Monitoring For Time Delay

William Willie Wells

***Abstract*—Time delay in a control system with an expected control rate may have detrimental effects on the controlled system. A ROS (robot operating system) node that monitors for time lapse between control node updates in mit_asctec, a system that derives a control input for autonomous flight of Ascending Technologies vehicles, was created. The time between updates of seven topics essential to the final vehicle control input are monitored. If the time lapse exceeds the expected time lapse, a hover message is published and normal mission execution is paused. One of three actions (normal execution is resumed, the vehicle lands then resumes, or the vehicle lands, shuts down, and the mission is restarted at the beginning) is then taken after waiting at least one status update period. The monitor node runs at a higher control rate than the mission control nodes. This minimizes packet delays associated with the processing of this node.**

I. Introduction

A. Motivation

Undesired time delays may be introduced into a system by unanticipated operating system interrupts, race conditions between two or more rate dependent processes, failed checksums, or out of order execution. In an out of order execution scenario, a task may be executed within its rate window but the time lapse between any two consecutive executions of the task may vary from the time it takes to execute the task to nearly twice the task period. The presence of such a varied time lapse results in a shorter or longer than expected controlled response (e. g. a controlled vehicle remaining at a position for a fourth of the expected time at that position before moving on to the next trajectory point). This behavior in extreme cases will result in an unmanned vehicle crashing and in general can cause a system to exceeds its bounds. Non routine system time lapses in a controlled system can be monitored in real time. Adequate bounds or limits can be set prior to performing a safety or protective critical action that places the system in a safe condition. This assumes that the monitoring entity can enact a system response before the system moves from a warning or potential damage condition to a fault or damaged condition.

B. Background

Time delay will be defined as any time lapse between events. Events within the monitored system are published messages that include time stamps relative to the time the messages were published.

One type of expected system time delay is inherent time delay of the system. This is the travel times associated with transferring data from one component in the system to another component in the system. An extreme example of this is communication between an object in space and an object on Earth. Another type of expected system time delay is time delay enforced by the system. Processes that execute at a set rate are in this category. A system that requires a wait period before performing one or more actions is a system with this type of delay.

Time delays that do not fit into the above categories are considered undesired time delays. If a significant time delay is detected, a protective action will be taken.

II. Related Work

Youcef-Toumi and Ito designed a time delay controller [19], which performs poorly for real systems with variations in packet delay.

Verdoy investigated packet delay variation in relation to internet quality of service [17]. Joe et. al. designed a jitter buffer control algorithm for quality of service in voice over internet protocol [8]. Yashiro et. al. designed a traffic shaping algorithm for jitter buffers [18]. Unlike the following work, these works focus on application domains that do not involve controlling a networked system.

Raptis et. al. designed a simulation of packet delay analysis for wireless networks [12]. Andre et. al. discusses the effects of jitter in networks of micro aerial vehicles [3]. Zhang et. al. presents a double disturbance observer to deal with time delay, external disturbance, and measurement noise in a wireless motion control system [20].

Several individuals have recently acknowledged the necessity of utilizing cyber physical system (CPS) approaches to analyze and model network controlled system [11], [16], and [9].

Srinivasan discusses the need for adaptive control in network controlled systems (NCS) due to the presence of packet delay variation [15]. Effects of jitter in a network controlled system based on cellular and IP networks was studied by [1], [21]. Jitter as a significant factor in network controlled systems is discussed by [14], [4], [5], [10], [7], [2], and [6].

III. Method

A. Test Setup

Fig. 1. Herbie–Ascending Technologies Hummingbird with ARDrone propellers connected to switching DC power supply after manual flight test.

A switching DC power supply is used to enable repetitive testing without the need to replace and exhaust the applicable unmanned aerial vehicle's supply of batteries. This power supply is switched on. A manual flight of the vehicle to be tested is conducted. The power supply is connected to the vehicle under test and the vehicle is turned on. Figure 2.1 shows the physical test configuration after the completion of these steps.

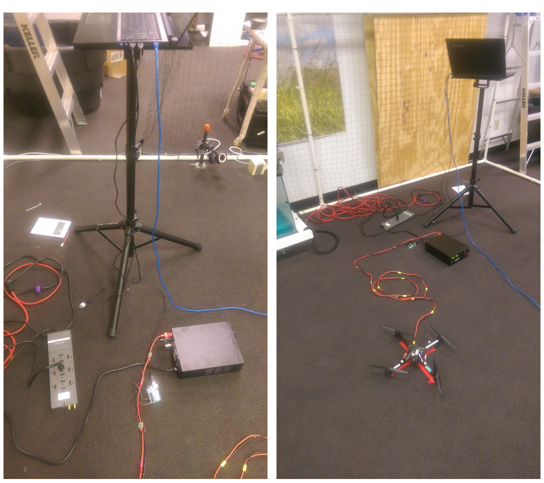

Fig. 2. Physical test setup.

Required software, Vicon Tracker and Robot Operating System (ROS), is started. The controlling station is connected to the power supply to enable receipt of sensor values. Communication, transmission and receiving of data packets, between vehicle under test and controlling station is established. Figure 2.2 shows the physical test configuration after the completion of these steps.

Testing software is launched. The vehicle under test is monitored as it performs predefined maneuvers. After the completion of the current test, the next test is prepped. Figure 2.3 shows four snapshots of the test procedure in progress as the unmanned aerial vehicle performs the predefined maneuvers.

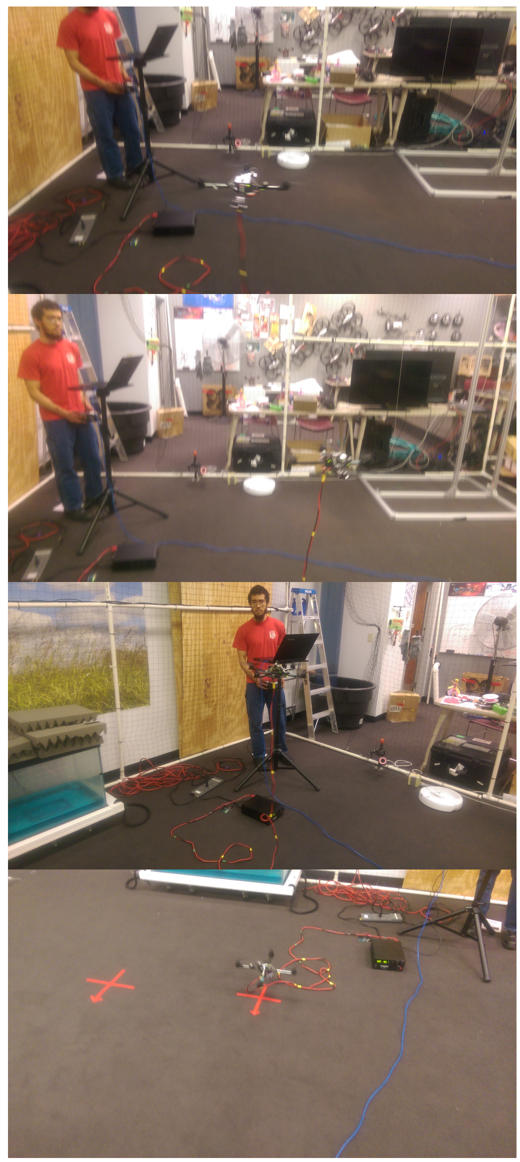

Fig. 3. Herbie–Ascending Technologies Hummingbird with ARDrone propellers progressing through the maneuvers of the flight test.

Testing setup may include start delays based on being able to connect to Vicon and other device connection issues. The test procedure itself is streamlined for efficient repetition.

B. Maneuvers

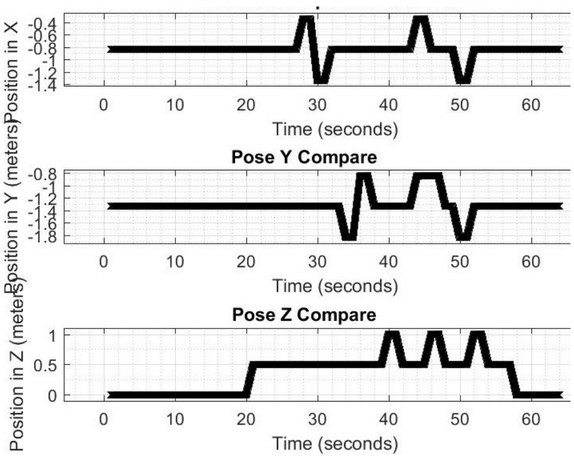

Fig. 4. Ideal task path in x, y, and z Cartesian axes with respect to time.

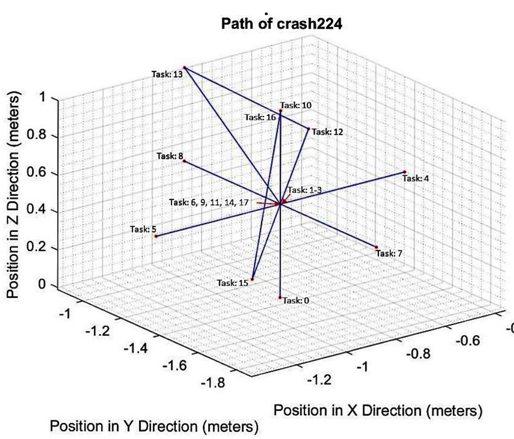

Fig. 5. A 3 dimensional plot of the ideal task path without time.

Flight testing launch file is launched. Launch height and a uniform magnitude for the base translation value (δ) can be specified in the launch file. The test maneuvers themselves are as follows. Turn motors on; test operation of motors by performing the maneuvers: pitch forward, pitch backward, roll left, roll right; launch; perform this rotation sequence while maintaining position $\pi \rightarrow \pi/2 \rightarrow 3\pi/2 \rightarrow 0(2\pi)$; translate along each Cartesian axis in 3 dimensional space with and without rotations; translate to positions that require change in multiple Cartesian axes coordinates with and without rotations; land; and turn motors off. The turning on of the motors is essential for the rest of the maneuvers and is the first visible state transition. Pitch and roll tests indicate vehicle orientation as well as indicating individual motors are operating properly. Launching enables testing translations and rotations while in flight. While on the ground and during launch the vehicle is initially rotated near π. A 90 degree counter clockwise (clockwise) rotation is equivalent to a 270 degree clockwise (counter clockwise) rotation. The selected rotation tests rotate the vehicle so that the vehicle faces positive y (North), negative x, positive x (East), and negative y this demonstrations the vehicle's ability to rotate to these facings while maintaining the same translation. The 180 degree rotation demonstrates the vehicle's ability to handle a maximum rotation as well as the quality of the gyroscope measuring yaw. Translations along each Cartesian axis demonstrate the vehicles ability to translate to specific positions in a Cartesian space. Simultaneous rotation and translation commands indicate the vehicle's ability to handle changes in both rotations and translations while implementing a single tasked waypose. Each pair of Cartesian translations (x and y, y and z, z and x) are tested once with a 90 degree rotation and once without a rotation. The last two flight maneuvers are a change in translation for all Cartesian axes with and then without rotation. The final translation and rotation is designed to be the same position and facing as the target launch position and facing. While maintaining this position and facing the vehicle is commanded to land. Landing is a prerequisite to turning off the motors. Turning off the motors places the vehicle in a safe condition prior to starting the next test.

The maneuvers themselves are designed to test vehicle response to common maneuvers. The pre-flight sequence is split into three stages: start up of the motors, testing of the motors, and launching of the vehicle. The pre-flight test of the motors is the only place in the procedure that tests maximum pitch and roll. An initial position in the horizontal plane is set once during the first state change. All subsequent changes are based on this, a rotation value of 0.0, and a launch height, which is set in the launch file. A counter keeps track of the target flight maneuver. Each flight maneuver has a move to position then hover component. After witnessing and being required to use this move implementation style in a previous project, it was deemed a better, stabler, implementation than a continuous move implementation style. When the maneuver counter exceeds a certain value a land request is set and the landing sequence begins.

C. Logs

During the test any abnormal behavior is annotated in a testing response form using Google Sheets. A safety operator ready to take manual control is always attentive during each flight. The output ROS bag file is updated. Bag files are in the format vehicleType_vehicleNameTestNumber.bag. Vehicle is reset to start origin, launch file is started again, and the above maneuvers are repeated.

At the end of the day's testing or when ever convenient thereafter the bag files are converted to .csv files using bag2csv [1]. The .csv files are then imported into MATLAB for analysis. An updated version bag2matlab [2] was created by Dave Anthony in June of 2016. This version is the version used for data analysis after the indicated date.

The flight testing node is shutdown after an electrically stopped state is reached. The rest of the nodes started by the cascaded series of launch files are shutdown as well. Automatic shutdown of running nodes provides a reliable way to extract a measure of test execution time. Without this measure in place topics may be recorded continuously beyond the actual length of the test. Launch file cascade structure used involves three layers. The innermost layer contains the nodes related to communication, the middle layer contains nodes related to basic position control, and the top layer contains the nodes specific to the flight testing project and specific to that particular test launch. Vehicle under test, vehicle type, vehicle type specific middle layer launch file, and recorded bag file name are the commonly modified fields in the top layer launch file. A subset of the available topics to record is recorded to reduce bag file size. The topics recorded are: target state, quad and robot control input, robot inertial measurement unit, robot and subject status, voltage and current sensor messages, actual state, subject position, target position, and Vicon position.

D. Control Packet Delay

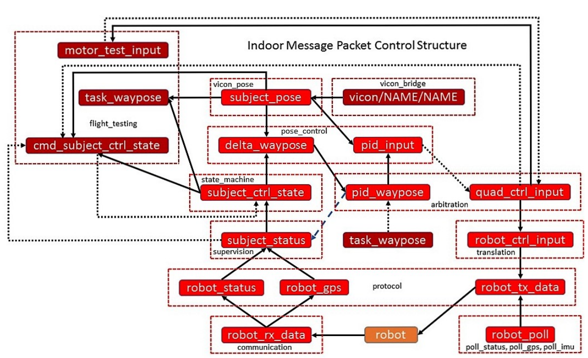

Fig. 6. Control message flow loop.

[1]https://github.com/unl-nimbus-lab
[2]https://github.com/unl-nimbus-lab/bag2matlab

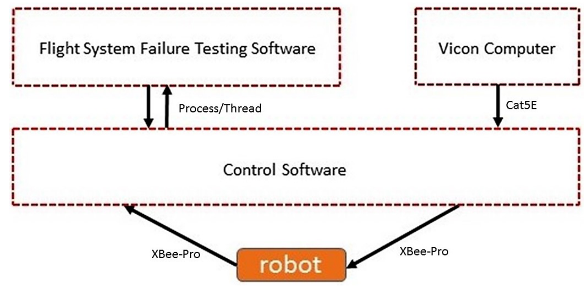

Fig. 7. A simplified control flow diagram.

Packet delay variance of sensor request and control input packets in autonomous systems result in observable abnormal behavior that is not intended by the designer of the control software. Using 6 vehicles 923 test flights with an execution time of 65 seconds were performed. There were 125 tests, which exhibited observable abnormal behavior that was due to packet delay variation. The topic robot poll is the only topic directly published to by multiple nodes (poll gps requested at 4 Hz, poll imu requested at 15 Hz, and poll status requested at 1 Hz). Robot poll messages are polled at 1 Hz, 4 Hz, and 15 Hz. Robot control input messages are updated at 30 Hz.

A diagram showing the control flow of data from the unmanned aerial vehicle and back is illustrated in Figure 2.10. Solid lines from topic A to topic B indicate topic A is a publishing dependency for topic B. A dashed line from topic A to topic B indicates that topic A is subscribed to but not required for publishing of topic B. A dotted line from topic A to topic B indicates that topic A provides data that results in the published data of topic B changing or that topic A is one of several state dependent message sources for topic B in the case of the arbitration node. Nodes are identified by dashed boxes and their node names in the figure. Figure 2.11 is a simplified version of Figure 2.10 and contains the communication medium used between the separate parts of the control system.

E. Observation

A node that uses a ROS timer with a time interval of 5 ms monitors for packet delay variation in subject pose, tasked waypose, status, state, command state, quad control input, and robot poll. The update intervals of control input and vehicle sensor request can be defined in a launch file along with a safe_time parameter that defines the amount of maximum allowable additional delay between sensor request updates. If the vehicle is in flight, the processing time of the previous packet update is subtracted from the processing time and compared to either the sum of the safe_time and the sensor request interval or the safe_time depending on the update rate of the monitored packet type (message). If the former exceeds the latter, a flag is set, a warning is printed to stdout, and a message is published on a topic named jitter. Then the processing time of the previous packet update is set as the processing time.

IV. Experiments

The monitor node described above was used in 30 test flights. Some of the 30 tests contained packet delay variations noticeable in post-processing but not in excess of the safe_time parameter value of 200 ms. After these tests were conducted, a simulation using the data from the 125 flights with packet time variation was ran in MATLAB.

V. Discussion

A node in ROS was created to monitor for packet time variation in control software. A simulation using vehicle flight logs to test vehicle response in the presence of packet delay variation after commanding a UAV to hover was created.

VI. Future Work

Future endeavors include implementing a reactive response after excessive packet delay variation is received with a UAV not just a simulation.

VII. Conclusion

Packet delay variance of sensor request and control input packets in autonomous systems results in observable abnormal behavior that is not intended by the designer of the control software.

References

[1] Afonso, L., Souto, N., Sebastiao, P., Ribeiro, M., Tavares, T., Marinheiro, R. (2016). Cellular for the skies: Exploiting mobile network infrastructure for low altitude air-to-ground communications. IEEE Aerospace and Electronic Systems Magazine, 31(8), 4-11.

[2] Ahmed, S., Hasan, M., Subhan, F. (2014, September). Filtering for networked controlled systems. In Computer, Communications, and Control Technology (I4CT), 2014 International Conference on (pp. 101-104). IEEE.

[3] Andre, T., Hummel, K. A., Schoellig, A. P., Yanmaz, E., Asadpour, M., Bettstetter, C., Zhang, S. (2014). Application-driven design of aerial communication networks. IEEE Communications Magazine, 52(5), 129-137.

[4] Bouazzawa, K. E. (2014). Dynamic output feedback for nonlinear networked control system with system delays and packet dropout. IFAC Proceedings Volumes, 47(3), 6496-6501.

[5] Dong, J., Kim, W. J. (2012). Markov-chain-based output feedback control for stabilization of networked control systems with random time delays and packet losses. International Journal of Control, Automation and Systems, 10(5), 1013-1022.

[6] Hajebi, P., AlModarresi, S. M. T. (2013, January). Online adaptive fuzzy logic controller using genetic algorithm and neural network for networked control systems. In Advanced Communication Technology (ICACT), 2013 15th International Conference on (pp. 88-98). IEEE.

[7] Hamdy, M., Abd-Elhaleem, S., Fkirin, M. A. (2016). Time-Varying Delay Compensation for a Class of Nonlinear Control Systems Over Network via Adaptive Fuzzy Controller. IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[8] Joe, D. W., Shim, J. S., Ju, Y. W., Park, S. C. (2016, December). Design of Jitter Buffer Control Algorithm for Guaranteeing the Medical Information Data Transmission Quality in Wireless Network Environment. In International Conference on Computer Science and its Applications (pp. 279-284).

[9] Li, W., Zhang, X., Li, H. (2014). Co-simulation platforms for co-design of networked control systems: An overview. Control Engineering Practice, 23, 44-56.

[10] Mahmoud, M. S. (2016). Networked Control Systems Analysis and Design: An Overview. Arabian Journal for Science and Engineering, 41(3), 711-758.

[11] Rahnama, A., Xia, M., Antsaklis, P. J. (2016). A Passivity-Based Design for Stability and Robustness in Event-Triggered Networked Control Systems with Communication Delays, Signal Quantizations and Packet Dropouts. ISIS, 003.

[12] Raptis, P., Vitsas, V., Paparrizos, K. (2009). Packet delay metrics for IEEE 802.11 distributed coordination function. Mobile Networks and Applications, 14(6), 772-781.

[13] Seiler, P., Sengupta, R. (2001). Analysis of communication losses in vehicle control problems. In American Control Conference, 2001. Proceedings of the 2001 (Vol. 2, pp. 1491–1496 vol.2).

[14] Shi, Y., Huang, J., Yu, B. (2013). Robust tracking control of networked control systems: application to a networked DC motor. IEEE Transactions on Industrial Electronics, 60(12), 5864-5874.

[15] Srinivasan, S. (2014). Role of adaptation in networked control systems: Overview and research trends. Journal of Control Science and Engineering, 2, 52-60.

[16] Sztipanovits, J., Koutsoukos, X., Karsai, G., Kottenstette, N., Antsaklis, P., Gupta, V., Wang, S. (2012). Toward a science of cyber–physical system integration. Proceedings of the IEEE, 100(1), 29-44.

[17] Verdoy, H. M. (2009). Investigation of packet delay jitter metrics in face of loss and reordering (Doctoral dissertation, Blekinge Institute of Technology).

[18] Yashiro, D., Yakoh, T. (2014). Feedback controller with low-pass-filter-based delay regulation for networked control systems. IEEE Transactions on Industrial Electronics, 61(7), 3744-3752.

[19] Youcef-Toumi K, Ito O. A time-delay controller for systems with unknown dynamics. J Dynam Syst Measur Control-Trans ASME 1990;112:133–42.

[20] Zhang, W., Tomizuka, M., Wei, Y. H., Leng, Q., Han, S., Mok, A. K. (2015, July). Robust time delay compensation in a wireless motion control system with double disturbance observers. In 2015 American Control Conference (ACC) (pp. 5294-5299). IEEE.

[21] Zhu, L., Yin, D., Yang, J., Shen, L. (2015, August). Research of remote measurement and control technology of UAV based on mobile communication networks. In Information and Automation, 2015 IEEE International Conference on (pp. 2517-2522). IEEE.