Enabling Collaborative Behavior among CubeSats

Daniel C. Browne, Michael E. West
Georgia Tech Research Institute – Electronic Systems Lab
400 10th Street, Atlanta, GA 30332-0840; 404-407-7264
daniel.browne@gtri.gatech.edu

ABSTRACT

Future spacecraft missions are trending towards the use of distributed systems or fractionated spacecraft. Initiatives such as DARPA’s System F6 are encouraging the satellite community to explore smaller, lower cost, and more robust solutions to replace the conventional monoliths in LEO today. Enabling collaborative behaviors among teams or formations of pico-satellites requires technology development in several subsystem areas including attitude determination and control, orbit determination and maintenance capabilities, as well as a means to maintain accurate knowledge of team member’s state. This paper presents a collaborative module, designed with the CubeSat framework in mind, to provide autonomous on-board orbit determination as well as inter-satellite link capabilities for maintaining state knowledge and sharing sensor data among a formation. The end goal is to enable collaborative behaviors while reducing inter-satellite communication to realize significant power savings. Simulation results indicate an average 75% reduction in the amount of inter-satellite communication with some scenarios showing more than a 90% reduction. Furthermore, parallel implementations of the described algorithms indicate further power savings is achievable by using multicore microcontrollers with core throttling.

INTRODUCTION

Popularized separately over the past decade, combining technology advances in both CubeSats and formation flying among small satellites will assist in realizing the next generation of spacecraft systems. Universities and amateurs have and are taking advantage of the CubeSat and P-POD (Poly-Picosatellite Orbiter Deployer) specifications to reach space, but all complete mission successes have been single satellite systems. Recent endeavors including NASA’s Afternoon-Train (more commonly referred to as the A-Train) have begun pursuing formation flight for larger spacecraft. The A-Train is currently composed of four satellites in a 13:30 sun-synchronous orbit that all cross the equator within several minutes of each other. Two more satellites are planned to join the formation in the next few years. The two scheduled missions do not include the GLORY satellite recently lost due to a failure to reach its intended orbit on March 4, 2011.

One CubeSat swarming mission currently in the planning stages is QB50. This mission plans to launch fifty 2U CubeSats into very low altitude orbits to collect data on the ionosphere. Although QB50 will utilize a swarm of CubeSats, all correlation of the data will be conducted during post-mission ground analysis.

Last summer the Air Force Space Command released a solicitation for a Space Environment Nano-Satellite Experiment (SENSE). The Request for Proposal sought two 3U CubeSat systems fitted with instruments to collect data on GPS signal scintillations due to the ionosphere. Selection of complementary sensors could provide a desire for collaboration among the CubeSats.

Currently in orbit is the Swedish Space Corporation’s PRISMA mission composed on the Mango and Tango spacecraft. PRISMA has already demonstrated formation flying on the order of meters, inter-satellite communication, and demonstration of a new green propulsion system. Recently, mission operations were temporarily handed off to the German Aerospace Center (DLR) for additional technology demonstration.

These example missions and initiatives establish the growing arena of using formations for future spacecraft systems. Furthermore, by utilizing a proven platform such as the CubeSat, time from conception to launch can continue to decrease. This research set out with the goal of designing a collaborative module offering on-board orbit determination, an inter-satellite link framework, and an autonomous process of maintaining accurate formation state knowledge among the team. The design process includes both algorithm development and implementation and hardware component selection, hoping to build upon the previous work of others. Furthermore, the module design needs to reduce the power requirement for collaborative operations by limiting the amount of inter-satellite communication.

Using standard low-power microcontrollers allows for significant power savings over RF transmissions. For
every second of RF transmission with 1 W of output power the same amount of energy could power a microcontroller completing 4 million operations per second for 20 minutes. By using an extended Kalman filter (EKF) with a tuned force model and GPS measurements, estimations of both the state and model parameters are calculated. Using this EKF provides significant improvement in the length of time a model’s propagation is valid, reducing inter-satellite communication; simulations at the end of the paper demonstrate this advantage.

The remaining sections of this paper describe the process followed in designing the collaborative module. Details on the orbit propagation implementation, on-board orbit determination algorithm, and inter-satellite communications framework are provided. Finally, simulations combining the individual components demonstrate the modules ability to realize power savings. Parallel implementations of the algorithms indicate that using core throttling on a multicore architecture would provide further energy savings. The paper concludes with a summary of plans for future work related to this research effort.\textsuperscript{5}

**ORT Orbit Propagation**

The first step in software development for the collaborative module is selecting an orbit propagation implementation. A propagation technique, force model, and integration routine need to be selected for a complete implementation. The following sections discuss each of these trade spaces.

**Orbit “Truth”**

Before experimenting with the accuracy of various propagation techniques, force models, and integration routines, a valid “truth” dataset needs to be defined. For purposes of this research two different datasets are selected. The first is generated using a.i. solution’s FreeFlyer software package. FreeFlyer provides a propagation engine that includes high-order non-spherical effects, third bodies, drag, and solar radiation pressure (SRP). A reference 700 km, sun-synchronous orbit is defined for propagation using FreeFlyer; Table 1 lists the initial orbit elements with an epoch of midnight, January 1, 2011 GMT.

<table>
<thead>
<tr>
<th>a (km)</th>
<th>e</th>
<th>i (deg)</th>
<th>Ω (deg)</th>
<th>ω (deg)</th>
<th>ν (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7083.14</td>
<td>0.01</td>
<td>98.21</td>
<td>280.40</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The second “truth” dataset is available from NASA’s Jet Propulsion Lab’s HORIZONS online ephemeris web service. HORIZONS online interface provides ephemerides for solar system bodies as well as JPL mission satellites. These are reliable orbit determination solutions calculated on the ground with ample measurements. The Wide-field Infrared Survey Explorer (WISE) is one of the satellites with available ephemerides. WISE was launched on December 14, 2009 into a 525 km altitude sun-synchronous orbit with approximate orbital elements listed in Table 2. WISE’s primary mission was to survey the sky to identify the most luminous galaxies in the universe and finding stars closest to our own Sun. Using HORIZONS, ephemerides with one minute increments are available for the first several days of 2011.

**Table 2: JPL’s WISE spacecraft initial orbital elements with a midnight, January 1, 2011 epoch.**

<table>
<thead>
<tr>
<th>a (km)</th>
<th>e</th>
<th>i (deg)</th>
<th>Ω (deg)</th>
<th>ω (deg)</th>
<th>ν (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6901</td>
<td>0.001</td>
<td>97.5</td>
<td>7.2</td>
<td>39.5</td>
<td>94.0</td>
</tr>
</tbody>
</table>

With two valid sets of data for comparison, design of the on-board orbit propagation implementation can commence. First discussed are the available propagation techniques, followed by the design of the force model and integration routine.

**Propagation Technique**

Three different propagation techniques are available for tracking the state of satellites in formation. Integration of the inertial Cartesian state is the simplest and arguably most straightforward. Depending on the force model selection, most accelerations are easily described in the Earth-centered inertial frame (ECIF).\textsuperscript{4} Furthermore, if a GPS receiver is chosen as the on-board sensor for orbit determination then the measurements map directly to the states, simplifying the extended Kalman filter. A disadvantage, however, of Cartesian integration is the need for a reference frame transformation to determine the relative position of spacecraft in a formation.

Orbital elements are the most common way of expressing the size, shape, and orientation of an orbit as well as the current position of a spacecraft on the orbit path. Several different variations of the orbital elements exist, but a common set is composed of the semi-major axis, eccentricity, inclination, longitude of ascending node, argument of periapse, and true anomaly defined by Equation (1), respectively. \textsuperscript{2} For familiarization, Figure 1 visualizes this set of orbital elements.

\[
\begin{bmatrix} a \\ e \\ i \\ Ω \\ ω \\ ν \end{bmatrix}
\]  \hspace{1cm} (1)

The second propagation technique uses the Lagrange Planetary Equations (LPEs) for integrating the changes
in orbital elements over time due to perturbations. Any conservative force represented as a potential can be included in the LPE equations of motion providing similar accuracy and flexibility as straight Cartesian integration. LPE integration has its disadvantages as well. First, if a GPS receiver is selected for on-orbit determination, there is a nonlinear mapping between the available measurements and state being estimated. Furthermore, a frame transformation is necessary to determine the relative position of neighboring spacecraft.

**Figure 1:** Five of the six orbital element parameters are pictured for familiarization. The semi-major axis, $a$, defines the size and energy of the orbit.

The last propagation technique explored is direct integration of the relative state. Clohessy and Wiltshire, expanding on Hill’s original work, define the simplest scenario for two spacecraft in near-circular, two-body orbits. Using these simplifications actually allows for a closed-form solution, removing the need to integrate other spacecraft at all. Much work has been conducted to expand on the CW equations to include perturbation effects including drag, SRP, third-bodies, and Earth’s $J_2$; however, including these effects often results in losing the nice closed-form solutions. The primary disadvantage with using relative motion is the difficulty in collecting measurements for the EKF. Either range and range rate measurements between satellites or differencing GPS solutions is required to calculate an orbit solution.

Considering each methods advantages and disadvantages, Cartesian integration in the ECIF frame seems most appropriate. With the chosen propagation technique the force model can be explored in order to determine an appropriate fidelity.

**Force Model**

Four primary types of perturbations (beyond the primary two-body acceleration) are considered: non-spherical effects, third bodies, drag, and solar radiation pressure (SRP). Perturbations due to a non-spherical Earth are of the most interest and examined first. The standard spherical harmonic representation is selected for describing the non-sphericity of the Earth. Equation (2) provides the generic potential equation for calculating the gravitational potential.

$$U = \frac{\mu}{r} \sum_{l=0}^{\infty} \sum_{m=1}^{l} P_l [\sin(\phi_{gcsat})] \left( \frac{R_\oplus}{r} \right)^l C_{lm} + \cdots$$

Note that the $P[\ldots]$s in the equation are Legendre Polynomials. This formulation gives a direct equation for implementation into the force model, but for higher-order potentials it is well known that the companion recursive approach is faster. For brevity details on spherical harmonics are absent, but can be found in provided references.

For this research three non-spherical potentials are selected for experimentation. Sets of coefficients for the $C$ and $S$ matrices are available from data collected from missions such as NASA’s GRACE (Gravity Recovery and Climate Experiment). The simplest model includes just the Earth’s oblateness: $J_2$. This term captures the fact that the Earth is larger at the equator than at the poles, causing a torque on non-equatorial orbits. Additionally, 4x4 and 20x20 fields are tested. Note that for each field, the primary GRACE field is simply truncated to the necessary size. An example of the 20x20 field, with the two-body and $J_2$ potentials removed, is provided in Figure 2.

**Figure 2:** Example of Earth’s gravitational field strength with two-body and $J_2$ potentials removed.
Each of the three non-spherical potential models are compared against two-body propagation and the *FreeFlyer* reference orbit in Figure 3 and Figure 4 and against the WISE reference orbit in Figure 5. The first obvious observation is the significant reduction in error from choosing any of the three non-spherical models rather than just two-body motion. Similar results are obtained for the WISE comparison and are therefore omitted for brevity. When comparing to *FreeFlyer* the $J_2$ model actually depicts the least amount of error. This does not necessarily mean that the simplest non-spherical model is best, since no other perturbations are currently implemented in the model. If drag and SRP are also added, then the small differences in the non-spherical perturbations may result in improvement.

Lastly, recall that in order to get the coefficients for a smaller field than the original dataset, the matrices are simply truncated; this is standard practice, but results in a set of coefficients not tuned to the field size.

When examining the comparison with the WISE orbit, the three models are even closer than in *FreeFlyer*, with the 4x4 field resulting in slightly reduced error at each trough in the sinusoidal curve. The sinusoidal error indicates that a cyclical perturbation is missing from the model and that capturing it would greatly improve the models accuracy.

![Figure 3: Error from two-body, $J_2$, 4x4, and 20x20 non-spherical force models when compared to the *Free Flyer* reference.](image1)

![Figure 4: Error for $J_2$, 4x4, and 20x20 non-spherical force models when compared to *FreeFlyer*.](image2)

![Figure 5: Error for $J_2$, 4x4, and 20x20 non-spherical force models when compared to WISE’s reference.](image3)

![Figure 6: Estimate flop counts increase exponentially with model complexity, encouraging the use of the $J_2$ model.](image4)
Drag is the next perturbation to consider. Calculating drag on a spacecraft is identical to any other object in motion, but some care must be taken when choosing the parameters for Equation (3).

\[
\vec{a}_{\text{drag}} = \frac{1}{2} \frac{c_p A}{m} \rho v_{\text{rel}}^2 \vec{v}_{\text{rel}} \frac{\vec{v}_{\text{rel}}}{|\vec{v}_{\text{rel}}|}
\] (3)

The acceleration equation is composed of the inverse of the ballistic coefficient, the local atmospheric density, and the velocity of the spacecraft relative to the atmosphere. The ballistic coefficient is assumed constant but will be estimated by the orbit determination EKF. To determine density a common set of data which generates a piece-wise continuous curve, provided in Figure 7, is used.\(^3\)

![Figure 7: A simple piece-wise continuous function provides an estimate of density at LEO altitude.](image)

The relative velocity is not contained in the state vector, but is rather the inertial velocity minus the cross term, as described by Equation (4).\(^3\)

\[
\vec{v}_{\text{rel}} = \frac{d\vec{r}}{dt} - \vec{a}_{\text{Earth}} \times \vec{r}
\] (4)

Verification of the drag model is completed by monitoring the change in semi-major axis over the course of a year. A spacecraft in a near-circular orbit with drag is constantly experiencing a resistive force. Therefore, the total energy of the orbit is decreasing which should result in the semi-major axis decreasing. Figure 8 confirms this assumption since it shows that over the course of a year there is an approximate 15 km reduction in the semi-major axis.

SRP is the third perturbation to examine. Energy from the Sun while in LEO is not obtained without some physical disturbance. That same solar flux (also referred to as intensity or irradiance) that is beneficial to be as large as possible for power generation also applies a small perturbing force to the spacecraft.

![Figure 8: Adding drag to the force model accurately results in the continual decrease of the orbit’s semi-major axis.](image)

Although the solar irradiance fluctuates in both short- and long-period phases, such as sudden flares and the 11-year cycle respectively, it averages 1367 W/m\(^2\). For power subsystem estimation the product of this solar flux value, solar panel area, cosine of the incidence angle, and solar panel efficiency provides an estimate on the power generated when in sunlight. Likewise, the solar radiation force can be calculated from the product of the solar pressure (\(p_{\text{SR}}\)), spacecraft reflectivity (\(c_R\)), and spacecraft projected area exposed to sunlight (A). First an estimate of solar pressure can be calculated assuming a constant solar flux and \(c\), the speed of light; an estimate of the solar pressure value is 4.56E-6 N/m\(^2\). Equation (5) provides the equation for calculating the acceleration due to SRP. For the orbit determination algorithm, note that the entire SRP coefficient, as seen as the first fraction in Equation (5), is estimated together since each term contains uncertainty.\(^3\)

\[
\vec{a}_{\text{SRP}} = -\frac{p_{\text{SR}} c_R A}{m} \vec{r}_{\text{sat-sun}}
\] (5)

Verification of the SRP model is conducted in the same manner as it was for drag. The reference FreeFlyer orbit is propagated over two days; Figure 9 plots the change in semi-major axis over the course of the propagation. Since the SRP force is only experienced by the satellite when not in eclipse, the first attribute to look for in the plot is flat regions of about 40 minutes. These are clearly identifiable along the top of the sinusoidal curve. Second, for a sun-synchronous orbit with a longitude of ascending node around local noon time (like the FreeFlyer reference orbit) the SRP force is resistive right after coming out of eclipse but accelerating when entering eclipse. The sinusoidal shape in the curve pictures this cyclical force. Finally, the overall reduction in the semi-major axis results from the force being applied while over the equator. When
the spacecraft is over the equator SRP is pushing the CubeSat toward the Earth, reducing the semi-major axis and increasing the eccentricity.

**Figure 9:** Solar radiation pressure causes a cyclical behavior in the semi-major axis.

The final family of perturbations to explore is accelerations due to third bodies. Of the celestial bodies in our solar system, the Sun and Moon, due to their size and distance, respectively, exert perturbations on LEO spacecraft that are four or more orders of magnitude greater than any other body. Since the perturbations from these two bodies are already small, no other third body is selected for incorporation in the model.

Tracking the position of the Sun and Moon with respect to the satellite (or Earth) is the first requirement for including third-body perturbations. In order to incorporate the tracking, several options are available. The first is to expand the state vector to include the Sun and Moon position and velocity and integrate their states alongside the satellites. With regards to memory, as well as computation to some degree, this is expensive and unnecessary. Another approach collects a discrete dataset of points, one data point per day, from the same HORIZONS database. From these data points the actual position is calculated using either two-body propagation or linear interpolation. Each interpolation scheme is tested and compared; results for the Moon are provided in Figure 10 and for the Sun in Figure 11.

For both the Moon and Sun using two-body propagation provides about a two order of magnitude reduction in position error over linear interpolation. When comparing the computational effort from execution time alone, the two-body propagation requires about 70x more time than linear interpolation. Comparing the error reduction and required execution time, it is nearly an even trade; due to improved accuracy, two-body interpolation shows a slight advantage. For this reason, the primary method of calculating the position of a third-body is to use two-body propagation. With the final perturbation in place, error analysis can be conducted to see how the final model improves upon the original.

**Figure 10:** A nearly even trade-off between accuracy and execution time exists when determining the Moon's position using linear or 2-body interpolation.

**Figure 11:** A nearly even trade-off between accuracy and execution time exists when determining the Sun's position using linear or 2-body interpolation.

Figure 12 and Figure 13 provide the 2-norm of the position and velocity error when compared to the FreeFlyer and WISE reference orbits, respectively. Note that except for the Final Model curve the plotted error is not from cumulative models, but rather just shows improvement over the J2-only model. Examining both figures, the most prominent perturbation is SRP. By including SRP the relative error is almost halved for both references. In short, the tuned model provides a balance between computational effort while providing accuracy on the order of kilometers even after propagating ten orbits.

**Figure 12:** Incremental improvements to the force model accuracy compared to the FreeFlyer reference orbit.
Figure 13: Incremental improvements to the force model accuracy compared to the WISE reference orbit.

**Integration Routine**

Selection of an integration routine is the final need for the on-board propagation of team member orbits. Various orders of Runge-Kutta routines are examined from fourth- through eighth-order; comparisons between fixed- and variable-step methods are also provided.

For a satellite in a near-circular LEO the applied forces are consistent in magnitude; therefore, a fixed-step integration routine should be sufficient. In order to test this hypothesis MatLab’s built in `ode45` and `ode113` routines are used for propagation of an orbit using the final model just described; the time steps over a ten-orbit propagation are saved and examined. A tolerance of $1 \times 10^{-12}$ is used. Figure 14 plots the time step size with respect to time for both `ode45` and `ode113`.

The first observation from looking at Figure 14 is cyclical behavior for both routines over the course of an orbit. Since the `ode45` time steps simply vary between 1 and 3 seconds the oscillations seen in `ode113` are of more interest. Over the course of a single orbit spikes in the step size are noticeable twice. This leads to a hypothesis that a discontinuity in the applied forces due to the spacecraft entering or leaving eclipse results in the integrator struggling to meet the integration tolerance. To test this hypothesis a binary value for whether the spacecraft is in eclipse (1) or not (0) is added to Figure 14 and is shown in Figure 15.

Adding the binary value to the plot shows that the sudden drops in `ode113`’s step size line-up almost perfectly with entering or leaving eclipse. Since SRP is determined as the likely culprit for requiring variable time steps, a similar orbit propagation is conducted removing SRP from the force model. Figure 16 shows the results from this new propagation.

Figure 14: Using MatLab’s built-in variable time step integrators with the defined force model results in cyclical behavior of the time step size.

When SRP is removed, `ode113` reaches a steady-state time step of about 55 seconds quickly and maintains that value for except a few anomalous steps; Figure 16 depicts flat time step behavior. Since it appears SRP is the only force requiring a variable time step, and it only requires a variable time step during the discrete steps when it is “turned-off” or “turned-on”, use of a fixed time step method is sufficient. By using a fixed-step method estimation of the total computational effort is simplified and it removes the unnecessary overhead of verifying a sufficiently small step is currently being used.
Using the force model described previously, step sizes of 15, 30, 45, 60, 90, and 120 are examined for each routine. Almost no difference is seen for step sizes of 15 or 30 seconds; Figure 17 provides a logarithm plot showing the difference between these routines and ode113’s result just to show how similar the results are for all of the methods when a small step size is chosen. When the time step increases to 45 or 60 seconds, the RK4 solution diverges slightly, but actually shows improved error for the example orbit. For the largest 90 and 120 second step sizes RK4 diverges with large error and RK5 breaks from the pack with slightly worse accuracy. Plots are excluded for brevity, but are available in a referenced thesis.5

Other than accuracy, execution time is the second part of the balancing equation when selecting an integration routine. Figure 18 provides the execution time for each time step and integration routine considered.

Using the force model described previously, five Runge-Kutta methods are explored to determine if there is an advantage to higher or lower orders with regards to allowable step size. RK4 coefficients are regularly available from many published sources; coefficients for RK5 through RK8 are available from a NASA technical report.10,28

In summary, a relatively simple force model and integration routine provides sufficiently accurate orbit prediction for this application. The next step is to place this model inside of an extended Kalman filter as a piece of the orbit determination software.
station uplinks the latest solution during a communications pass. Although such a method could be employed, assuming a large enough network of ground stations is available, including ground stations removes the autonomy desired of the formation. Realizing autonomous orbit determination requires the selection of an on-board sensor.

Over the past several years low-power and small footprint GPS receivers have been developed by research labs and corporations such as the German Aerospace Center (DLR) and SpaceQuest.\textsuperscript{19,20,26} The accuracy of receivers varies from model to model, but many provide 3σ position accuracy of 10 meters. Velocity solutions are available on some models, but these 3σ accuracies range from 3 to 100 cm/sec, depending on the processing conducted by the unit.

Extended Kalman Filter

Especially if a full state measurement including position and velocity expressed in ECIF is available, using GPS data directly for tracking satellites’ states is an option. One of the research goals, however, is to reduce the power required for tracking the formation. To achieve this goal it was hypothesized that developing a balanced model for on-board propagation while using real-time measurements for state estimation would provide a lower power solution. Combining predictions based on integration via a force model with real-time sensor data, which is known to include Gaussian noise, requires a filter.

Originally described by Peter Swerling but refined by Rudolph Kalman in 1960, the Kalman filter (KF, a.k.a sequential estimation algorithm) has become extensively used in the processing of data from known noisy sensors.\textsuperscript{28} At its root, a Kalman filter provides a means to optimally combine noisy sensor data with predications based on propagation via a dynamic model to calculate a state estimate. Using the filter the state estimate is more accurate than the model prediction or measurements alone. Although the collaborative module assumes a single sensor, the Kalman filter algorithm has no limit on the number of sensor measurements that can be taken into consideration; adding sensors allows for further estimate improvement. Furthermore, it is not necessary for sensors to measure the desired state directly; a mapping between the measurements and state, often referred to as an observation model, is utilized to expand the realm of useful observations. For example, the orbit determination problem seeks a 6D state vector of either Cartesian coordinates or orbital elements; the state itself does not need to be measured but instead range and range rate measurements from ground stations with known locations can be utilized alongside a model propagation in Cartesian space, as long as the mapping from the measurements to the state is provided.\textsuperscript{28}

There are multiple flavors of the Kalman filter including the standard linear filter, the extended, and the unscented. Per the name just given, the standard or linear Kalman filter requires the use of a linear prediction model and linear relationship between the sensor measurements and desired state. Many times a model can be linearized and still represent reality sufficiently, but when a non-linear model is deemed necessary the extended Kalman filter (EKF) provides a solution. In an EKF the state prediction and mapping from the measurements to states can now be non-linear functions. In order to calculate the covariance matrix estimate at each step, however, the Jacobian (a matrix of partial derivatives) of the state and state derivative must be calculated. Depending on the non-linear function(s), the partial derivatives may be analytical expressions, but if not the Jacobian can be numerically estimated. By using the Jacobian for the covariance estimation, the extended Kalman filter is essentially linearizing the functions in the neighborhood of interest.\textsuperscript{28} When a dynamic or observation model is highly non-linear, this linearization can result in the EKF providing poor estimates. The unscented Kalman filter (UKF) uses the unscented transform method to select a minimal set of points around the model mean to be propagated via the dynamic model. From the individual point propagations the mean (new state estimate) and covariance are estimated.\textsuperscript{14} Recalling previous discussion of the force model, the problem at hand is not linear, but does not seem to present itself as highly non-linear: the most non-linear portion being the solar radiation pressure due to entering and leaving eclipse. For this effort, an unscented Kalman filter is not selected for exploration, but future work could compare this EKF implementation with an UKF approach for verification of the near-linear assumption.

Filter Tuning

Early in the filter design process the model noise matrix, \( Q \), was identified as sensitive parameter requiring careful tuning. The model noise matrix provides an estimate of the error in the prediction (based on propagation) for a given unit of time. Including \( Q \) ensures that the filter does not trust the sensor measurements too much resulting in solution divergence. Examining the preceding orbit propagation error results, over a 15 second time step the position and velocity predictions show error on the order of centimeters (1E-9 km) and millimeters per second (1E-12 km/s), respectively. A trade-space of 1E-6 to 1E-15 for both units is explored to select tuned parameters.
The starting point for analysis is actually with the model noise matrix completely neglected (or set to all zeros). Figure 19 shows that the solution begins to diverge immediately and continues to do so. Note that what appears as a red line along the bottom is actually the error between “truth” and the set of GPS measurements; the filter error is far greater than the raw sensor measurements.

Large model noise error is next explored with the position and velocity parameters set to 1 meter and 1 m/s over a 15 second interval. The model prediction is far more accurate than this, so choosing such a large model noise results in the solution diverging even more rapidly than when neglecting it altogether. Figure 20 depicts the quick divergence.

Finally, tuning the parameters to the appropriate values of 1 cm (1E-9 km) and 10 mm/s (1E-11 km/s) for a 15 second interval provides accurate filter solutions. Figure 22 in the following sections shows filter convergence using these model noise parameters.

EKF Validation

Before using the “truth” data identified previously, the EKF is tested using data generated from the same model built into the filter. Position and velocity 3σ noise of 10 m and 30 cm/sec is added to simulate GPS measurements. Figure 22 shows the EKF provides a solution with a fifth of the error from GPS alone.

Since the first set of test data is generated using the defined model, the EKF’s estimate of the six model parameters should show little error. Figure 23 shows the difference between the actual model parameter and the EKF estimate. Over the course of ten orbits the gravitational parameters and $J_2$ are all estimated to nearly the same value as the model. The error shown for the gravitational parameters is 8 or more orders of magnitude less than the actual value and $J_2$’s estimate is 3 orders of magnitude less and appears to still be converging. Although the absolute error for the drag and SRP coefficients is small, the error is on the same order of magnitude or more as the actual values. Nonetheless, the parameters do converge to values of appropriate magnitude indicating the filter is operating correctly.

Testing the filter using “truth” data with added noise is required next. When both position and velocity measurements are available the same results as seen in large steady-state error. Figure 21 shows a consistent error equal to about five times the 3σ sensor accuracy.
Figure 22 are obtained. No additional figures are provided for brevity. A final test of the filter removes velocity measurements since, depending on the selected GPS receiver, they are either non-existent or have high variance. Even with removing velocity measurements the filter provides nearly the same reduction in error. The greatest difference is the warm-up period to reach a converged solution: with velocity measurements only a quarter-orbit is necessary while half an orbit is required without velocity. The primary reason for this is the initial estimate of velocity available. When no velocity measurement is available as an initial condition, the most straight-forward method is to calculate the slope between two GPS position measurements; this results in a relatively inaccurate initial condition so a longer time to convergence is expected. Only two orbits are displayed so that the different data points are more visible; note that the filter continues to perform as seen between 1 and 2 orbits up to the total 10 orbits.

Figure 23: The EKF provides estimates of six model parameters; using test data generated using the same model results in most of the parameters being estimated nearly exactly.

Figure 24: Even when no velocity measurements are available the filter provides a solution with nearly the same reduction in error.
INTER-SATELLITE COMMUNICATION

Enabling autonomous collaboration between CubeSats requires communication among the team. Although ISLs are not uncommon in larger satellites, little has been accomplished in inter-satellite communication for small satellites of the nano or pico scale. As will be summarized in this section, an S-band COTS solution (with flight heritage) for both the hardware and protocol has been selected for the collaborative module. By selecting a commercially available solution, the developed software solely needs the ability to maintain a wired serial communications link with the S-band module. First, details on the message preface and ephemeris message follow. After determining the size and structure of regular communications, trade studies on protocol, RF band, and hardware selection can be conducted. A summary of these studies are provided.

Message Preface/Wrapper

Using a well-defined protocol, such as those considered in this research, assists in data integrity with bit error checking and ensuring that packets received out of order are properly reordered and processed correctly. With these tasks already being handled by the protocol, only an additional small wrapper or preface is utilized in the collaborative module. Table 3 provides the list of six parameters consuming 37 bytes of data.

Table 3: A generic 37-byte message wrapper provides the information a spacecraft team member requires to process the appended data.

<table>
<thead>
<tr>
<th>Byte(s)</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - - - 1</td>
<td>Originator ID</td>
<td>Unique identifier of s/c or specific module of s/c.</td>
</tr>
<tr>
<td>2 - - 3</td>
<td>Destination ID</td>
<td>Unique identifier of s/c or specific module of s/c. Zero is reserved for one-to-all broadcasts.</td>
</tr>
<tr>
<td>4 -- 11</td>
<td>Message ID</td>
<td>Auto-incrementing value for each s/c separately.</td>
</tr>
<tr>
<td>12</td>
<td>Message Type</td>
<td>Identifier of message type. Message types may be mission-specific.</td>
</tr>
<tr>
<td>13 - 20</td>
<td>Timestamp</td>
<td>Time at which a message was first sent. Future re-transmits will use the original timestamp.</td>
</tr>
<tr>
<td>21 - 36</td>
<td>Checksum</td>
<td>Provided so that receiver can ensure received data is correct</td>
</tr>
</tbody>
</table>

The first and second parameters use the same discrete set of available IDs for a mission, with the exception that the Destination ID can be set to 0 to indicate a one-to-all broadcast message. Note that an ID can either be assigned to an entire spacecraft or a specific module on a specific spacecraft; mission requirements will drive which systems require IDs. The Message ID starts at 0 for each spacecraft and auto-increments; this is used by a receiver to determine whether it is a new message or not. For missions that require the sharing of various sensor data the Message Type provides a means to identify what type of data is appended. It can additionally be used to select the appropriate checksum function. The Timestamp is added for good measure as a secondary means to ensure that a newly received message is indeed the most recent message of a specific type. Finally, the Checksum provides a second level of validation (beyond the protocol) to ensure the received data matches the original transmission. This message preface can be used for any inter-satellite communication as a standard header.

Ephemeris Update Message

The contents of the primary state sharing message are a result of the state selected for integration. Besides the thirteen parameters that define the state (position, velocity, attitude, and attitude rates), team members share seven additional parameters. These seven parameters are the gravitational parameters of the Earth, Sun, and Moon; the Earth's J₂ term; the coefficients used for calculating drag and SRP forces; and the corresponding time defined in seconds since a globally selected epoch. Note that for the three gravitational parameters and J₂, rather than sending the whole value the difference between the spacecraft’s estimated value, and a globally defined reference value (the initial condition) is transmitted. For each value in the message 8 bytes are used to represent the number in double precision. Table 4 summarizes the message structure. With a message of this size, common data rates and protocols will complete transmission of the 160 bytes with a single packet and in well under a second.

Note that attitude and attitude rate information is already included in the ephemeris message, although this first research effort focuses on just the orbit propagation and determination portion of the state. Many mission applications may additionally require attitude information in order conduct autonomous task planning.

Sensor Data

The collaborative module design provides no specific sensor data formats since each sensor type requires a tailored format. Once a message type is defined, however, the module takes care of generating the message preface, broadcasting the message, and ensuring it is delivered. Depending on the application, either the standard checksum function can be utilized (such as MD5), or a specific checksum function for each message type can be defined.
The driving factor behind this versatility is to ensure the protocol ensures it meets future inter-satellite link (ISL) needs. A versatile protocol capable of handling packetized data needing to be transmitted, while considering the common message size and data applications.

This research is to provide a single hardware plus board software complexity through code reuse. After considering specific COTS hardware, a final commercial transparent serial communications protocol is considered. Table 5 provides a summary of each protocol’s advantages and disadvantages.

### Table 4: A 20-parameter message provides the data necessary to update a spacecraft’s ephemeris.

<table>
<thead>
<tr>
<th>Byte(s)</th>
<th>Parameter [unit]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 7</td>
<td>x [km]</td>
<td>Cartesian x position</td>
</tr>
<tr>
<td>8 - 15</td>
<td>y [km]</td>
<td>Cartesian y position</td>
</tr>
<tr>
<td>16 - 23</td>
<td>z [km]</td>
<td>Cartesian z position</td>
</tr>
<tr>
<td>24 - 31</td>
<td>a [km/sec]</td>
<td>Cartesian x velocity</td>
</tr>
<tr>
<td>32 - 39</td>
<td>v [km/sec]</td>
<td>Cartesian y velocity</td>
</tr>
<tr>
<td>40 - 47</td>
<td>w [km/sec]</td>
<td>Cartesian z velocity</td>
</tr>
<tr>
<td>48 - 55</td>
<td>q0</td>
<td>1st quaternion parameter</td>
</tr>
<tr>
<td>56 - 63</td>
<td>q1</td>
<td>2nd quaternion parameter</td>
</tr>
<tr>
<td>64 - 71</td>
<td>q2</td>
<td>3rd quaternion parameter</td>
</tr>
<tr>
<td>72 - 79</td>
<td>q3</td>
<td>4th quaternion parameter</td>
</tr>
<tr>
<td>80 - 87</td>
<td>p [rad/sec]</td>
<td>1st component of angular velocity vector relating body and ECIF frames</td>
</tr>
<tr>
<td>88 - 95</td>
<td>q [rad/sec]</td>
<td>2nd component of angular velocity vector relating body and ECIF frames</td>
</tr>
<tr>
<td>96 - 103</td>
<td>r [rad/sec]</td>
<td>3rd component of angular velocity vector relating body and ECIF frames</td>
</tr>
<tr>
<td>104 - 111</td>
<td>lp [sec]</td>
<td>Time state is valid.</td>
</tr>
<tr>
<td>112 - 119</td>
<td>Δμθθ [km^2/sec^2]</td>
<td>Difference from initial gravitational parameter</td>
</tr>
<tr>
<td>120 - 127</td>
<td>ΔJ2</td>
<td>Difference from initial Earth-oblateness parameter</td>
</tr>
<tr>
<td>128 - 135</td>
<td>(cepA/m) [m^2/kg]</td>
<td>S/C drag parameter</td>
</tr>
<tr>
<td>136 - 143</td>
<td>(pmmA/m) [N/kg]</td>
<td>S/C SRP parameter</td>
</tr>
<tr>
<td>144 - 151</td>
<td>Δμθθ [km^2/sec^2]</td>
<td>Difference from initial gravitational parameter</td>
</tr>
<tr>
<td>152 - 159</td>
<td>Δμv [km^2/sec^2]</td>
<td>Difference from initial gravitational parameter</td>
</tr>
</tbody>
</table>

### Protocol

Understanding the common message size and data needing to be transmitted, while considering the unknown mission-specific sensor data sharing, trades between protocols can be conducted. A simple yet robust solution is desirable. Although the aforementioned message structure is tailored to the ephemeris data broadcast among CubeSats, selection of a versatile protocol capable of handling packetized data ensures it meets future inter-satellite link (ISL) needs. The driving factor behind this versatility is to ensure the same protocol can be used for transmitting of mission-specific sensor data between CubeSats. The end goal of this research is to provide a single hardware plus software module that enables collaborative behavior among CubeSats; enabling larger data transfer is a necessity for such a module to have multiple mission applications. Additional factors that play into the trade studies are average data rate, maximum range, robustness, availability of COTS hardware, and TRL (technology readiness level).

The concept of small satellite formations has resulted in the electrical engineering community exploring existing terrestrial protocols to see how robust they are to the LEO environment. In recent years studies have been conducted to determine how common protocols like 802.11 (WiFi) or the newer 802.16 (WiMax) would perform in orbit. Since both of these protocols have plentiful COTS components (although not radiation hardened), have high TRLs, and are known for sufficient data rates for their terrestrial applications, both of these are explored further.

Commonality of use among previous CubeSat missions drove selection of a third protocol for examination: X.25, also referred to as AX.25. This protocol, used by the amateur radio community, is by far the most prevalent from surveying unofficial lists of CubeSat missions. Furthermore, this same protocol could be used for ground communications, limiting the total onboard software complexity through code reuse.

After considering specific COTS hardware, a final commercial transparent serial communications protocol is considered. Table 5 provides a summary of each protocol’s advantages and disadvantages.

### Table 5: Summary of protocol option advantages and disadvantages.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| 802.11 (WiFi) | ▪ Ample COTS components  
▪ Sufficient data rate for small messages  
▪ Can be modified to increase range and max power | ▪ Need radiation hardened COTS  
▪ Current range on the order of 100s meters (some research is pointing towards increasing this limit) |
| 802.16 (WiMax) | ▪ Range up to 50 km  
▪ High data rates  
▪ COTS not as common  
▪ COTS available not radiation hardened  
▪ No ad-hoc support | | |
| AX.25 | ▪ Extensive use on CubeSat missions  
▪ Flexible framework for mission-specific needs | ▪ Open protocol, so needs modification if secure data transfer is required  
▪ Packet handling is manual |
| Microhard, Inc. | ▪ Flight experience  
▪ Plug-and-Play for CubeSat  
▪ 50+ km range  
▪ 128-AES encryption available | ▪ Closed, transparent protocol  
▪ Limited hardware selection |
**RF Band**

While comparing protocols, it is fitting to consider RF bands at the same time, since certain protocols are more frequently found in certain bands. For small satellites, the two primary bands are UHF (usually around 430 or 440 MHz) or S-Band (around 2.4 GHz). Both bands are open spectrum available for use on CubeSats, although licenses are still often required. Table 6 summarizes the advantages and disadvantages of each band. It is the selection of S-Band which also led to the decision to use Microhard Systems, Inc.’s protocol and MHX2420-SL modem for inter-satellite comms.

**Table 6: UHF and S-Band are the two most common RF bands among small satellites and are the only two considered in this research for an ISL.**

<table>
<thead>
<tr>
<th>Band</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>UHF</td>
<td>• Very common among CubeSats</td>
<td>• Likely not preferred for defense applications</td>
</tr>
<tr>
<td></td>
<td>• Lots of proven COTS hardware</td>
<td>• For data-heavy application may provide insufficient bandwidth</td>
</tr>
<tr>
<td></td>
<td>• Sufficient data rate for small messages</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Sufficient range given available W</td>
<td></td>
</tr>
<tr>
<td>S-Band</td>
<td>• Has been indicated to be preference of defense projects</td>
<td>• Less components to choose from, however COTS still available</td>
</tr>
<tr>
<td></td>
<td>• Higher data rates, if needed</td>
<td>• Lower TRL than UHF, but still has CubeSat flight experience</td>
</tr>
<tr>
<td></td>
<td>• Sufficient range given available W</td>
<td></td>
</tr>
</tbody>
</table>

**MHX2420-SL**

The final communications frequency band and protocol are driven by the selection of Microhard Systems, Inc.’s MHX2420-SL spread spectrum commercial S-Band modem. NASA’s GenseSat, a 3U CubeSat experiment launched last decade, utilized the previous generation of this module. The module integrates directly with other CubeSat COTS hardware such as Pumpkin, Inc.’s motherboard. By selecting the extended sensitivity version data rates are slightly slower ranging from 19.2 to 230.4 kbps. Lastly, the module provides variable broadcast power, providing a set of ten discrete power settings range from 0.1 to 1 W. Utilizing the available knowledge of transmission propagation length to set the power, the module can realize further power savings.

**FINAL ALGORITHMS, SIMULATIONS, AND RESULTS**

**Knowledge Maintenance Algorithm**

Ensuring each CubeSat maintains a sufficiently accurate state predication of the other team members requires the defining of an algorithm for broadcasting updates. A simple tolerance check algorithm is implemented in the collaborative module software in order to determine when a CubeSat needs to update its teammates. The following steps describe the process:

1. Initialize tensor of state vectors by each CubeSat conducting a one-to-all broadcast containing a GPS position solution and either an initial velocity guess from differencing or the companion GPS solution.
2. Each CubeSat begins receiving a new GPS measurement every 15 seconds.
3. The on-board processor propagates each team member, including itself, forward in time 15 seconds using either direct propagation or the state transition matrix (STM), described more below.
4. After receiving a GPS measurement, the EKF processes the sensor data along with the propagated prediction to determine a new state estimate including position, velocity, and the 6 model parameters.
5. The new estimate is compared to the straight propagation from Step 3 which is not impacted by GPS measurements.
   a. If the difference between the new estimate and the straight propagation is greater than a mission-defined tolerance, a one-to-all broadcast with the latest state estimate is completed.
   b. If the difference is within a set tolerance, no additional processing is necessary.
6. The algorithm continues with Step 3.

**Advantage of EKF**

With LEO GPS receivers that provide $3\sigma$ accuracy for position of 10 m and some with velocity of 3 cm/sec use of an EKF may be considered unnecessary or even ill-designed. If a selected GPS receiver provides no velocity measurement, an EKF for state estimation is required regardless. However, there is still an advantage to using an EKF when an entire state measurement is available. The first simulation compares the number of update messages sent by each CubeSat using the filter versus not using it. If the filter is utilized then the algorithm just described is in place. With no EKF the satellite is still propagating each team member plus itself for tracking states, but no processing of GPS measurements is completed. Each new GPS solution is compared to current state prediction based on propagation; if the knowledge tolerance is not met then a message is broadcasted.
Figure 25 shows the message count for four different CubeSats over a ten orbit period. With a GPS receiver providing 3σ accuracies of 10 m and 30 cm/s and a knowledge tolerance set to 10 m, the average time between broadcasts increases from 1.87 to 6.86 minutes.

Figure 25: With a GPS receiver providing measurements with 3σ accuracies of 10 m and 30 cm/sec, using an EKF increase the average time between messages from 1.87 to 6.86 minutes when a 10 m knowledge tolerance is in place.

The relationship between the GPS receiver position accuracy and knowledge tolerance greatly determines the effectiveness of the EKF. For example, Figure 26 uses the same GPS receiver as Figure 25, but the knowledge tolerance is loosened to 100 m. Using an EKF in this scenario provides no advantage.

Figure 26: If the knowledge tolerance is much larger than the GPS 3σ accuracy then little to no gain is achieved by the EKF. This simulation uses the same 3σ accuracies of 10 m and 30 cm/sec but a tolerance of 100 m.

To show that it is indeed a function of the relationship between the GPS position accuracy and the tolerance, Figure 27 uses the same tolerance but a receiver providing only 100 m and 1 m/s 3σ accuracies. The reduction in inter-satellite communication is even greater than when the tolerance and accuracy are both 10 m. An order of magnitude reduction in the number of necessary messages provides significant power savings.

Figure 27: If a poor GPS receiver, providing only 3σ accuracies of only 100 m and 1 m/sec, then a knowledge tolerance of 100 m now results in increasing the time between messages from 1.59 to 20.68 minutes.

Another test with this first simulation setup removes the velocity measurement altogether. Using a 10 m tolerance and 10 m 3σ accuracy, average time between messages increased from 1.52 to 4.88 minutes. Figure 28 provides a plot of the message counts.

Figure 28: When only GPS position measurements with 3σ accuracy of 10 m are available, the average time between messages still increases from 1.52 to 4.88 minutes (with no filter 30 cm/sec 3σ accurate measurements must be available).

A final test uses the loose 3σ GPS accuracies of 100 m and 1 m/s but a knowledge tolerance of 20 m. When no
filter is used a message is broadcasted almost every time a new GPS measurement is available (every 15 seconds); using the EKF increases the average time between broadcasts to almost 4 minutes.

**STM in place of Direct Propagation**

Integrating each spacecraft in a team is not overly computational intensive for small teams, but assuming a single processor implementation, the algorithm does not scale. An answer to that problem is to use the state transition matrix (STM) with known initial perturbations ($\delta x_0$) to calculate teammate CubeSat’s position. The STM provides a linearization of the dynamics over the propagated region to map initial perturbations to predict the final perturbation at a future state. Using an STM is scalable since it needs to only be propagated once, and furthermore it is already being propagated within the EKF. Relatively cheap matrix multiplication is all that is required in order to calculate other spacecraft’s state.

A primary disadvantage of this approach is the tracking of a knowledge tolerance. In the first algorithm where a CubeSat propagates each team directly, S/C A is able to know the state that S/C B is predicting for S/C A; this allows for S/C A to easily know when to update S/C B with an ephemeris message. Using the STM removes this ability, therefore the primary check of this simulation is to determine the region of validity for an STM linearization.

Figure 29 provides two different plots. The first shows the actual difference between the prediction from STM matrix multiplication in blue with the tolerance of 10 m plotted in red. If examined closely, the STM prediction quickly drifts, resulting in errors of 100’s km due to linearization. The cause for such poor performance from the linearization is the absolute distance between spacecraft. The lower plot in Figure 29 shows the 2-norm of the distance between S/C A and S/C B.

Generally, an STM has a valid region on the order of 10s or low 100s km, but the figure shows that the two spacecraft drift apart to as much as 2000 km. A second simulation chose two spacecraft with even closer initial conditions; the maximum drift over ten orbits is limited to 300 km, but the STM still results in errors of up to 2 km. If scaling on a single core is absolutely necessary, expanding the STM from first-order into second- or third-order should provide a larger region of validity.

**Parallel Implementations**

One way to handle large formations is to select a multicore computing solution. Recent advances in microcontrollers are bringing multicore solutions to the commercial market such as the XMOS or ARM Cortex chips. Testing on a shared memory platform demonstrates the algorithm’s ability to use these chips.

Two types of scaling need to be explored when testing the parallelization of an algorithm. The first, and more commonly thought of, is referred to as strong scaling. Strong scaling takes a constant problem size and adds computing resources (cores); perfect strong scaling occurs when the speed-up increases linearly with the number of cores. Weak scaling is less common but often more applicable to a particular problem. In weak scaling tests the problem size and computational resources are increased in tandem, so that the ratio of work per core is constant. Perfect weak scaling occurs when the execution time is constant with increasing problem size (since the computational resources are also increasing).

Strong and weak scaling tests are completed on both shared and distributed memory systems. Understanding the hardware is necessary for analyzing the results. Although the long-term goal is to demonstrate the feasibility on multicore microcontrollers, the hardware utilized for testing is a 6-node server. The Hogwarts cluster, available to Georgia Institute of Technology Computational Science and Engineering students, allows for execution of jobs on up to 5 nodes utilizing all 8 cores per node. Each node houses 2 quad-core Intel® Xeon X5570 processors with 8 MB of L2 cache per processor and both Hyperthreading and Turbo Boost technology enabled, allowing up to 16 threads per core to execute concurrently. Forty-eight GB of RAM are available on each node as well as ample hard disk storage. The six nodes are connected by gigabit Ethernet.
**Shared Memory Scaling**

For shared memory testing, code is implemented in OpenMP. The first tests look at just the scaling of the direct propagation of a team of satellites: an embarrassingly parallel problem when the number of threads does not exceed the number of satellites.

Figure 30 provides the strong scaling results for two different OMP implementations: one uses straight OMP pragmas while the other uses the built-in `parfor`. For this problem the formation consists of 32 satellites. Scaling is near-ideal for up to eight cores; the reduction in performance is likely due to the Hyperthreading and sharing of L2 cache. For 9 to 15 cores the flat-line is a result of uneven distribution of work, which is once again realized when all 16 threads are executed.

![Figure 30: Strong scaling for a shared memory implementation shows good results up to 8 cores. Note that the architecture only has 8 physical cores, but allows Hyperthreading.](image1)

Figure 31 shows the weak scaling results. For these simulations the team size is large enough such that there are two spacecraft per core. Once again, near-ideal performance (and some even anomalous timing results showing better than ideal performance) is demonstrated for up to 8 cores. It is still the Hyperthreading and sharing of caches which cause the 40% increase in execution time for teams of 18 to 32 spacecraft on 9 to 16 cores, respectively.

A second OMP implementation goes beyond just the embarrassingly parallel spacecraft propagation and includes the entire update algorithm. In this implementation the thread repeatedly splits and recombines; this can often leads to reduced performance. However, examining Figure 32 and Figure 33, relatively good scaling is observed. The same problem sizes from the first strong and weak scaling test are used. The use of Hyperthreading is not considered in these simulations.

![Figure 32: Strong scaling for the full algorithm still demonstrates good speed-up considering thread splitting/combining and sequential portions of code.](image2)

Figure 33: Weak scaling for the full algorithm still demonstrates promising scaling.
Distributed Memory Implementations

Shared memory architectures are more popular among new multicore microcontrollers, but future designs may find advantages to using distributed memory setups with message passing interfaces (MPI). To test the scalability of an MPI setup, code is implemented using OpenMPI. Only results for the integration portion of the algorithm are provided. Figure 34 provides strong scaling results; Figure 35 plots weak scaling performance. The strong scaling test uses a problem size of 32 spacecraft. Note that only powers of 2 for the number of cores are tested so that the distribution of work is consistent. Four nodes are used for each simulation requiring 4 or more satellites. Up to assigning at least 2 satellites per core, the results are nearly ideal. For the weak scaling results, the decrease in scaling performance between 16 and 32 cores is likely a result of cache sharing on nodes; this increases memory I/O which increase overall execution time per satellite.

Figure 34: Strong scaling for a distributed memory implementation shows near ideal performance when each core propagates at least 2 spacecraft.

Figure 35: Weak scaling for a distributed memory implementation shows promising results.

CONCLUSIONS AND FUTURE WORK

The algorithms demonstrated in this paper provide a solution for one of the pieces needed to enable collaborative behaviors among CubeSats. By using a tuned propagation model and EKF, tracking the Cartesian state of teammates reduces the amount of inter-satellite communication required. Initial parallel implementations of these algorithms demonstrate good scalability characteristics. Using multicore microcontrollers, with core throttling implemented, could provide further power savings; by adding cores with reduced clock speeds, total power consumed is reduced while keeping the execution time constant. This stems from the fact that the power consumed by a processor is proportional to the square or cube of its clock frequency.

With the goal being to provide a technology to enable collaborative behaviors, mentioning of specific implementations which could benefit from this module is necessary. The first, most obvious enabled tasks is collision avoidance. Although not a collaborative behavior, teams of satellites in close proximity must be equipped with a mechanism to ensure collision avoidance. Using the software discussed in this paper provides this capability directly. A second capability provided is a light-weight, standardized message and message preface for inter-satellite communication. Iteration on the message structure may be in order, but defining a standard allows for development of hardware and software which meet that standard. Satellites not originally intended to communicate when in LEO could, as a result of extended mission modifications, be fitted such that inter-satellite communication is already enabled.

Additional work is planned for the messaging framework discussed. Current standards for heterogeneous systems, such as JAUS, are far too large and complex for a low-power CubeSat communications system. A light-weight framework such as the one proposed in this paper is better suited to the CubeSat platform; however, modifications and extensions are needed for the standard to work well for a system of heterogeneous vehicles including unmanned aerial, ground, or surface vehicles.

Situational awareness (SA) is becoming a popular topic among mission proposals. One example mission is the acquisition, observation, and identification of an unknown object in orbit. A team of CubeSats, equipped with visual and infrared imagers, lasers for range measurements, and other sensors can acquire and observe an unknown object. Using the algorithms discussed in this paper provide the two capabilities described in a preceding paragraph, as well as means to
plan data collection. Extending this work to include high level commands for data collection could then use this module’s outputs for mission planning. For example, if a team of three CubeSats is each fitted with a different SA sensor, the CubeSat with an infrared sensor may identify a feature of interest better captured by a visual imager. At the same time, the CubeSat fitted with a laser range sensor identifies a different feature which also merits visual imager observations. The spacecraft fitted with the visual imager receives a message from each of its teammates with a recommended task. With the collaborative module, first, the communication framework is already in place allowing the sharing of sensor data or results. Second, the visual imaging CubeSat already has the necessary information of where each of the other CubeSats was located at the time the feature was detected; therefore it can optimally plan a trajectory, while considering its current tasking, to explore these two other features. The problem has the additional complexity of understanding the unknown object’s orbit and attitude. Once its motion is characterized, however, the module can propagate its state just as if it were a teammate.

A second application, possibly more demanding depending on SA tolerances, is synthetic aperture radar (SAR). SAR takes distributed sensor data such as from an antenna or imager and fuses it to create more valuable data. In the case of distributed antennas, a group of satellites, fitted with small dishes, can “simulate” a much larger antenna that may not even fit on a launch vehicle. For imagers observing the Earth, higher resolution or possibly even 3D renderings of features is enabled with an SAR system. Using the collaborative module for sharing state information greatly simplifies execution of formation maintenance algorithms. SAR has demanding tolerances for pointing and even more so for knowledge. For a team of spacecraft in a SAR formation, having shared state knowledge allows for optimal formation reorganization to occur in a distributed rather than centralized system. With each spacecraft equipped with the same formation reorganization algorithm, each CubeSat will come to the same solution and the reorganization can occur without inter-satellite communication.

Near-term work includes implementing the algorithms on a shared memory multicore microcontroller architecture such as the ARM Cortex. Power consumption measurements can be compared with a simple microcontroller, Texas Instrument’s MSP430 for example, to experimentally determine energy savings. Additionally, extensions to this work to include control algorithms for attitude and orbit control devices within the propagation model and EKF will improve its performance for satellites equipped with such hardware. With these additions, simulations of formations conducting SA or SAR operations will provide further insight into the achievable power savings and overall system efficiency utilizing the collaborative module.

Acknowledgments

The Electronic Systems Lab (ELSYS) of the Georgia Tech Research Institute (GTRI) sponsored the research described in this paper. We also acknowledge Dr. Carlee Bishop, Dr. Ryan Russell, and Dr. Rich Vuduc for their varied technical expertise and assistance throughout our research effort.

References

10. Feehlberg, E., “Classical fifth-, sixth-, seventh-, and eighth-order runge-kutta formulas with


