A Programmable SDN+NFV-based Architecture for UAV Telemetry Monitoring

Kyle J. S. White*, Ewen Denney†, Matt D. Knudson‡ Angelos K. Marnerides§, Dimitrios P. Pezaros*
* School of Computing Science, University of Glasgow, G12 8QQ, Scotland
mail@kylewhite.com, dimitrios.pezaros@glasgow.ac.uk
SGT/NASA†, NASA‡ Ames Research Center, Moffett Field, CA 94035, USA
ewen.denney, matt.knudson@nasa.gov
§InfoLab21, School of Computing & Communications, Lancaster University, LA1 4WA, UK
a.marnerides2@lancaster.ac.uk

Abstract—**The explosive growth in the worldwide use of Unmanned Aerial Vehicles (UAVs) has raised a critical concern with respect to the adequate management of their ad hoc network configuration as required by their mobility management process. As UAVs migrate among ground control stations, associated network services, routing and operational control must also rapidly migrate to ensure a seamless transition. In this paper, we present a novel, lightweight and modular architecture which supports high mobility and situational-awareness through the application of Software Defined Networking (SDN) and Network Function Virtualization (NFV) principles on top of the UAV infrastructure. By combining SDN+NFV programmability we can achieve a robust migration of UAV-related network services, such as network monitoring and anomaly detection as well as smooth UAV migration that confronts high mobility requirements. The proposed container-based monitoring and anomaly detection Network Functions (NFs) as employed within our architecture can be tuned to specific UAV types providing operators better insight during live, high-mobility deployments. We evaluate our architecture against telemetry from over 80 flights from a scientific research UAV infrastructure showing our ability to tune and detect emerging challenges.**

Keywords*—*SDN, NFV, Unmanned Aerial Vehicles, Air Traffic Management, Situational Awareness

I. Introduction

The Federal Aviation Administration (FAA) forecasts that there will be 2.7 million [8] non-hobbyist (>55lbs) commercial UAVs in the U.S. National Airspace System by 2020. On the anticipated introduction of further policy and regulatory frameworks which allow for Beyond Visual Line of Sight (BVLOS) flight, expected numbers look set to increase significantly. This predicted explosion and integration of Unmanned Aerial Systems (UAS) in which UAVs are part of has numerous consequences for the infrastructure of Air Traffic Management (ATM) systems worldwide. Among wider challenges including spectrum allocation, and data security and safety, a core concern is the ad hoc network configuration required for mobility management for UAS. With integrated UAS, the network infrastructure will require the ability to migrate network configuration, including network functions associated with specific UAVs, to ensure resilient, uninterrupted service. Numerous UAS offer migration functionality including the Viking 400. With the FAA transitioning to an IP-based infrastructure through NextGen[1], and a similar modernisation program taking place in Europe through SESAR[2], the opportunities to embed state-of-the-art networking infrastructure support for the emerging UAS needs are currently ideal. Present safety regulation ensures early UAV activities take place in areas above low population density outwith controlled airspace. Perhaps consequently, existing core applications for UAVs of this size have used such regions for tasks including agricultural work, environmental monitoring, oil exploration, wildlife and land management. These distributed areas of operation often suffer from poor networking infrastructure (in terms of both bandwidth and connectivity) even for low levels of demand. To achieve reliable and safe UAS integration into the wider controlled airspace worldwide, resilient, reliable, recoverable, and low-latency network infrastructures and systems are vital. Most currently deployed UAV systems are isolated, operating independently, and often employing their own bespoke protocols, hardware infrastructures and software systems. As standardisation improves, systems which allow for multiple UAV operations management will become the norm. Existing UAS have begun simplifying, easing the process of integration and compatibility. Early military UAV systems required three large rugged server racks for their Ground Control Systems (GCS) comprising radio, pilot-in-command and mission payload racks. Much of this equipment can now be run from a laptop with control software, e.g., the Piccolo controller[3].

For safe and secure operations, UAV operators require real-time telemetry monitoring, alert systems, and mission payload processing as part of the GCS. This information is especially critical for BVLOS operations. Increasingly, there is demand for tailored functionality [10] which can assist the operator with the current task, particular environment and payloads e.g., visual surveillance equipment, heat mapping or crop dusting tools. Since long complex operations can involve multiple tasks, environments and strains, flexibility and both reactive and pro-active adaptability of this functionality over time are also highly desirable attributes. Significant replication of standard telemetry-based monitoring functionality is prevalent, yet with isolated, independent implementations spread across different vendor-specific ground control systems, this scale

[1]https://www.faa.gov/nextgen; Accessed: June 2016
[2]http://www.sesarju.eu; Accessed: June 2016
[3]http://www.cloudcaptech.com; Accessed: June 2016

cannot be beneficially exploited. In this paper, we propose a modular programmable network architecture which, through exploiting the latest paradigms of SDN and NFV, aims to:

- Increased situational-awareness available to pilots and payload operators during UAV missions through dynamic deployment and migration of modular context-specific processing functionality;
- Increased continuity of service in deployments with potentially weak backbone networks, such as on ships and moving vehicles;
- Reduce the latency of telemetry monitoring and applications related to situational-awareness such as anomaly detection through a distributed approach embodied in our architecture;
- Reduce backbone utilisation volumes required for UAS operations in the face of outages or traffic spikes.

Applying our new architecture which places Virtual Network Functions (VNF) at edge switches allows for programmable NFs to be deployed on-demand on the more reliant and greater bandwidth backbone links to GCS, while different types of UAV can remain more lightweight hosts, specialising in their sensor capabilities. Middlebox functionality for UAVs is a vital next step to increase the overall resilience of the wider UAS. This need is emphasised by Tvaryanas' [13] findings that the U.S. military UAV accident rate was as high as 1 per 1,000 flight hours, on aggregate. In comparison, the accident rate for general aviation (manned) flight in the U.S. is 1 per 100,000 flight hours. With greater insight available on demand through tailored NFs which migrate with UAVs, more information will be available to operators to detect and diagnose emerging issues. As Pastor et al. [16] state, in the brief history of UAS accidents, many are directly attributable to errors by pilots attempting to manage unexpected challenging incidents without an adequate situational awareness. Another high-profile UAV accident study found that for most of the aircraft systems, electromechanical failure was more of a causal factor than human error [12]. One critical finding from an analysis of the data is that each of the existing systems is very different, leading to varying kinds of accidents and human factors issues, which strengthens the need for integration and standardisation through a common network architecture and the ability to deploy programmable detection modules which can operate in high mobility environments.

In Section II, we examine a detailed Concept of Operations (ConOps) before presenting our new architecture with design, implementation and routing details in Section III. An evaluation of our architecture is presented as a UAS incident case study in Section IV. We review related work in Section V and Section VI explores future work before concluding the paper.

II. Current Concept of Operations

Figure 1 details the ConOps where our network architecture contributions can be evaluated. The figure represents a typical reconnaissance set up with multiple UAVs of diverse types, operating in various regions with different payload capabilities, e.g., visual or IR cameras. Mobile GCS are on land and at sea, connecting and communicating control information to UAVs within range using radio antennas. Satellite links (and

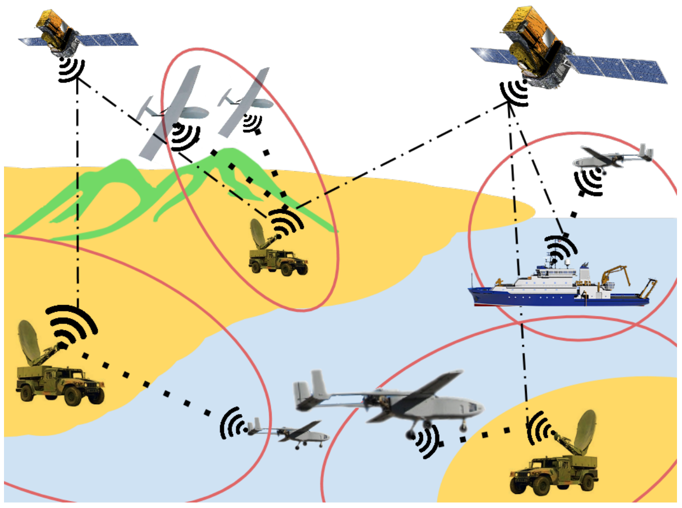
Fig. 1. UAS ConOps for high-mobility communications infrastructure

others, e.g., WiFi, microwave) are used to connect mobile GCS with each other in an abstracted ad hoc mesh network topology. There is also a remote, centralised command and control centre. Some of these links are very costly, such as, e.g., satellite communications. Currently, telemetry analysis, such as any anomaly detection, takes place on data streamed to the centralised control centre via such expensive links or not at all. Many of the UAV GCS uplinks are unreliable, leading to loss of streamed data, where packet latency and out-of-order delivery is equivalent to data loss. Interference from particles or being out of range are some of the common causes of lost link failures. As a result, deploying code to run on UAVs over such links is a poor design choice. Figure 1 also shows UAVs transitioning from sea to land and from higher to lower altitudes. During such transitions, different UAV-specific monitoring and detection modules can assist operators, e.g., calculations for icing alerts at higher, colder altitudes or the rate of increased fuel burn at lower altitudes. The ConOps also shows the lowest UAV transitioning from control on the leftmost GCS to the rightmost GCS. There is a hand-over phase when a UAV migrates to the command and control of another GCS, e.g., due to a change in range or a primary GCS becoming unavailable. Autopilot systems are available to fly during the transition. For the purposes of our networking architecture and when comparing against related work, we can consider the UAVs to be (migrating) hosts, the local GCS to be switches and the command and control centre to be the network controller.

III. Architecture

Traditional function-specific middleboxes and in-network devices such as, e.g., firewalls, are placed on the traffic path between the source and destination. In our approach, we extend and configure the GCS to become a UAS VNF server. Our UAS VNF architecture, developed and adapted from our early design [11], meets the following objectives:

a) High-mobility lightweight deployments: Deployment of NFs is simple, transparent, and fast for the subscribing hosts, taking <250ms to start a NF and redirect the traffic through it. Simplicity arises from the lack of a provisioning cycle either in the lead time to acquire the appliance or to setup a new server and associated routing rules.

b) Distributed processing for lower utilisation and latency: Moving programmable, adaptable, modular processing from the command centres to the GCS reduces the traffic on the more expensive and often strained backbone links from the remote GCS to the command centres. Streamed telemetry no longer needs to traverse these links, and processing NFs are lightweight to move and more infrequent. Latency of anomaly detection is also reduced. By placing detection nearer the UAV, anomalies are detected 'locally', without the need to route traffic over high latency links.

c) Increased situation-awareness: Operators can deploy context-specific NFs on-demand to better understand operational challenges and to inform their decision-making, e.g., deploying a granular connectivity monitoring NF to observe more detail on an intermittent loss link fault.

d) Infrastructure independence: Traffic routing is handled independently from default routing policies, allowing forwarding of traffic from hosts to ephemeral NFs in OpenFlow (OF) enabled environments. Decoupling default routing from policy enforcement routing reduces the risk of misconfiguration of the individual network elements.

e) Open Innovation: By using Linux-based containers, NFs can utilise the existing wealth of tools and programs available for native Linux, without having to adapt these to work in a bespoke environment. Containers are more lightweight than VMs, ensuring migration is quick and highly mobile.

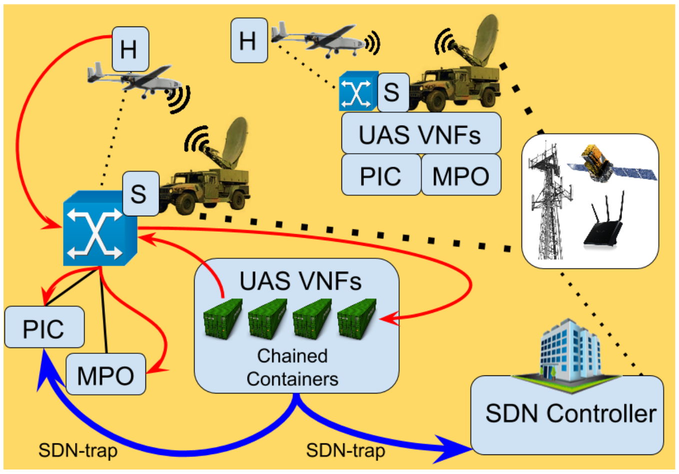

Fig. 2. UAS VNF architecture designed for high-mobility UAV-specific NFV

Figure 2 shows our UAS VNF architecture as deployed in the field. The UAVs are Hosts with Open vSwitch instances located at the mobile GCS vehicles. These switches route traffic from the host UAV to the Pilot in Command (PIC) and Mission Payload Operator (MPO) displays. The switches have OF rules to route traffic based on particular rules to the configured NF. With no NFs configured, telemetry from the UAV goes via the GCS switch to the PIC and MPO directly. If multiple UAVs of different types are operating from a single GCS, the OF tables can be configured to route traffic from each UAV to a different set of NFs designed for the operating parameters of that UAV type. Similarly, if NFs exist which are common to both types of UAV, traffic can be routed to the same NFs. The chained containers hosting the VNFs are situated at the GCS with the SDN controller located at the central command centre where it can be logically centralised and physically distributed for resilient oversight of the architecture. Considering the PIC and MPO are also Hosts, the architecture can be configured to route traffic from the GCS to the UAV via a set of NFs. For example, access control or further security measures for sensitive environments for protection against, e.g., replay or (D)DoS attacks via firewalls and rate limiters.

Chaining containers allows for smaller modular functionality to act independently. For example, a NF can run on a relatively inexperienced pilot's GCS to monitor the number of commands sent. If this NF flagged a series of anomalies, the wider GCS team or central command centre could deploy another NF configured to watch for anomalous pilot command sequences through, e.g., frequent repetition of a set or individual commands. This could help diagnosis if the UAV was being unresponsive or if human factors such as anxiety were involved. The ability to monitor and detect all issues simultaneously is infeasible due to the processing and storage capabilities available, especially in mobile remote environments. By chaining NFs and allowing for real-time updates, the processing and hardware available can be utilised to host a vast array of context and UAV-specific network functions which can inform and alert operators.

Figure 2 shows 'SDN-traps' which, similar to SNMP-traps, act as monitoring notifications for traditional data network operators. Our UAS VNF architecture allows NFs to be configured to send SDN-traps to both the PIC and SDN-Controller. For example, if a fuel monitoring NF with a SDN-trap configured for when fuel $< 15\%$, the mission-control team for a multi-UAV-GCS operation can see where coverage will be lost and move UAV capability as required to ensure mission objectives are met.

A. Routing

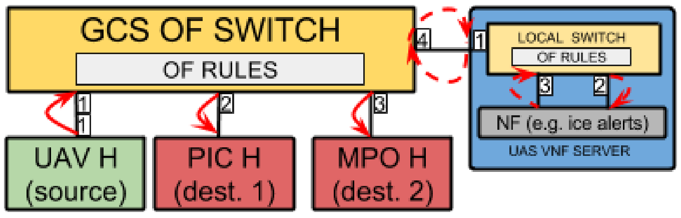

Fig. 3. UAS VNF OpenFlow routing architecture

Switch	Match	Action
GCS	in_port: 2, src_ip: UAV1	out_port: 4
Local	in_port: 1, src_ip: GCS	out_port: 2
Local	in_port: 3, src_ip: NF1	out_port: 4
GCS	in_port: 4, src_ip: local_switch	out_port: 2,3

TABLE I. OPENFLOW TABLE ENTRIES FOR UAS VNF MANAGEMENT

Figure 3 details the routing design in the GCS Open vSwitch. The GCS switch is connected to the UAV host on port 1. Other UAVs can connect to new ports. PIC and MPO are connected on ports 2 and 3 respectively, a single laptop environment would require only one port for both roles. The UAS VNF server connects on port 4, with another local switch routing traffic through the containerised VNFs. In this case, GCS source traffic is forwarded from port 1 to 3, and traffic egressing the NF is sent back to local switch port 3 and on to GCS port 4. Incoming traffic on port 4 is mirrored and sent to both the PIC or MPO displays, if applicable. This routing design allows for additional NFs to be deployed without interfering with the GCS switch routing for the UAVs. Table I shows the OF routing table entries for this setup.

B. Implementation

In a multi-display GCS set up, the current GCS switching capabilities which route traffic to the PIC and MPO processors would be replaced with an Open vSwitch. In a single laptop controller GCS environment, the OF switching can take place in situ with the laptop acting as both a switch and a host. Our UAS VNF server comprises lightweight Linux-based chained containers, each of which performs a virtual NF before routing the traffic on to the next process, reporting any alerts to the PIC or centralised controller as configured. The architecture is built using the Python SDN-Controller, RYU [15]. This lightweight controller is component-based with pre-defined components which can be modified and extended to create a customised controller application allowing for easy programmability of both the north and southbound SDN interfaces.

Our UAS VNF architecture uses Linux containers that provide a lightweight equivalent to VMs, allowing each container to use the host OS kernel to isolate processes, network routing tables, and their associated resources. This approach does not require each isolated function to run on a separate OS image, hence allowing a much higher network function-to-host density and smaller overall footprint. Using containers, commodity compute devices, such as, e.g., laptops, are now able to host up to hundreds of NFs. The minimal cost of starting and stopping containers as well as the single package encapsulation allows for NFs to roam alongside the UAV. As the UAV roams, the associated NFs can be started on a different GCS and traffic rerouted to it through modifying the corresponding OF rules.

C. Migration

When a UAV moves between GCSs, the operators at the SDN-controller can manually migrate the required NFs by clicking on the web app user interface, deploying the required NFs on the new GCS UAS VNF server. Operators at the GCS can also place requests for NFs. Automated migration is also possible if the entire environment is defined with static IP-addresses assigned to each of the UAVs in the operating environment. To achieve this, NFs are associated with individual UAVs. When the GCS OF switches receive packets from a new IP address (a new UAV which has migrated to this GCS), the initial packets are sent to the SDN-controller. The controller then follows the standard SDN paradigm by receiving these forwarded packets and replying with the OF rules and NFs to install based on the allocation configurations for the particular UAV. This network configuration and NF migration incurs no delay to the overall current ConOps migration process. NFs can migrate between GCSs (<250ms) well within the time it takes for secure handshakes to establish control between the UAV and new GCS (>1 second).

IV. Evaluation: SIERRA Case Study

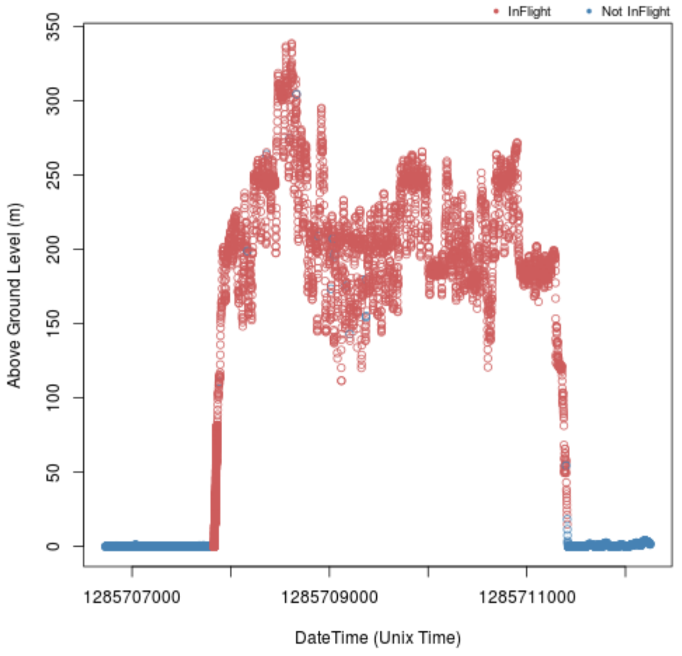

Fig. 4. SIERRA flight classified by *InFlight* for metres Above Ground Level (AGL) over time

To evaluate our UAS VNF architecture, we created a suite of NFs to assist operators with issues commonly cited by subject experts: NASA UAV pilots. Our initial efforts focused on the issue of UAVs saturating operators with alerts of current operating conditions. An example scenario would be a UAV with fuel reserves for 10 hours of flight, and a warning notification built into the aircraft hardware to notify the pilot every minute when fuel levels are below a threshold, e.g., $<$ 10%. Under planned or emergency circumstances where these warnings would come into effect, it is likely this notification frequency would be an unhelpful distraction to pilots. To mitigate this, we generated Python templates which aggregate such notifications ensuring that, when under special conditions which may demand pushing the UAV beyond normal operating thresholds, the pilot will not be adversely distracted. Our NF scripts allow for the setting of new thresholds in software, which are easily programmable and adaptable during live deployment, unlike those set in the UAV hardware sensor systems. The next set of NFs we developed were more specifically tailored to UAV types. The telemetry of all UAVs have multivariate inter-dependencies, with physics underlying the models of many of these such as the relationship between altitude and pressure, altitude and fuel burn rates, and outside air temperature and internal temperature readings. These models are an excellent definition of normal operating behaviour which can be used to rapidly detect unexpected, anomalous behaviour. While operators receive real-time readings of many of these values, it is often in the subtle emerging trends where problems can first be observed. To show the ease of which our architecture can be tailored to different UAV types, we used telemetry from the unmanned Sensor Integrated Environmental Remote Research Aircraft's (SIERRA) historic flights (80+ flights worldwide over a number of years) to

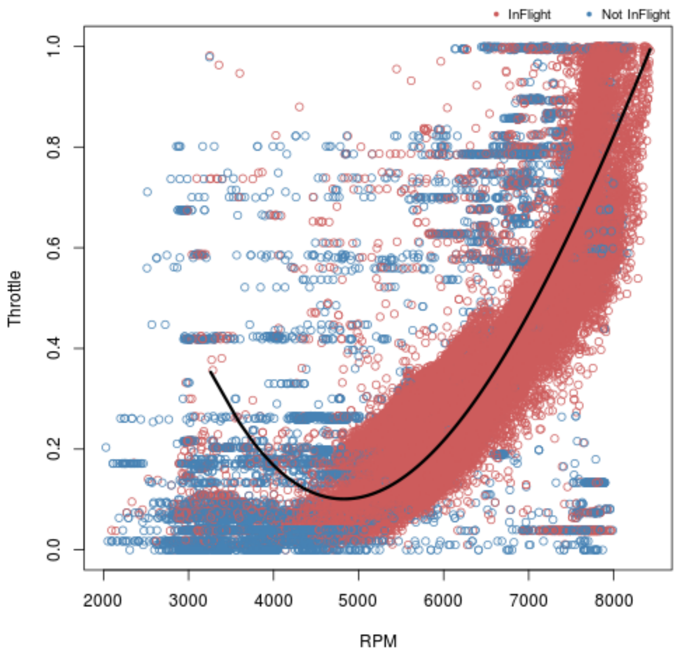

Fig. 5. SIERRA aggregate classified flight history for Throttle vs RPM

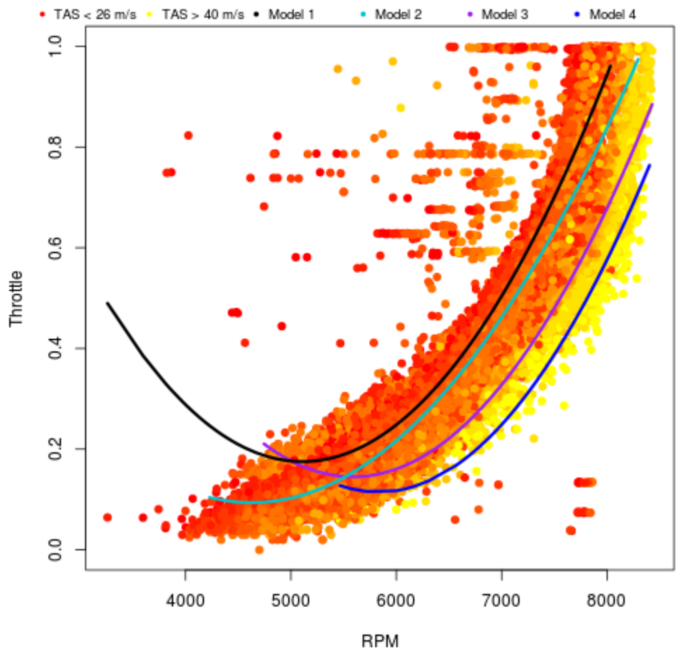

Fig. 6. SIERRA aggregate *InFlight* data with polynomial regression models

tune the initial models in our NF suite based on the normal operating parameters observed. Telemetry relationships vary heavily across the different flight phases: takeoff, inflight and landing. We began by developing a simplified classification with input from our domain expert, to determine *InFlight* status. We determined the simplified *InFlight* classification as:

- $True\ Airspeed > 26\ m/s$
- $Throttle\ Acceleration \neq 0$
- $Revolutions\ Per\ Minute\ (RPM)\ > 2000$

Figure 4 shows the application of our classifier to a flight recording from the dataset. The graph shows the flight profile with points classified as *InFlight* (red) and other flight phases (blue). The model is very successful with near-perfect accuracy for this flight. This is seen through the flight profile, with the vast majority of *InFlight* (red) points with AGL > 0 and grounded, takeoff and landing phases coloured blue. Figure 5 shows this same classification applied to the aggregation of all historic SIERRA flights, with the polynomial regression model for throttle against RPM for all *InFlight* data in black. The classification removes much of the noise seen from takeoff and landing phases. There is a great deal of variance from the model, which mitigates the successful detection of abnormal behaviour. Figure 6 shows only the *InFlight* data with the True Airspeed (TAS) coloured on a red to yellow spectrum from 26-40+ m/s, respectively. Four polynomial models are also shown, highlighting the variance in Throttle to RPM relationship as TAS increases. This tightens our overall model, allowing for a better understanding of any emerging deviance among these three parameters behaviours against the norm. Should one of these parameters cause data points to lie nearer a different polynomial model than expected, this is likely the beginning of an electromechanical or sensor failure. Along with other dependencies, we took these pre-conditions for SIERRA *InFlight* and our polynomial regression models with the appropriate confidence intervals and tuned our detection NF in Python for deviations from the norm.

On July 26, 2013 the SIERRA lost engine power 4.5 hours into its 6 hour scheduled flight and crashed into the Beaufort Sea, 65 nautical miles north of Oliktok Point, Alaska, where the controlling GCS was located. The UAV was undertaking a sea ice survey for the Marginal Ice Zone Observations and Processes EXperiment (MIZOPEX) project. The MIZOPEX reconnaissance mission involved multiple UAVs of different types. The NASA mishap accident investigation report [14] found that the only indications to the SIERRA team of the impending crash were the A/C engine's revolutions per minute reading of zero RPM and the electrical bus voltage at 4V lower. The lower voltage confirmed the engine had stopped turning. At this time, the report states no pilot instructions could have avoided the loss of the UAV. However, on further analysis of the telemetry prior to the engine failure, investigators discovered the throttle demand increased by over 40% and continued to rise, while the engine struggled to maintain its cruising RPM of 6,000 as much as one hour prior to the crash. With 6,000 RPM, 0.15 throttle and consistent TAS, a 40% increase in Throttle and steady RPM, would show telemetry points transitioning over polynomial models 4 through 2 over time. This information was not displayed to the pilot through the real-time GCS information. The RPM also plummeted at times to anomalous lows of 4,000, only seen on Model 1 without very few observed *InFlight* data points, and the engine behaviour was described as sporadic. Had the team been notified of these anomalies and returned SIERRA to base, the report concludes the mishap could have been prevented.

This incident highlights the need for greater monitoring and anomaly detection systems. Prior to the SIERRA crash and following the 40% increase in throttle, there were eight *ice warning* alerts in a 25 minute window, a significant increase in frequency. In discussions with domain experts, it was clear this sensor had a significant safety margin and for operations in cold environments had to be ignored to some degree. To evaluate our system, we replayed the SIERRA crash flight data in our simulated architecture ConOps with our suite of SIERRA calibrated NFs. On replaying the crash flight telemetry in real-time, our UAS VNF architecture detected the increased frequency in *ice warning* alerts, issuing an SDN-trap notification to both the SDN-controller and PIC. The deviation in throttle was also detected shortly after the $>40\%$ increase without a corresponding increase in RPM, a transition across models 4 through 2 in Figure 6, also throwing numerous SDN-trap alerts to both the PIC and the SDN-controller. The results of this case study show our UAS VNF architecture empowers UAS operators to deploy and adapt the monitoring and detection functionality in live operations, also helping to adapt and rectify the sensor sensitivity of built-in alerting systems of various UAV types. Using our modular suite of NFs tuned to the SIERRA in our replay experiment, we successfully detected real-world anomalous behaviour as the transitioning across three of the UAV-tuned models in Figure 6 with consistent TAS. From this case study, we have a set of models for use in a range of deployments which can be tailored to specific contexts and distinct UAVs with historic flight data.

V. Related Work

In recent years, the state-of-the-art in networking has centred on two complementary, yet distinct, concepts: SDN and NFV. SDN [3] is a network architecture which allows for abstraction and virtualisation through the decoupling of the network's control and data planes. OpenFlow [4] is the first and most widely used SDN implementation NFV [2] is a transformation in the delivery of network functions from bespoke, specialised hardware, to network functions in software which can run on a range of commodity hardware which can be migrated to, or instantiated at, various locations within the network topology, on demand. This paper builds on our prior SDN+NFV work designed for efficient enterprise networks, Glasgow Network Functions (GNF) [1], tailoring it to meet the needs of UAS ConOps with a suite of specific NFs, modified migration and routing configuration.

Khalastchi et al. present the case for anomaly detection in unmanned vehicles [5], arguing for computationally lightweight systems to avoid additional computational load on the vehicle, potentially introducing more faults. However, all functionality is located on the UAV. Having NFs at GCSs allows for more rapid NF migration and deployment. Increasing the processing and complexity of different UAVs will increase vendor lock-in and reduce the openness required for wide-scale integration.

Other network edge services (e.g. OpenEdge [7]) and NFV frameworks (e.g., ClickOS [6] or FlowOS [9]) rely on bespoke platforms, hypervisors, or commodity x86 servers with resource-hungry VMs preventing their use in wide-area deployments where high NF density and mobility is paramount. These bespoke architectures also prevent widely-used applications and tools being deployed without modification. In the UAS context, it is therefore preferable to use a container-based Linux architecture where lightweight VNFs can run native Linux tools and be migrated on demand.

VI. Conclusions & Future Work

In this paper, we have presented a novel UAS VNF architecture which, through a suite of lightweight, container-based monitoring and detection NFs statistically tuned to specific UAV types, enhances the situation-awareness and the highly demanding mobility requirements of the overall UAS environment. Through exploiting state-of-the-art SDN and NFV principles, our platform and UAV independent architecture gives mission controllers the opportunity to adapt their live telemetry monitoring and anomaly detection capabilities on demand. Situation-awareness is increased with the ability to program the network to provide a better understanding of emerging challenges in times of need and further contribute to the overall resilience domain. High mobility UAV and GCS deployments are supported through rapid migration of modules and a distributed approach placing demands on the strongest, most resilient links in the wider network infrastructure. Linux-based containers implemented within our architecture ensure open innovation with reuse of existing libraries without modification and reduced utilisation on traditionally strained network infrastructures in low density regions. We have demonstrated the applicability of our architecture through an evaluation and case study where our generic detection models, tuned to the SIERRA, detected the pre-conditions as much as 1 hour prior to its crash under replay conditions. Future work will focus on tuning our NF suite to further UAV types, such as the Viking 400, and developing as well as deploying a real-world test-bed.

Acknowledgments

The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, and EP/L005255/1.

References

[1] R. Cziva, S. Jouet, K. J. S. White and D. P. Pezaros, *Container-based network function virtualization for software-defined networks*, IEEE Symposium on Computers and Communication, Larnaca, 2015.

[2] E. T. S. Institute. 2012 Network Functions Virtualisation, White Paper.

[3] Open Networking Foundation. SDN Architecture. Tech. rep. Feb. 2016.

[4] N. McKeown et al. *OpenFlow: enabling innovation in campus networks*. In ACM SIGCOMM Computer Communication Review 2008.

[5] E. Khalastchi et al. *Online anomaly detection in unmanned vehicles*. 2011 Int. Conference on Autonomous Agents and Multiagent Systems.

[6] J. Martins et al. *ClickOS and the art of NFV*. In USENIX NSDI, 2014.

[7] J. Kunz et al. *Openedge: A dynamic and secure open service edge network*. In IEEE/IFIP NOMS, 2016.

[8] FAA. Aerospace Forecast Fiscal Years 2016-2036.

[9] M. Bezahaf, A. Abdul, and M. Laurent. *Flowos: A flow-based platform for middleboxes*. In Proceedings of Hot Topics in Middleboxes and Network Function Virtualization, 2013, ACM.

[10] P. Royo, J. Lopez, E. Pastor, C. Barrado. *Service abstraction layer for UAV flexible application development*. In 46th AIAA 2008.

[11] K. J. S. White, D. P. Pezaros, and C. W. Johnson. *Principles for increased resilience in critical networked infrastructures*. ICRAT 2014.

[12] K. Williams, *A summary of unmanned aircraft accident/incident data: Human factors implications*. FAA, 2004.

[13] A. Tvaryanas, W. Thompson. *HFACS analysis of 221 mishaps over 10 years*. Aviation, space, and environmental medicine, 2006.

[14] NASA, Ames Research Center. *SIERRA Mishap Classification* 2013.

[15] RYU SDN Controller. https://osrg.github.io/ryu/. Accessed: June 2016.

[16] E. Pastor et al. *In-flight contingency management for unmanned aerial vehicles*. Journal of Aerospace Computing and Communication 9, 2012.