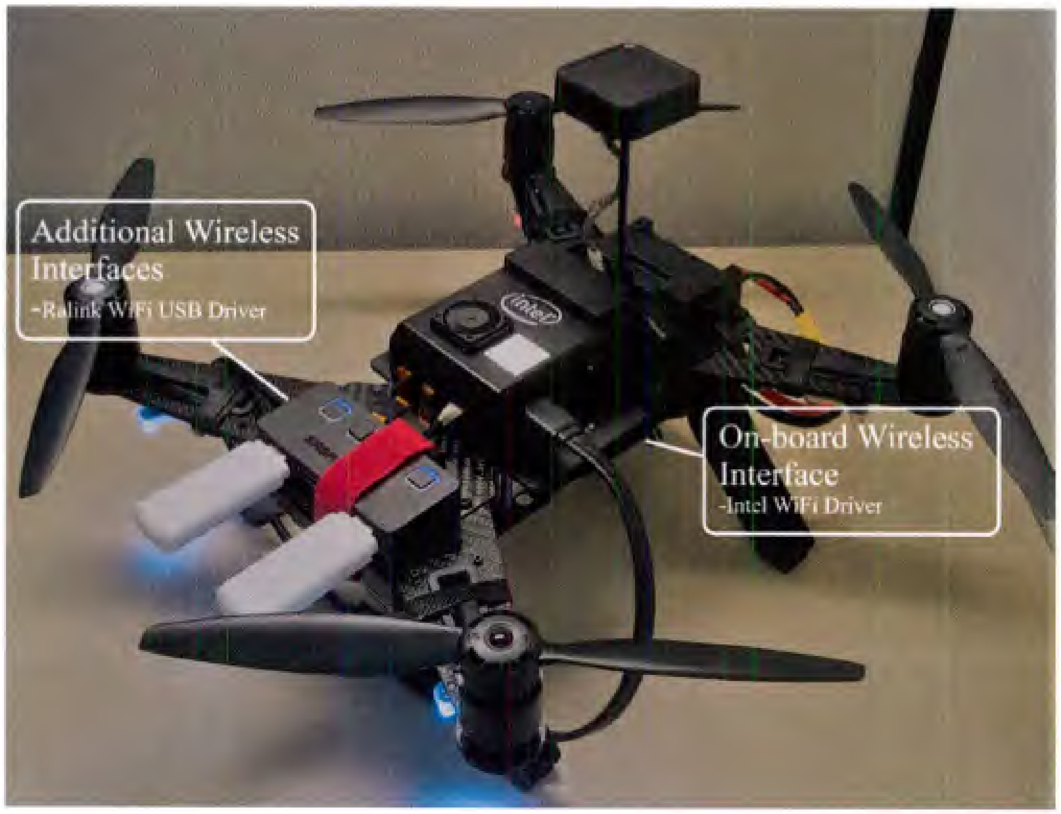


Fig. 7: Intel Aero Ready-to-Fly drone as an aerial OF switch

such as those related to coordinating with the ground controller and SDN controller simultaneously. It receives both network and UAS information, and implements the necessary control directives originated from heuristic algorithms that centrally solve RC problem. As shown in Fig. 6, FOCUS is built on top of the OpenDayLight (ODL) SDN controller and DronecodeSDK [28]. The former orchestrates flows in the UAN being controlled by a REST application programming interface (API) and the latter aggregates location information of the UASs and makes this information available to the controller via the telemetry adapter. Through these tightly coupled APIs, the network information required by the FOCUS is aggregated and forwarded to the sub-modules (NFO, CFO). These modules, residing in the offsite controller, in turn calculate the optimal allocation of network and computational flows in the UAN and define routing matrices ($\mathbf{R}^{(n)}$ and $\mathbf{R}^{(c)}$). They then initiate control feedback flows via HTTP requests through REST API back to the UAN.

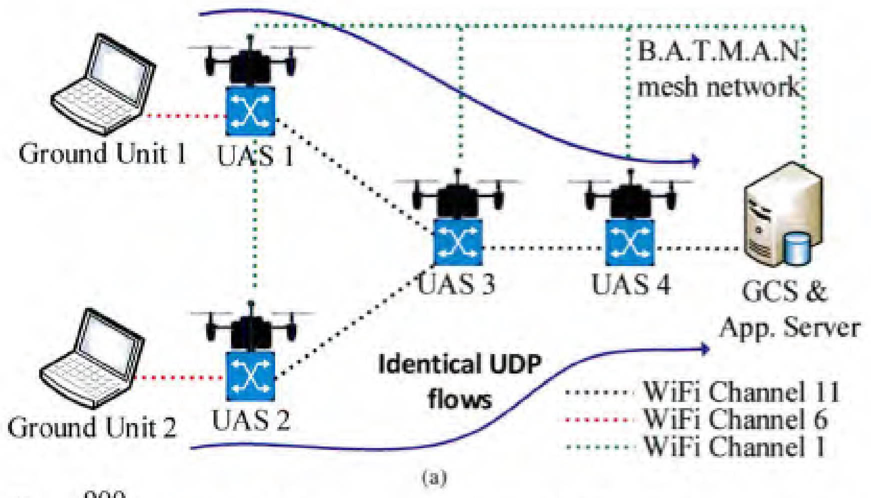
In addition to the control plane design, we also utilize Docker [29] and OpenVswitch [30] (OVS) on UASs at the data plane. Docker hosts a container with OpenCV library to run image processing as computational load (this can be swapped for other applications), while OVS connects to ODL as a traditional Openflow switch. SDNs are classically installed on reliable (often wired) network connections where the control/data planes are not easily impaired. To bring more robustness to the UAN, we utilize a distributed 2nd-layer routing, called 'Better Approach To Mobile Ad-hoc Networking' (BATMAN) [31]. It allows control directives and data to flow over through multiple different pathways to target UAS, even when direct link to the controller is impaired, albeit with an increased latency. Furthermore, it statistically determines the wireless link quality among the nodes and generates numerical values, which are aggregated at the software controller to estimate the throughput capacity on the links. This information is forwarded to FOCUS Optimization framework as seen in Fig. 6, for to be used in constructing of queue-based network model and in solving the optimization problem.

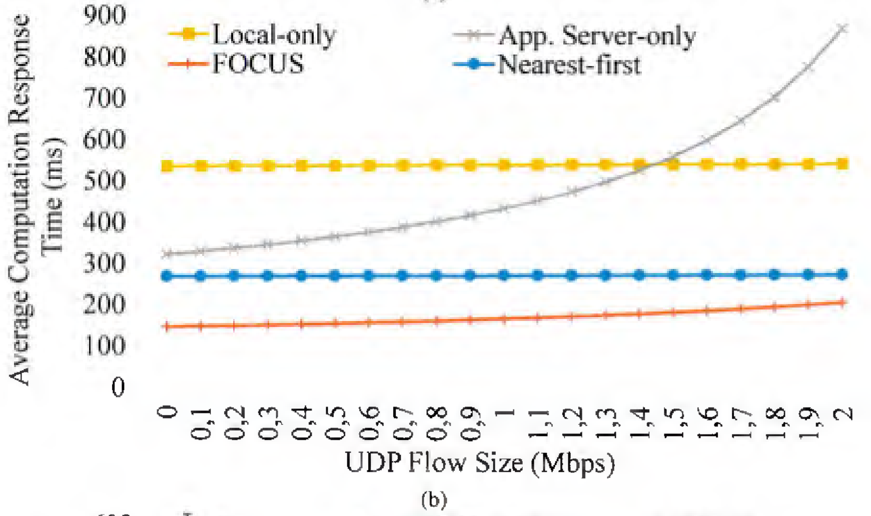
VI. Performance Evaluation

In this section, we validate the performance of FOCUS in terms of overall network traffic and computational response time, in two separate approaches. We conduct our experiments on a small scale UAN testbed, which consists of 4 UASs. The key insights from these experiments then are extended with large-scale simulations consisting of 40 UASs written in C.

A. Results on small-scale testbed implementation

We use two laptops as ground units and a high performance server as the control station, on where SDN controller (ODL), docker image with OpenCV libraries and dronecode flight controller run. 4 Intel Aero UASs create a mesh network with 4 hops





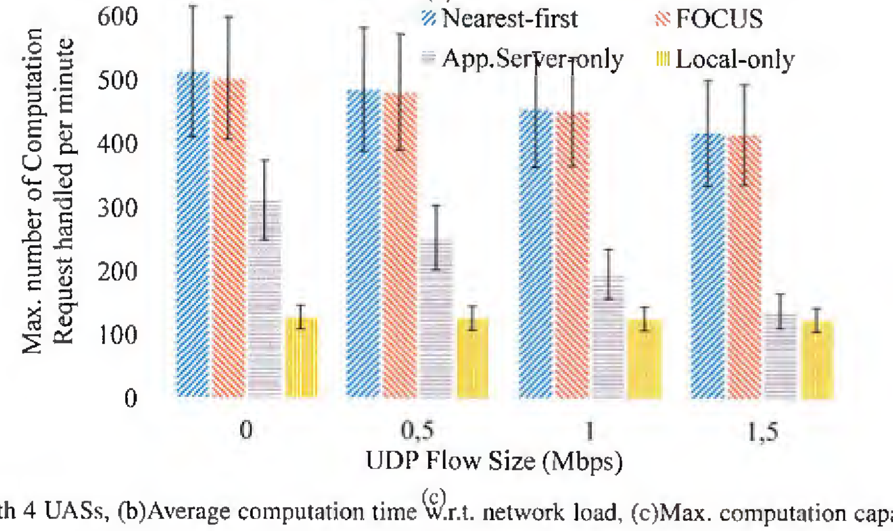


Fig. 8: (a)Testbed with 4 UASs, (b)Average computation time w.r.t. network load, (c)Max. computation capacity of the network

between the ground units and the server, as shown in Fig. 8a. The general hardware specifications are similar to Section III. During the experiment, UASs are positioned in hovering motion at 3-meter above the ground separated by 10-meter distances from each other and from the ground entities in the outdoor drone testing facility at Northeastern University. In our testbed, each UAS is equipped with three wireless interface cards (two RALINK WiFi dongles and one on-board Intel WiFi interface) as seen in Fig. 7, where each interface uses a non overlapping WiFi channel for different tasks. One of them is dedicated to the BATMAN protocol, another is used to create link between UASs and the last one is utilized for the link between ground units and UAS, on channels 1, 6, and 11, respectively. The UASs are positioned in such a way that Ground Unit 1 can only connect with UAS 1 and Ground Unit 2 can connect with UAS 2, creating a separate data path for each ground unit to reach the server. A 780x480 pixels picture of file size 1024 Kilobits is used as payload for image processing. These payloads are created and forwarded with 200 msec average inter-generation time at each ground unit. We used Binary Robust Independent Elementary Features (BRIEF) [32] algorithm as a feature point descriptor on these images. The network load is emulated by creating UDP flows from each ground unit to the server as shown in Fig. 7. We stress the network by gradually increasing the UDP data rate on both flows from 0 to 2Mbps. For comparison, we run other task allocation methods on the central controller:

- *Nearest first*, where the computation is allocated to the nearest neighbor node initially. Based on the load conditions on the nearest neighbor, the task may be allocated to the next-nearest neighbor, and so on.
- *Local-only*, where the computation task is sent only to the UAS, with which the ground unit has an active link.
- *App-Server only*, where the all computation tasks should be done at the server.

From Fig. 8b, we infer that the performance of FOCUS is better than the others under low to medium network load conditions, at around 150ms. It slowly begins to approach the computational response time of *Nearest first* as the network load increases to 2Mbps. This happens because with increasing network load, it becomes more beneficial for FOCUS to allocate computational tasks on the nodes nearest to the ground station (UAS 1 and 2) to mitigate the negative effects of forwarding delay on highly saturated links (e.g., the path from UAS 3 to 4). The *Local-only* approach performs worse because the task allocation is done to only those nodes that are within 1 hop from the ground station. This approach is quite immune to the increase in network load. However, the computation response time is higher than FOCUS because the task is not allocated to the optimal UAS, based on the global network knowledge. Doing the computing task on the server in the *App-Server only* approach is not scalable, since the computation response time increases exponentially with the network load.

The maximum number of computational tasks that are handled per minute in the network under increasing network load is shown in Fig. 8c. FOCUS and *Nearest first* provide higher capacity running the most number of computations per minute. *Local-only* and *App-server only* approaches result in much less capacity in terms of computations per minute, since they are localized to certain specific nodes in the network.

From these experimental results, we see that FOCUS incurs the minimum computation response time while having the capability to run the highest number of computations per minute, when compared to other classical methods.

B. Simulation results

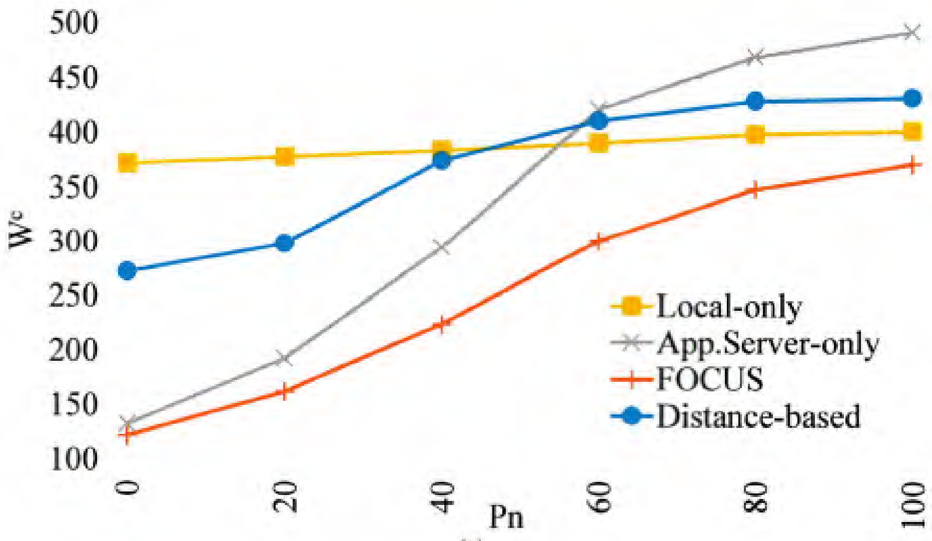
Next, we evaluate the performance of FOCUS through a numerical simulation to study large scale scenarios. We consider a grid topology in which the UASs are placed at equal distances and are connected in a 'cross formation', where each UAS can have at most four neighbors. We then place the GCS at one corner of the grid using the model described in Section IV. We define P_c and P_n as the probability that each UAS will receive $\Omega_i^{(c)}$ and $\Omega_i^{(n)}$ from the ground units, respectively. There are 40 UASs and we fix the value for the sets $\Upsilon^{(c)}$ and $\Upsilon^{(n)}$. Unless specified otherwise, we use these values for the model parameters: $\mu_{N+1}^{(c)}=100$, $\mu_i^{(c)}=5$, $\mu_i^{(n)}=125$, $\Omega_i^{(c)}=2$, $\Omega_i^{(n)}=3$ (in Mbps) and, $P_n=0.75$, $P_c=0.8$.

Fig. 9a shows $\mathcal{W}^{(c)}$ value generally increases with the only-network traffic. Here, we also show the *Distance-based* method that corresponds to a greedy algorithm in which a node sends its computation requests to the GCS only if its distance to the latter is less then half of the network graph diameter. Else, it shares the requests among its neighbors. The *Local-only* method is not affected by the network traffic; the *App-Server only* method works well with low traffic load but it is the worst when the traffic load become high. The *Distance-based* method combines both cloud computation and fog computation but as soon as the network become congested close to the GCS, the performance drops. Finally, FOCUS is able to cope with different traffic loads, striking a balance between fog and cloud computation.

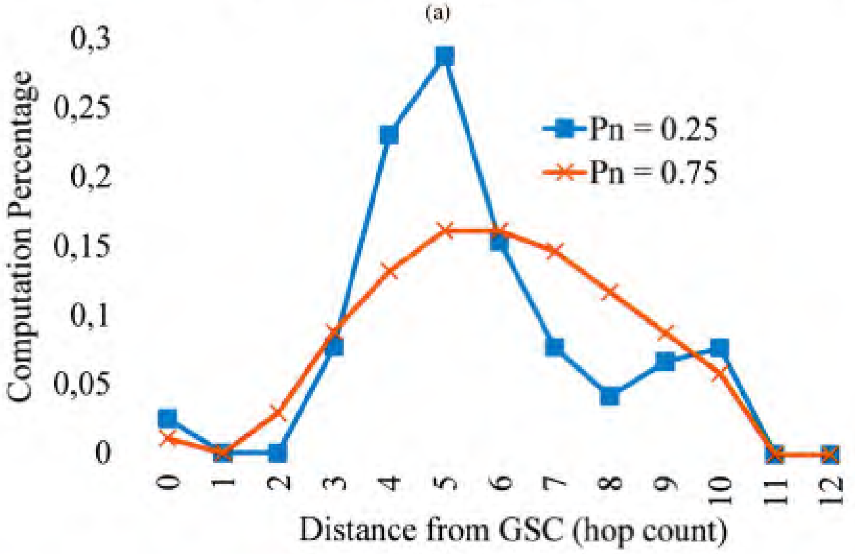
In Fig. 9b, we show how FOCUS distributes the computation requests along the UAN. Here we plot two values for P_n: 0.25 and 0.75. We see when network traffic is high, the cloud (point 0 in x-axis) is not preferred. However, with lower P_n some computation occurs in the cloud. With low network load, the computation is largely contained in the middle section of the UAN (as the GCS is at one corner of the grid). At the same time, we have few UASs that are part of the only-network path towards the GCS while some others have their network queue empty. If we increase the P_n, we see the computation is spread uniformly around the network. This is because of a more uniform distribution of the intra-network bound packets. Thus, FOCUS dynamically distributes fog computation tasks around the network based on the traffic conditions.

VII. Conclusion and Future Work

We proposed a fog computing architecture, called FOCUS, for UAS software-defined mesh networks. We first showed that increasing CPU utilization of UAS has negligible effect on flight time, and characterized the trade-off between computation time/location under different network and computation loads. Then, we formulated the joint problem of network and computation flow optimization, with heuristics having well defined complexity, along with a systems-level implementation. Experiments and simulations validated the approach of allocating computational tasks in a principled manner, revealing over 32% latency improvement compared to greedy or end-server only allocation methods.



(a)



(b)

Fig. 9: (a) $\mathcal{W}^{(c)}$ varying network-only traffic P_n, (b) Percentage of computation executed w.r.t. the distance from the *GCS*.

References

[1] McKinsey&Company, "Commercial drones are here: The future of unmanned aerial systems," *Technical Report*, 2017.

[2] L. Gupta, J. Rain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Commun. Surv. and Tut.*, vol. 18, no. 2, pp. 1123–1152, 2005.

[3] "OpenDayLight - Software Defined Network and Network Function Virtualization," https://www.opendaylight.org/, accessed on April, 2018.

[4] H. Genc, Y. Zu, T. W. Chin, M. Halpern, and V. J. Reddi, "Flying IoT: Toward Low-Power Vision in the Sky," *IEEE Micro*, vol. 37, no. 6, pp. 40–51, November 2017.

[5] I. Bekmezci, O. Koray Sahingoz, and S. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Netw. (Elsevier)*, vol. 11, no. 3, pp. 1254–1270, 2013.

[6] C. Rametta and G. Schembra, "Designing a softwarized network deployed on a fleet of drones for rural zone monitoring," *Future Internet*, vol. 9, no. 1, pp. 1–21, 2017.

[7] K. J. S. White, E. Denney, M. D. Knudson, A. K. Mamerides, and D. P. Pezaros, "A programmable SDN+NFV-based architecture for UAV telemetry monitoring," in *14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan 2017, pp. 522–527.

[8] B. Barritt, T. Kichkaylo, K. Mandke, A. Zalcman, and V. Lin, "Operating a uav mesh internet backhaul network using temporospatial sdn," in *2017 IEEE Aerosp. Conf.*, March 2017, pp. 1–7.

[9] Z. Yuan, X. Huang, L. Sun, and J. Jin, "Software defined mobile sensor network for micro uav swarm," *IEEE Int.Conf. on Control and Robot. Eng. (ICCRE)*, 2016.

[10] Z. Zhang, H. Wang, and H. Zhao, "An sdn framework for uav backbone network towards knowledge centric networking," in *IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS): Knowl. Centric Netw.*, 2018, pp. 456–461.

[11] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Utilizing fog comput. for multi-robot syst." in *2018 2nd IEEE Int. Conf. on Robotic Computing (IRC)*, Jan 2018, pp. 102–105.

[12] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-Based IoT Platform: A Crowd Surveillance Use Case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, February 2017.

[13] X. Wang, A. Chowdhery, and M. Chiang, "Networked drone cameras for sports streaming," in *2017 IEEE 37th Int. Conf. on Distrib. Comput. Syst. (ICDCS)*, June 2017, pp. 308–318.

[14] R. I. Meneguette, A. Boukerche, and A. H. M. Pimenta, "Avarac: An availability-based resource allocation scheme for vehicular cloud," *IEEE Trans. on Intell. Transp. Sys.*, pp. 1–12, 2018.
[15] Y. Hui, Z. Su, T. H. Luan, and J. Cai, "Content in motion: An edge computing based relay scheme for content dissemination in urban vehicular networks," *IEEE Trans. on Intell. Transp. Sys.*, pp. 1–14, 2018.
[16] R. Florin and S. Olariu, "Toward approximating job completion time in vehicular clouds," *IEEE Trans. on Intell. Transp. Sys.*, pp. 1–10, 2018.
[17] S. Jeong, O. Simeone, and J. Kang, "Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning," *IEEE Trans. on Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, 2018.
[18] M. Narang, S. Xiang, W. Liu, J. Gutierrez, L. Chiaraviglio, A. Sathiaseelan, and A. Merwaday, "UAV-assisted edge infrastructure for challenged networks," in *2017 IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 60–65.
[19] A. Kattepur, H. K. Rath, and A. Simha, "A-Priori Estimation of Computation Times in Fog Networked Robotics," in *2017 IEEE Int. Conf. on Edge Computing (EDGE)*, June 2017, pp. 9–16.
[20] M. A. Messous, H. Sedjelmaci, N. Houari, and S. M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *IEEE ICC*, May 2017, pp. 1–6.
[21] H. Shakhatreh, A. Khreishah, J. Chakareski, H. B. Salameh, and I. Khalil, "On the continuous coverage problem for a swarm of uavs," in *2016 IEEE 37th Sarnoff Symp.*, Sep. 2016, pp. 130–135.
[22] A. Trotta, M. D. Felice, F. Montori, K. R. Chowdhury, and L. Bononi, "Joint coverage, connectivity, and charging strategies for distributed uav networks," *IEEE Trans. on Robot.*, vol. 34, no. 4, pp. 883–900, 2018.
[23] "Stress - Simple workload generator for POSIX systems," http://people.seas.harvard.edu/ apw/stress/, accessed on May, 2018.
[24] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. New York, USA: Wiley-Interscience, 2008.
[25] E. Bonfoh, S. Medjiah, and C. Chassot, "A parsimonious monitoring approach for link bandwidth estimation within sdn-based networks," in *4th IEEE Conf. on Netw. Softwarization and Workshops (NetSoft)*, 2018.
[26] G. Baier, E. Köhler, and M. Skutella, "The k-splittable flow problem," *Algorithmica*, vol. 42, no. 3, pp. 231–248, Jul 2005.
[27] D. P. Bertsekas and D. A. Castañon, "A forward/reverse auction algorithm for asymmetric assignment problems," *Comput. Optim. and Appl.*, vol. 1, no. 3, pp. 277–297, Dec 1992.
[28] "DroneCodeSDK - MAVLink API library for the dronecode platform," https://www.dronecode.org/sdk/, accessed on March, 2018.
[29] "Docker - Docker, Inc. - Operating System Level Virtualization," https://www.docker.com, accessed on May, 2018.
[30] "OpenvSwitch - Linux Foundation, Collaborative Projects - OpenvSwitch," https://www.openvswitch.org/, accessed on Feb., 2018.
[31] "B.A.T.M.A.N - Better Approach To Mobile Ad-hoc Networking," https://www.open-mesh.org/, accessed on Feb., 2018.
[32] C. M, L. V, S. C, and F. P., "Binary robust independent elementary features," *Eur. Conf. on comput. vision*, pp. 778–792, 2010.

ONR N00014-17-1-2046 - PROJECT 7 -Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology Technical Lead: Albert-László Barabási

Nima Dehmamy[*]
CSSI, Kellogg School of Management
Northwestern University, Evanston, IL
`nimadt@bu.edu`

Albert-László Barabási[†]
Center for Complex Network Research,
Northeastern University, Boston MA
`alb@neu.edu`

Rose Yu
Khoury College of Computer Sciences,
Northeastern University, Boston, MA
`roseyu@northeastern.edu`

Abstract

To deepen our understanding of graph neural networks, we investigate the representation power of Graph Convolutional Networks (GCN) through the looking glass of *graph moments*, a key property of graph topology encoding path of various lengths. We find that GCNs are rather restrictive in learning graph moments. Without careful design, GCNs can fail miserably even with multiple layers and nonlinear activation functions. We analyze theoretically the expressiveness of GCNs, concluding that a modular GCN design, using different propagation rules with residual connections could significantly improve the performance of GCN. We demonstrate that such modular designs are capable of distinguishing graphs from different graph generation models for surprisingly small graphs, a notoriously difficult problem in network science. Our investigation suggests that, depth is much more influential than width, with deeper GCNs being more capable of learning higher order graph moments. Additionally, combining GCN modules with different propagation rules is critical to the representation power of GCNs.

1 Introduction

The surprising effectiveness of graph neural networks [17] has led to an explosion of interests in graph representation learning, leading to applications from particle physics [12], to molecular biology [37] to robotics [4]. We refer readers to several recent surveys [7, 38, 33, 14] and the references therein for a non-exhaustive list of the research. Graph convolution networks (GCNs) are among the most popular graph neural network models. In contrast to existing deep learning architectures, GCNs are known to contain fewer number of parameters, can handle irregular grids with non-Euclidean geometry, and introduce relational inductive bias into data-driven systems. It is therefore commonly believed that graph neural networks can learn arbitrary representations of graph data.

Despite their practical success, most GCNs are deployed as black boxes feature extractors for graph data. It is not yet clear to what extent can these models capture different graph features. One prominent feature of graph data is *node permutation invariance*: many graph structures stay the same

[*]work done when at Center for Complex Network Research, Northeastern University, Boston, MA

[†]Center for Cancer Systems Biology, Dana Farber Cancer Institute, Boston MA, Brigham and Women's Hospital, Harvard Medical School, Boston MA, Center for Network Science, Central European University, Budapest, Hungary

under relabelling or permutations of the nodes. For instance, people in a friendship network may be following a similar pattern for making friends, in similar cultures. To satisfy permutation invariance, GCNs assign global parameters to all the nodes by which significantly simplifies learning. But such efficiency comes at the cost of expressiveness: GCNs are *not* universal function approximators [34]. We use GCN in a broader sense than in [20], allowing different propagation rules (see below (4)).

To obtain deeper understanding of graph neural networks, a few recent work have investigated the behavior of GCNs including expressiveness and generalizations. For example, [28] showed that message passing GCNs can approximate measurable functions in probability. [34, 24, 25] defined expressiveness as the capability of learning multi-set functions and proved that GCNs are at most as powerful as the Weisfeiler-Lehman test for graph isomorphism, but assuming GCNs with infinite number of hidden units and layers. [32] analyzed the generalization and stability of GCNs, which suggests that the generalization gap of GCNs depends on the eigenvalues of the graph filters. However, their analysis is limited to a single layer GCN for semi-supervised learning tasks. Up until now, the representation power of multi-layer GCNs for learning graph topology remains elusive.

In this work, we analyze the representation power of GCNs in learning graph topology using *graph moments*, capturing key features of the underlying random process from which a graph is produced. We argue that enforcing node permutation invariance is restricting the representation power of GCNs. We discover pathological cases for learning graph moments with GCNs. We derive the representation power in terms of number of hidden units (width), number of layers (depths), and propagation rules. We show how a modular design for GCNs with different propagation rules significantly improves the representation power of GCN-based architectures. We apply our modular GCNs to distinguish different graph topology from small graphs. Our experiments show that depth is much more influential than width in learning graph moments and combining different GCN modules can greatly improve the representation power of GCNs. [3]

In summary, our contributions in this work include:

- We reveal the limitations of graph convolutional networks in learning graph topology. For learning graph moments, certain designs GCN completely fails, even with multiple layers and non-linear activation functions.
- we provide theoretical guarantees for the representation power of GCN for learning graph moments, which suggests a strict dependence on the depth whereas the width plays a weaker role in many cases.
- We take a modular approach in designing GCNs that can learn a large class of node permutation invariant function of of the graph, including non-smooth functions. We find that having different graph propagation rules with residual connections can dramatically increase the representation power of GCNs.
- We apply our approach to build a "graph stethoscope": given a graph, classify its generating process or topology. We provide experimental evidence to validate our theoretical analysis and the benefits of a modular approach.

Notation and Definitions A graph is a set of N nodes connected via a set of edges. The adjacency matrix of a graph A encodes graph topology, where each element A_{ij} represents an edge from node i to node j. We use AB and $A \cdot B$ (if more than two indices may be present) to denote the matrix product of matrices A and B. All multiplications and exponentiations are matrix products, unless explicitly stated. Lower indices A_{ij} denote i, jth elements of A, and A_i means the ith row. A^p denotes the pth matrix power of A. We use $a^{(m)}$ to denote a parameter of the mth layer.

2 Learning Graph Moments

Given a collection of graphs, produced by an unknown random graph generation process, learning from graphs requires us to accurately infer the characteristics of the underlying generation process. Similar to how moments $\mathbb{E}[X^p]$ of a random variable X characterize its probability distribution, graph moments [5, 23] characterize the random process from which the graph is generated.

[3] All code and hyperparameters are available at `https://github.com/nimadehmamy/Understanding-GCN`

2.1 Graph moments

In general, a pth order graph moment M_p is the ensemble average of an order p polynomial of A

$$M_p(A) = \prod_{q=1}^{p}(A \cdot W_q + B_q) \tag{1}$$

with W_q and B_q being $N \times N$ matrices. Under the constraint of node permutation invariance, W_q must be either proportional to the identity matrix, or a uniform aggregation matrix. Formally,

$$M(A) = A \cdot W + B, \quad \text{Node Permutation Invariance} \Rightarrow \quad W, B = cI, \quad \text{or} \quad W, B = c\mathbf{1}\mathbf{1}^T \tag{2}$$

where $\mathbf{1}$ is a vector of ones. Graph moments encode topological information of a graph and are useful for graph coloring and Hamiltonicity. For instance, graph power A^p_{ij} counts the number of paths from node i to j of length p. For a graph of size N, A has N eigenvalues. Applying eigenvalue decomposition to graph moments, we have $\mathbb{E}[A^p] = \mathbb{E}[(V^T \Lambda U)^p]) = V^T \mathbb{E}[\Lambda^p] U$. Graphs moments correspond to the distribution of the eigenvalues Λ, which are random variables that characterize the graph generation process. Graph moments are node permutation invariant, meaning that relabelling of the nodes will not change the distribution of degrees, the paths of a given length, or the number of triangles, to name a few. The problem of learning graph moments is to learn a functional approximator F such that $F : A \to M_p(A)$, while preserving node permutation invariance.

Different graph generation processes can depend on different orders of graph moments. For example, in Barabási-Albert (BA) model [1], the probability of adding a new edge is proportional to the degree, which is a first order graph moment. In diffusion processes, however, the stationary distribution depends on the normalized adjacency matrix $\hat{A}$ as well as its symmetrized version $\hat{A}_s$, defined as follows:

$$D_{ij} \equiv \delta_{ij} \sum_k A_{ik} \qquad \hat{A} \equiv D^{-1}A \qquad \hat{A}_s \equiv D^{-1/2}AD^{-1/2} \tag{3}$$

which are *not* smooth functions of A and have no Taylor expansion in A, because of the inverse D^{-1}. Processes involving D^{-1} and A are common and per (2) D and $\text{Tr}[A]$ are the only node permutation invariant first order moments of A. Thus, in order to approximate more general node permutation invariant $F(A)$, it is crucial for a graph neural network to be able to learn moments of A, $\hat{A}$ and $\hat{A}_s$ simultaneously. In general, non-smooth functions of A can depend on A^{-1}, which may be important for inverting a diffusion process. We will only focus on using A, $\hat{A}$ and $\hat{A}_s$ here, but all argument hold also if we include A^{-1}, $\hat{A}^{-1}$ and $\hat{A}_s^{-1}$ as well.

2.2 Learning with Fully Connected Networks

Consider a toy example of learning the first order moment. Given a collection of graphs with $N = 20$ nodes, the inputs are their adjacency matrices A, and the outputs are the node degrees $D_i = \sum_{j=1}^{N} A_{ij}$. For a fully connected (FC) neural network, it is a rather simple task given its universal approximation power [19]. However, a FC network treats the adjacency matrices as vector inputs and ignores the underlying graph structures, it needs a large amount of training samples and many parameters to learn properly.

Fig. 1 shows the mean squared error (MSE) of a single layer FC network in learning the first order moments. Each curve corresponds to different number of training samples, ranging from 500–10,000. The horizontal axis shows the number of hidden units. We can see that even though the network can learn the moments properly reaching an MSE of $\approx 10^{-4}$, it requires the same order of magnitude of hidden units as the number of nodes in the graph, and at least $1,000$ samples. Therefore, FC networks are quite inefficient for learning graph moments, which motivates us to look into more power alternatives: graph convolution networks.

2.3 Learning with Graph Convolutional Networks

We consider the following class of graph convolutional networks. A single layer GCN propagates the node attributes h using a function $f(A)$ of the adjacency matrix and has an output given by

$$F(A, h) = \sigma\left(f(A) \cdot h \cdot W + b\right) \tag{4}$$

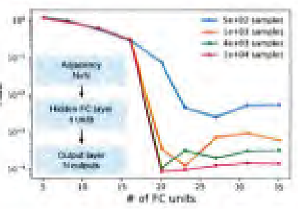


Figure 1: Learning graph moments (Erdős-Rényi graph) with a single fully-connected layer. Best validation MSE w.r.t number of hidden units n and the number of samples in the training data (curves of different colors).

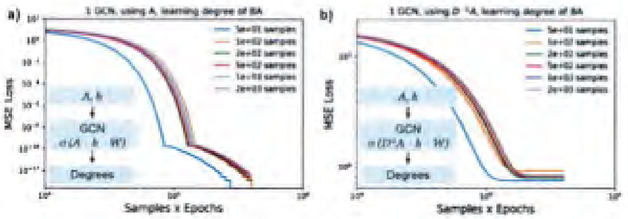


Figure 2: Learning the degree of nodes in a graph with a single layer of GCN. When the GCN layer is designed as $\sigma(A \cdot h \cdot W)$ with linear activation function $\sigma(x) = x$, the network easily learns the degree (a). However, if the network uses the propagation rule as $\sigma(D^{-1}A \cdot h \cdot W)$, it fails to learn degree, with very high MSE loss (b). The training data were instances of Barabasi-Albert graphs (preferential attachment) with $N = 20$ nodes and $m = 2$ initial edges.

where f is called the propagation rule, h_i is the attribute of node i, W is the weight matrix and b is the bias. As we are interested in the graph topology, we ignore the node attributes and set $h_i = 1$. Note that the weights W are only coupled to the node attributes h but not to the propagation rule $f(A)$. The definition in Eqn (4) covers a broad class of GCNs. For example, GCN in [20] uses $f = D^{-1/2}AD^{-1/2}$. GraphSAGE [16] mean aggregator is equivalent to $f = D^{-1}A$. These architectures are also special cases of Message-Passing Neural Networks [13].

We apply a single layer GCN with different propagation rules to learn the node degrees of BA graphs. With linear activation $\sigma(x) = x$, the solution for learning node degrees is $f(A) = A$, $W = 1$ and $b = 0$. For high-order graph moments of the form $M_p = \sum_j (A^p)_{ij}$, a single layer GCN has to learn the function $f(A) = A^p$. As shown in Figure 2, a single layer GCN with $f(A) = A$ can learn the degrees perfectly even with as few as 50 training samples for a graph of $N = 20$ nodes (Fig. 2a). Note that GCN only requires 1 hidden unit to learn, which is much more efficient than the FC networks. However, if we set the learning target as $f(A) = D^{-1}A$, the same GCN completely fails at learning the graph moments regardless of the sample size, as shown in Fig. 2b. This demonstrates the limitation of GCNs due to the permutation invariance constraint. Next we analyze this phenomena and provide theoretical guarantees for the representation power of GCNs.

3 Theoretical Analysis

To learn graph topology, fully connected layers require a large number of hidden units. The following theorem characterizes the representation power of fully connected neural network for learning graph moments in terms of number of nodes N, order of moments p and number of hidden units n.

Theorem 1. *A fully connected neural network with one hidden layer requires $n > O(C_f^2) \sim O(p^2N^{2q})$ number of neurons in the best case with $1 \leq q \leq 2$ to learn a graph moment of order p for graphs with N nodes. Additionally, it also needs $S > O(nd) \sim O\left(p^2N^{2q+2}\right)$ number of samples to make the learning tractable.*

Clearly, if a FC network fully parameterizes every element in a $N \times N$ adjacency matrix A, the dimensions of the input would have to be $d = N^2$. If the FC network allows weight sharing among nodes, the input dimension would be $d = N$. The Fourier transform of a polynomial function of order p with $O(1)$ coefficients will have an L_1 norms of $C_f \sim O(p)$. Using Barron's result [2] with $d = N^q$, where $1 \leq q \leq 2$ and set the $C_f \sim O(p)$, we can obtain the approximation bound.

In contrast to fully connected neural networks, graph convolutional networks are more efficient in learning graph moments. A graph convolution network layer without bias is of the form:

$$F(A, h) = \sigma(f(A) \cdot h \cdot W) \tag{5}$$

Permutation invariance restricts the weight matrix W to be either proportional to the identity matrix, or a uniform aggregation matrix, see Eqn. (2). When $W = cI$, the resulting graph moment $M_p(A)$ has exactly the form of the output of a p layer GCN with linear activation function.

We first show, via an explicit example, that a $n < p$ layer GCN by stacking layers of the form in Eqn. (5) cannot learn pth order graph moments.

Lemma 1. *A graph convolutional network with $n < p$ layers cannot, in general, learn a graph moment of order p for a set of random graphs.*

We prove this by showing a counterexample. Consider a directed graph of two nodes with adjacency matrix $A = \begin{pmatrix} 0 & a \\ b & 0 \end{pmatrix}$. Suppose we want to use a single layer GCN to learn the second order moment $f(A)_i = \sum_j (A^2)_{ij} = \sum_k A_{ik} D_k$. The node attributes h_{il} are decoupled from the propagation rule $f(A)_i$. Their values are set to ones $h_{il} = 1$, or any values independent of A. The network tries to learn the weight matrix $W_{l\mu}$ and has an output $h^{(1)}$ of the form

$$h^{(1)}_{i\mu} = \sigma\left(A \cdot h \cdot W\right)_{i\mu} = \sigma\left(\sum_{j,l} A_{ij} h_{jl} W_{l\mu}\right), \tag{6}$$

For brevity, define $V_{i\mu} \equiv \sum_l h_{il} W_{l\mu}$. Setting the output $h^{(1)}$ to the desired function $A \cdot D$, with components $h^{(1)}_{1\mu} = h^{(1)}_{2\mu} = ab$, (hence μ can only be 1) and plugging in A, the two components of the output become

$$h^{(1)}_{1\mu} = \sigma\left(D_1 V_{1\mu}\right) = \sigma\left(a V_{1\mu}\right) = ab \qquad h^{(1)}_{2\mu} = \sigma\left(D_2 V_{2\mu}\right) = \sigma\left(b V_{2\mu}\right) = ab. \tag{7}$$

which must be satisfied $\forall a, b$. But it's impossible to satisfy $\sigma\left(a V_{1\mu}\right) = ab$ for $(a, b) \in \mathbb{R}^2$ with $V_{1\mu}$ and $\sigma(\cdot)$ independent of a, b. □

Proposition 1. *A graph convolutional network with n layers, and no bias terms, in general, can learn $f(A)_i = \sum_j \left(A^p\right)_{ij}$ only if $n = p$ or $n > p$ if the bias is allowed.*

If we use a two layer GCN to learn a first order moment $f(A)_i = \sum_j A_{ij} = D_i$, for the output of the second layer $h^{(2)}_{i\nu}$ we have

$$h^{(2)} = \sigma^{(2)}\left(A \cdot \sigma^{(1)}\left(A \cdot h \cdot W^{(1)}\right) \cdot W^{(2)}\right),\ h^{(2)}_{1\nu} = \sigma^{(2)}\left(a \sum_\mu \sigma^{(1)}\left(b V^{(1)}_{2\mu}\right) W^{(2)}_{\mu\nu}\right) = a \tag{8}$$

Again, since this must hold for any value of a, b and ν, we see that $h^{(2)}_{1\nu}$ is a function of b through the output of the first layer $h^{(1)}_{2\mu}$. Thus $h^{(2)}_{1\nu} = a$ can only be satisfied if the first layer output is a constant. In other words, only if the first layer can be bypassed (e.g. if the bias is large and weights are zero) can a two-layer GCN learn the first order moment. □

This result also generalizes to multiple layers and higher order moments in a straightforward fashion. For GCN with linear activation, a similar argument shows that when the node attributes h are not implicitly a function of A, in order to learn the function $\sum_j \left(A^p\right)_{ij}$, we need to have exactly $n = p$ GCN layers, without bias. With bias, a feed-forward GCN with $n > p$ layers can learn single term order p moments such as $\sum_j \left(A^p\right)_{ij}$. However, since it needs to set the some weights of $n - p$ layers to zero, it can fail in learning mixed order moments such as $\sum_j (A^q + A^p)_{ij}$.

To allow GCNs with very few parameters to learn mixed order moments, we introduce residual connections [18] by concatenating the output of every layer $[h^{(1)}, \ldots, h^{(m)}]$ to the final output of the network. This way, by applying an aggregation layer or a FC layer which acts the same way on the output for every node, we can approximate any polynomial function of graph moments. Specifically, the final $N \times d^o$ output $h^{(final)}$ of the aggregation layer has the form

$$h^{(final)}_{i\mu} = \sigma\left(\sum_{m=1}^{n} a^{(m)}_{\mu} \cdot h^{(m)}_{i}\right), \qquad h^{(m)} = \sigma(A \cdot h^{(m-1)} \cdot W^{(m)} + b^{(m)}), \tag{9}$$

where $\cdot$ acts on the output channels of each output layers. The above results lead to the following theorem which guarantees the representation power of multi-layer GCNs with respect to learning graph moments.

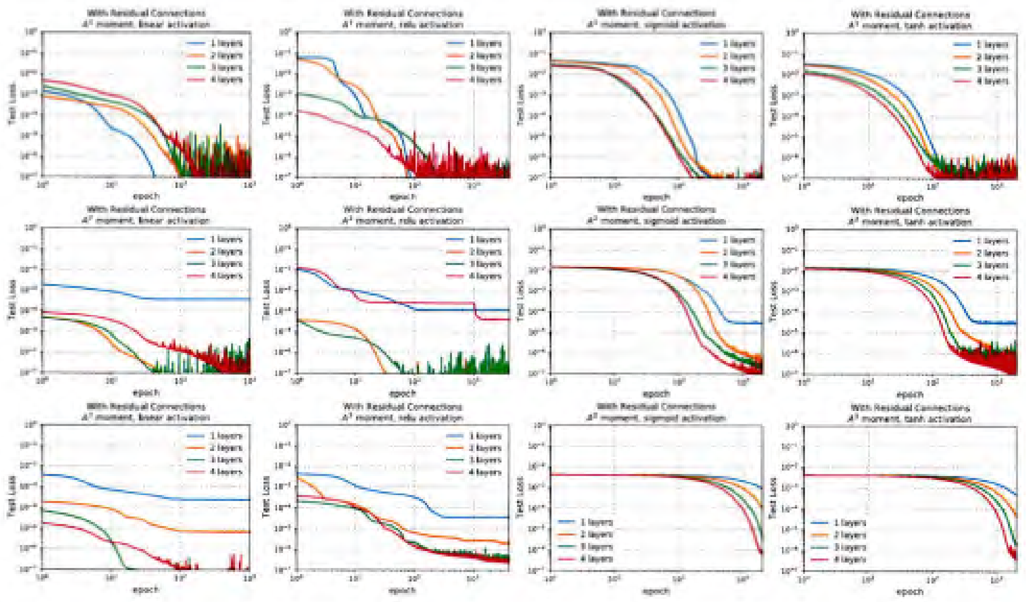


Figure 4: Test loss over number of epochs for learning first (top), second (middle) and third (bottom) order graph moments $M_p(A) = \sum_j (A^p)_{ij}$, with varying number of layers and different activation functions. A multi-layer GCN with residual connections is capable of learning the graph moments when the number of layers is at least the target order of the graph moments. The graphs are from our synthetic graph dataset described in Sec. 6.

Theorem 2. *With the number of layers n greater or equal to the order p of a graph moment $M_p(A)$, graph convolutional networks with residual connections can learn a graph moment M_p with $O(p)$ number of neurons, independent of the size of the graph.*

Theorem 2 suggests that the representation power of GCN has a strong dependence on the number of layers (depth) rather than the size of the graph (width). It also highlights the importance of residual connections. By introducing residual connections into multiple GCN layers, we can learn any polynomial function of graph moments with linear activation. Interestingly, Graph Isomophism Network (GIN) proposed in [34] uses the following propagation rule:

$$F(A, h) = \sigma\left([(1 + \epsilon)I + A] \cdot h \cdot W\right) \tag{10}$$

which is a special case of our GCN with one residual connection between two modules.

4 Modular GCN Design

In order to overcome the limitation of the GCNs in learning graph moments, we take a modular approach to GCN design. We treat different GCN propagation rules as different "modules" and consider three important GCN modules (1) $f_1 = A$ [22] (2) $f_2 = D^{-1}A$ [20], and (3) $f_3 = D^{-1/2}AD^{-1/2}$ [16]. Figure 3a) shows the design of a single GCN layer where we combine three different GCN modules. The output of the modules are concatenated and fed into a node-wise FC layer. Note that our design is different from the multi-head attention mechanism in Graph Attention Network [31] which uses the same propagation rule for all the modules.

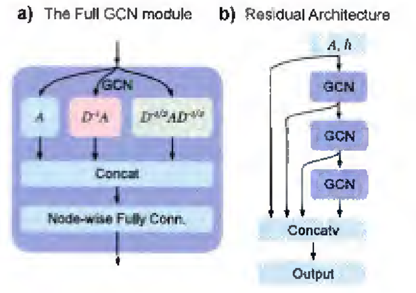


Figure 3: GCN layer (a), using three different propagation rules and a node-wise FC layer. Using residual connections (b) allows a n-layer modular GCN to learn any polynomial function of order n of its constituent operators.

However, simply stacking GCN layers on top of each other in a feed-forward fashion is quite restrictive, as shown by our theoretical analysis for multi-layer GCNs. Different propagation rules cannot be written as Taylor expansions of each other, while all of them are important in modeling the graph generation process. Hence, no matter how many layers or how non-linear the activation function gets, multi-layer GCN stacked in a feed-forward way cannot learn network moments whose order is not precisely the number of layers. If we add residual connections from the output of every layer to the final aggregation layer, we would be able to approximate any polynomial functions of graph moments. Figure 3b) shows the design of a muli-layer GCN with residual connections. We stack the modular GCN layer on top of each other and concatenate the residual connections from every layer. The final layer aggregates the output from all previous layers, including residual connections.

We measure the representation power of GCN design in learning different orders of graph moments $M_p(A) = \sum_j (A^p)_{ij}$ with $p = 1, 2, 3$. Figure 4 shows the test loss over number of epochs for learning first (top), second (middle) and third (bottom) order graph moments. We vary the number of layers from 1 to 4 and test with different activation functions including linear, ReLU, sigmoid and tanh. Consistent with the theoretical analysis, we observe that whenever the number of layers is at least the target order of the graph moments, a multi-layer GCN with residual connections is capable of learning the graph moments. Interestingly, Jumping Knowledge (JK) Networks [35] showed similar effects of adding residual connections for Message Passing Graph Neural Networks.

Our modular approach demonstrates the importance of architectural design when using specialized neural networks. Due to permutation invariance, feed-forward GCNs are quite limited in their representation power and can fail at learning graph topology. However, with careful design including different propagation rules and residual connections, it is possible to improve the representation power of GCNs in order to capture higher order graph moments while preserving permutation invariance.

5 Related Work

Graph Representation Learning There has been increasing interest in deep learning on graphs, see e.g. many recent surveys of the field [7, 38, 33]. Graph neural networks [22, 20, 17] can learn complex representations of graph data. For example, Hopfield networks [28, 22] propagate the hidden states to a fixed point and use the steady state representation as the embedding for a graph; Graph convolution networks [8, 20] generalize the convolutional operation from convolutional neural networks to learn from geometric objects beyond regular grids. [21] proposes a deep architecture for long-term forecasting of spatiotemporal graphs. [37] learns the representations for generating random graphs sequentially using an adversarial loss at each step. Despite practical success, deep understanding and theoretical analysis of graph neural networks is still largely lacking.

Expressiveness of Neural Networks Early results on the expressiveness of neural networks take a highly theoretical approach, from using functional analysis to show universal approximation results [19], to studying network VC dimension [3]. While these results provided theoretically general conclusions, they mostly focus on single layer shallow networks. For deep fully connected networks, several recent papers have focused on understanding the benefits of depth for neural networks [11, 29, 28, 27]) with specific choice of weights. For graph neural networks, [34, 24, 25] prove the equivalence of a graph neural network with Weisfeiler-Lehman graph isomorphism test with infinite number of hidden layers. [32] analyzes the generalization and stability of GCNs, which depends on eigenvalues of the graph filters. However, their analysis is limited to a single layer GCN in the semi-supervised learning setting. Most recently, [10] demonstrates the equivalence between infinitely wide multi-layer GNNs and Graph Neural Tangent Kernels, which enjoy polynomial sample complexity guarantees.

Distinguishing Graph Generation Models Understanding random graph generation processes has been a long lasting interest of network analysis. Characterizing the similarities and differences of generation models has applications in, for example, graph classification: categorizing a collections of graphs based on either node attributes or graph topology. Traditional graph classification approaches rely heavily on feature engineering and hand designed similarity measures [30, 15]. Several recent work propose to leverage deep architecture [6, 36, 9] and learn graph similarities at the representation level. In this work, instead of proposing yet another deep architecture for graph classification, we provide insights for the representation power of GCNs using well-known generation models. Our insights can provide guidance for choosing similarity measures in graph classification.

6 Graph Stethoscope: Distinguishing Graph Generation Models

An important application of learning graph moments is to distinguish different random graph generation models. For random graph generation processes like the BA model, the asymptotic behavior ($N \to \infty$) is known, such as scale-free. However, when the number of nodes is small, it is generally difficult to distinguish collections of graphs with different graph topology if the generation process is random. Thus, building an efficient tool that can probe the structure of small graphs of $N < 50$ like a stethoscope can be highly challenging, especially when all the graphs have the same number of nodes and edges.

BA vs. ER. We consider two tasks for graph stethoscope. In the first setting, we generate $5,000$ graphs with the same number of nodes and vary the number of edges, half of which are from the Barabasi-Albert (BA) model and the other half from the Erdos-Renyi (ER) model. In the BA model, a new node attaches to m existing nodes with a likelihood proportional to the degree of the existing nodes. The $2,500$ BA graphs are evenly split with $m = 1, N/8, N/4, 3N/8, N/2$. To avoid the bias from the order of appearance of nodes caused by preferential attachment, we shuffle the node labels. ER graphs are random undirected graphs with a probability p for generating every edge. We choose four values for p uniformly between $1/N$ and $N/2$. All graphs have similar number of edges.

BA vs. Configuration Model One might argue that distinguishing BA from ER for small graphs is easy as BA graphs are known to have a power-law distribution for the node degrees [1], and ER graphs have a Poisson degree distribution. Hence, we create a much harder task where we compare BA graphs with "fake" BA graphs where the nodes have the same degree but all edges are rewired using the Configuration Model [26] (Config.). The resulting graphs share exactly the same degree distribution. We also find that higher graph moments of the Config BA are difficult to distinguish from real BA, despite the Config. model not fixing these moments. Distinguishing BA and Config BA is very difficult using standard methods such as a Kolmogorov-Smirnov (KS) test. KS test measures the distributional differences of a statistical measure between two graphs and uses hypothesis testing to identify the graph generation model. Figure 5 shows the KS test values for pairs of real-real BA (blue) and pairs of real-fake BA (orange) w.r.t different graph moments. The dashed black lines show the mean of the KS test values for real-real pairs. We observe that the distributions of differences in real-real pairs are almost the same as those of real-fake pairs, meaning the variability in different graph moments among real BA graphs is almost the same as that between real and Config BA graphs.

Table 1: Test accuracy with different modules combinations for BA-ER. $f_1 = A$, $f_2 = D^{-1}A$, and $f_3 = D^{-1/2}AD^{-1/2}$.

Modules	Accuracy
f_1	53.5 %
f_3	76.9 %
f_1, f_3	89.4 %
f_1, f_2, f_3	98.8 %

Classification Using our GCN Module We evaluate the classification accuracy for these two settings using the modular GCN design, and analyze the trends of representation power w.r.t network depth and width, as well as the number of nodes in the graph. Our architecture consists of layers of our GCN module (Fig. 3, linear activation). The output is passed to a fully connected layer with softmax activation, yielding and $N \times c$ matrix (N nodes in graph, c label classes). The final

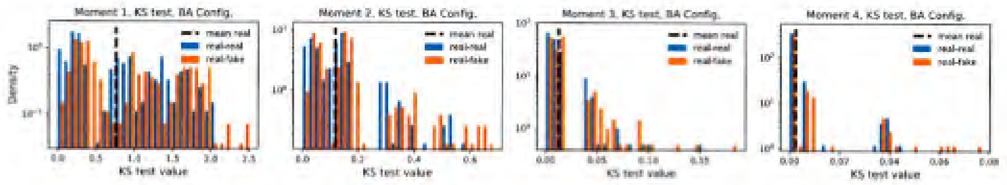


Figure 5: Distribution of Kolmogorov-Smirnov (KS) test values for differences between graph the first four graph moments $\sum_i (A^p)_{ij}$ in the dataset. "real-real" shows the distribution of KS test when comparing the graph moments of two real instances of the BA. All graphs have $N = 30$ nodes, but varying number of links. The "real-fake" case does the KS test for one real BA against one fake BA created using the configuration model.

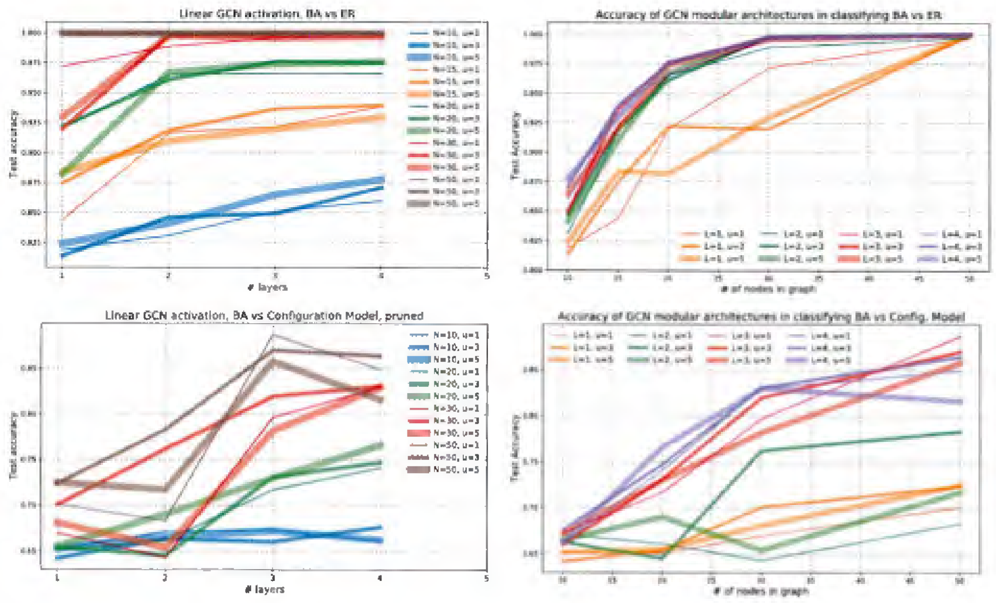


Figure 6: Classify graphs of **Barabasi-Albert** model vs. **Erdos-Renyi** model (top) and **Barabasi-Albert** model vs. **configuration** model (bottom). Left: test accuracy with respect to network depth for different number of nodes (N) and number of units (U). Right: test accuracy with respect to graph size for different number of layers (L) and number of units (U).

classification is found by mean-pooling over the N outputs. We used mean-pooling to aggregate node-level representations, after which a single number is passed to a classification layer. Figure 6 left column shows the accuracy with increasing number of layers for different number of layers and hidden units. We find that depth is more influential than width: increasing one layer can improve the test accuracy by at least 5%, whereas increasing the width has very little effect. The right column is an alternative view with increasing size of the graphs. It is clear that smaller networks are harder to learn, while for $N \geq 50$ nodes is enough for 100% accuracy in BA-ER case. BA-Config is a much harder task, with the highest accuracy of 90%.

We also conduct ablation study for our modular GCN design. Table 1 shows the change of test accuracy when we use different combinations of modules. Note that the number of parameters are kept the same for all different design. We can see that a single module is not enough to distinguish graph generation models with an accuracy close to random guessing. Having all three modules with different propagation rules leads to almost perfect discrimination between BA and ER graphs. This demonstrates the benefits of combining GCN modules to improve its representation power.

7 Conclusion

We conduct a thorough investigation in understanding what can/cannot be learned by GCNs. We focus on graph moments, a key characteristic of graph topology. We found that GCNs are rather restrictive in learning graph moments, and multi-layer GCNs cannot learn graph moments even with nonlinear activation. Theoretical analysis suggests a modular approach in designing graph neural networks while preserving permutation invariance. Modular GCNs are capable of distinguishing different graph generative models for surprisingly small graphs. Our investigation suggests that, for learning graph moments, depth is much more influential than width. Deeper GCNs are more capable of learning higher order graph moments. Our experiments also highlight the importance of combining GCN modules with residual connections in improving the representation power of GCNs.

Acknowledgments

This work was supported in part by NSF #185034, ONR-OTA (N00014-18-9-0001).

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[3] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[5] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. Graph theory, volume 244 of. *Graduate texts in Mathematics*, 2008.

[6] Stephen Bonner, John Brennan, Georgios Theodoropoulos, Ibad Kureshi, and Andrew Stephen McGough. Deep topology classification: A new approach for massive graph classification. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3290–3297. IEEE, 2016.

[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[9] James P Canning, Emma E Ingram, Sammantha Nowak-Wolff, Adriana M Ortiz, Nesreen K Ahmed, Ryan A Rossi, Karl RB Schmitt, and Sucheta Soundarajan. Predicting graph categories from structural properties. *arXiv preprint arXiv:1805.02682*, 2018.

[10] Simon S Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *arXiv preprint arXiv:1905.13192*, 2019.

[11] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940, 2016.

[12] Steven Farrell, Paolo Calafiura, Mayur Mudigonda, Dustin Anderson, Jean-Roch Vlimant, Stephan Zheng, Josh Bendavid, Maria Spiropulu, Giuseppe Cerati, Lindsey Gray, et al. Novel deep learning methods for track reconstruction. *arXiv preprint arXiv:1810.06111*, 2018.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[14] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[15] Ting Guo and Xingquan Zhu. Understanding the roles of sub-graph features for graph classification: an empirical study perspective. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 817–822. ACM, 2013.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*, 2018.

[22] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[23] Yaw-Ling Lin and Steven S Skiena. Algorithms for square roots of graphs. *SIAM Journal on Discrete Mathematics*, 8(1):99–118, 1995.

[24] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[25] Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pages 4663–4673, 2019.

[26] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[27] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2847–2854. JMLR. org, 2017.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[29] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

[30] Johan Ugander, Lars Backstrom, and Jon Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1307–1318. ACM, 2013.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[32] Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. *arXiv preprint arXiv:1905.01004*, 2019.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[35] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

[36] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[37] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

[38] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018.