Development of an Architecture of Sun-Synchronous Orbital Slots to Minimize Conjunctions

B. Weeden[1] and K. Shortt.[2]
Secure World Foundation, Montreal, Quebec J4Z 2A5, Canada
Canadian Space Society, Toronto, Ontario M3K 2E1, Canada

Sun-synchronous orbit (SSO) satellites serve many important functions, primarily in the areas of Earth reconnaissance and weather. The orbital parameters of altitude, inclination and right ascension which allow for the unique utility of Sun-sync orbit limit these satellites to a very specific region of space. The popularity of these satellite missions combined with the use of similar engineering solutions has resulted in the majority of current Sun-sync satellites within this region having very similar inclinations and altitudes while also spaced around the Equator in right ascension, creating the opportunity for conjunctions at the polar crossing points and a serious safety issue that could endanger long-term sustainability of SSO. This paper outlines the development of a new architecture of SSO zoning to create specific slots separating SSO satellites in altitude, right ascension and time at all orbital intersections while minimizing the limitations on utility. A methodical approach for the development of the system is presented along with the work-to-date and a software tool for calculating repeating ground track orbits. The slot system is intended to allow for continued utility of and safe operation within SSO while greatly decreasing the chance of collisions at orbital intersections. This architecture is put forward as one possible element of a new Space Traffic Management (STM) system with the overall goal of maintaining the safe and continued used of space by all actors.

I. Introduction

IN the summer of 2007, a group of 117 students from 24 countries participated in the International Space University Space Studies Program, held in Beijing, China. A subset of 30 students worked on a team project on the subject of Space Traffic Management.[1] One of the concepts discussed as part of that paper is the feasibility of establishing a slot system for the Sun-synchronous region of Earth orbit, similar to that which is used in geostationary orbit. The goal of this paper is further examine the concept of SSO slots through a more rigorous treatment of the subject. In the process, several issues with the original SSO slot proposal in the ISU report will be discussed.

This paper outlines the current situation with regards to SSO, including that of both operational satellites and debris. From there it discusses the current solutions to the problem and how the slot concept works in conjunction with those. This paper then presents a methodology for designing the slot architecture, based on the unique mission design elements of SSO. Finally, it discusses the next steps to be taken and areas for further analysis. The intent of this paper is not to thoroughly explain the underlying orbital mechanics that create the utility of SSO, but rather to build on those concepts and discuss the issue of designing a slot architecture. The authors suggest reading the excellent paper entitled "A-B-Cs of Sun-Synchronous Orbit Mission Design" by R. Boain for background on basic Sun-synchronous orbital mechanics and mission design concepts. [2]

II. Current Situation in SSO

The primary entity tracking objects in Earth orbit is the United States Strategic Command's Joint Space Operations Center (JSpOC) at Vandenberg Air Force Base, California. The JSpOC currently tracks man-made objects in Earth orbit, including both debris and operational spacecraft data is published publically in the satellite catalog on the Space Track website found at http://www.space-track.org. This catalog consists of objects greater

[1] Technical Consultant, Secure World Foundation, 5610 Place Bayard, Brossard, Quebec J4Z 2A5, Canada
[2] President, Canadian Space Society, Parc Downsview Park, 65 Carl Hall Road, Toronto, Ontario M3K 2E1, Canada

than 10 cm in diameter, as this is the generally-accepted lower limit of current tracking capability.[3,4] As of 1 August 2007, out of the entire satellite catalog there were 4192 tracked objects in low Earth near-polar orbits, defined as objects with apogees less than 2,000 kilometers and inclinations between 96.5 and 102.5 degrees. [2] Of these, approximately 138 were active satellites. [2] The region bounded by the aforementioned inclination and altitude ranges will be used in this paper to define the SSO region of Earth orbit. Figures 1 through 4 below were developed using the data from the Space Track catalog and show the distribution of all near-polar objects and all active satellites, respectively, as a function of this inclination and altitude.

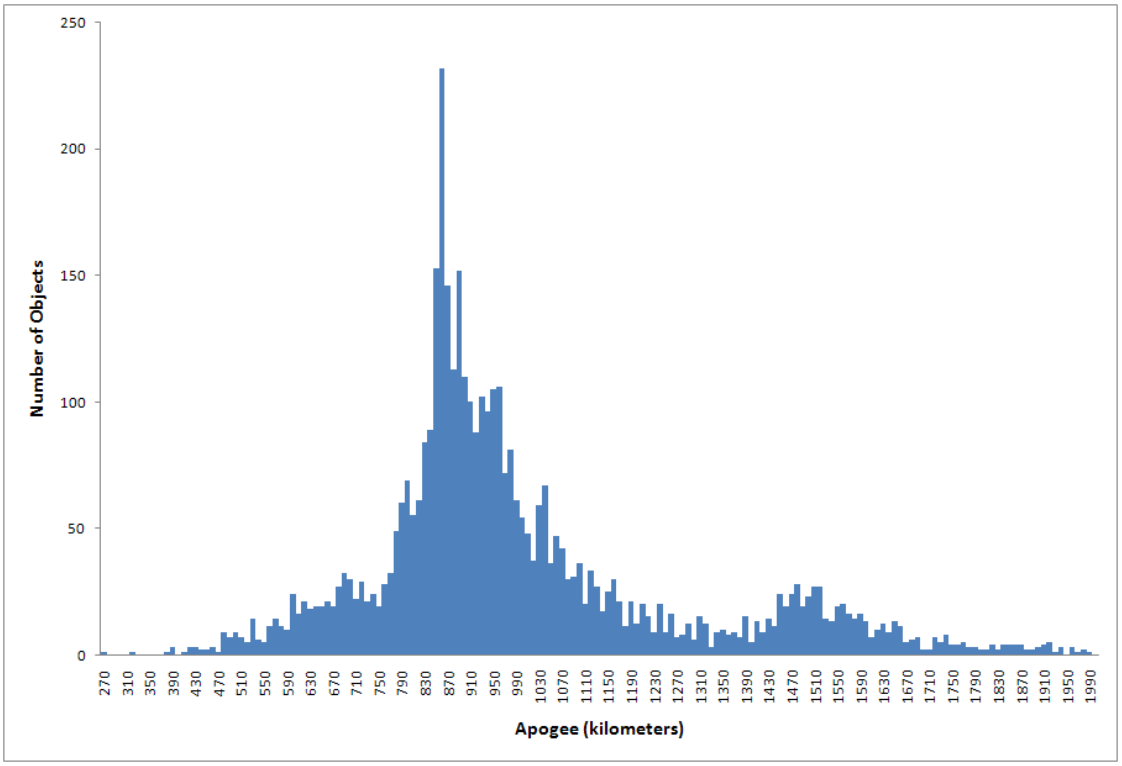

Figure 1. All space objects in near-polar LEO orbit by apogee

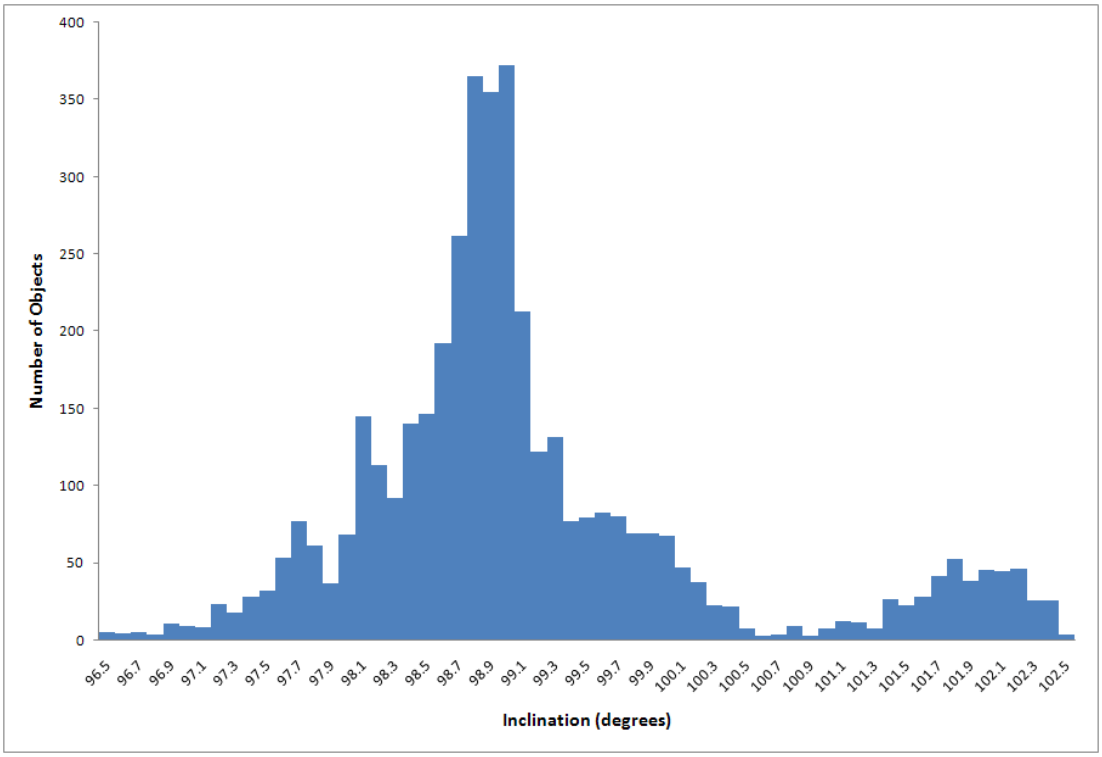

Figure 2. All space objects in near-polar orbit by inclination

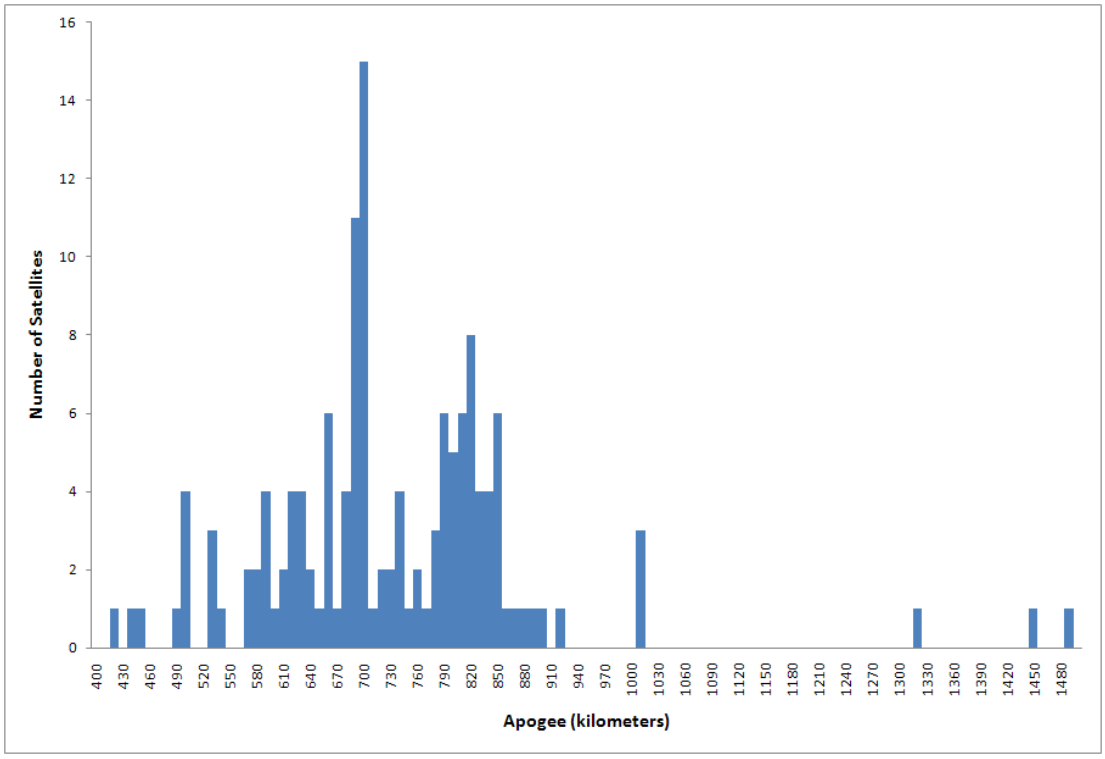

Figure 3. All active spacecraft in near-polar LEO orbit by apogee

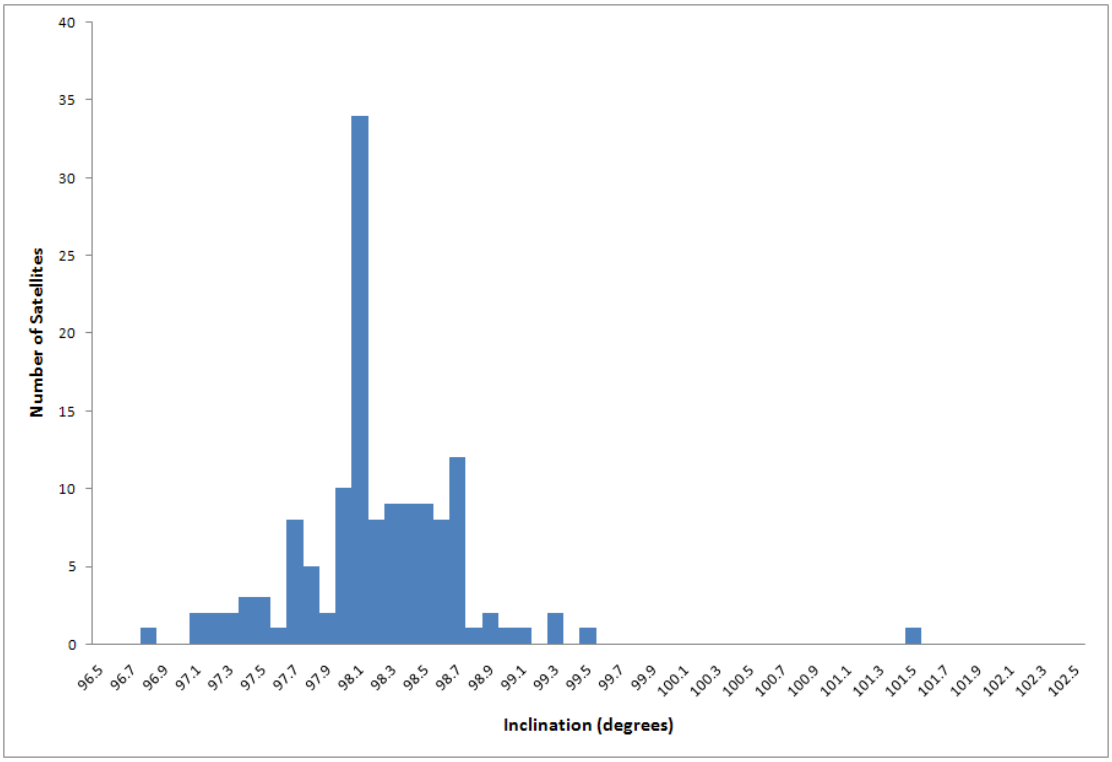

Figure 4. All active spacecraft in near-polar LEO orbit by inclination

From these plots, two things are apparent. First, the two populations, all objects and active satellites, share similar distributions and are located in the same regions of space. This is not surprising as the debris is largely a function of space activity and thus the location of the debris should be strongly correlated with the locations of high space activity. Sun-synchronous orbit certainly counts in this regards. Secondly, the active satellites are clustered in a narrow band of inclinations between 97° and 99° and between 500 km and 900 km in altitude. This indicates that there is a "sweet spot" in the Sun-synchronous zone where mission designers and engineers have coalesced towards common designs and solutions.

Similar graphs showing the geostationary (GSO) population would show an even higher correlation to a specific altitude and inclination, as operational satellites in that region are essentially all following the same orbit but are spaced along that orbit in anomaly (or latitude as seen from the Earth's surface). The situation in SSO is different because there is an additional orbital parameter that is critical to SSO mission design but not GSO: right ascension. SSO satellites are spaced in right ascension around the Equator which creates zones at both poles where the orbits cross every revolution. Figure 5 shows the orbits of the active SSO satellites and these zones can clearly be seen.[5]

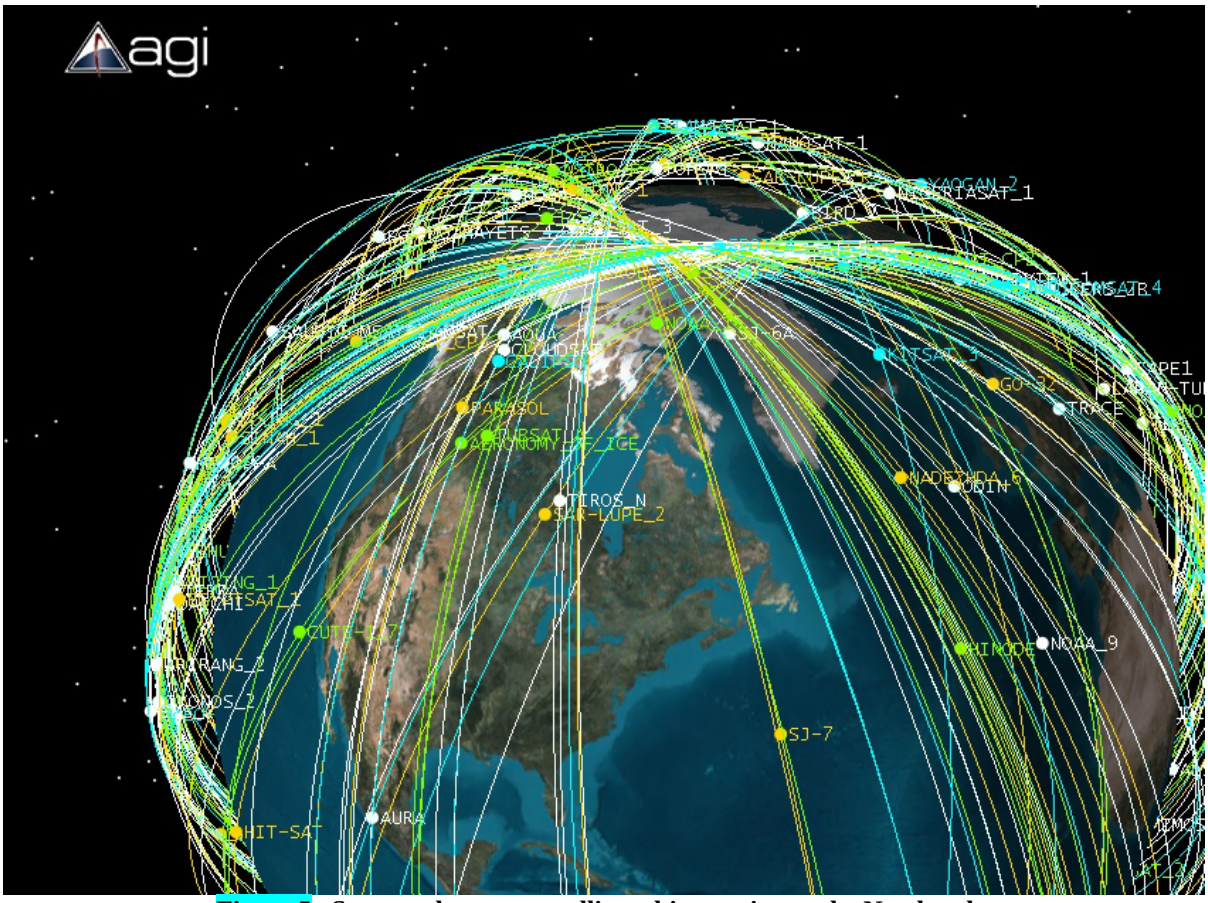

Figure 5. Sun-synchronous satellite orbit crossing at the North pole.

These crossing zones, coupled with the higher debris density, creates a much higher risk of collision for SSO as compared to GSO where all the satellites are essentially moving in the same direction and thus head-on collisions are rare. Collisions in SSO are still rare but the number of conjunctions is steadily increasing, resulting in more spacecraft performing avoidance maneuvers.[6] A spacecraft conjunction is defined here as the situation in which two spacecraft trajectories intersect presenting the possibility of a collision. The process used to determine if such a case exists is called conjunction assessment.

The number of satellites in SSO is expected to continue to grow as more countries seek remote sensing data and services such as Google Earth which utilize the data proliferate. Euroconsult recently released a new report which predicts that 199 additional Earth observation satellites will be launched by 29 countries between 2007 and 2016.[7] Many of those satellites will be placed into SSO orbits. And unlike GSO, there are currently no restrictions on where actors can place satellites within SSO.

This predicted growth of the SSO population, coupled with the lack of structure, indicates there is an increasing need to develop a system or architecture to minimize the probability of collisions and better predict conjunction opportunities. The following sections will outline possible solutions to this problem and their implications on SSO design constraints. This will then lead to the rationale for construction of an SSO architecture that can be developed to minimize spacecraft conjunctions while allowing for maximum utility of Sun-synchronous orbit.

III. Possible Solutions

In order to minimize and potentially solve the problem of SSO conjunctions and safety, a three-pronged approach needs to be taken as summarized by Figure 6. The first essential piece is debris mitigation. These efforts are aimed at both minimizing the creation of debris and the impact of debris on spacecraft. As result of the high priority given to this subject by the United Nations Committee on the Peaceful Uses of Outer Space (UNCOPUOS), the Inter-Agency Space Debris Coordination Committee (IADC) was formed. Over the course of the last few years,

the IADC has developed a set of debris mitigation guidelines which have subsequently been adopted by UN COPUOS. While an important first step, these effect guidelines can have on the overall collision problem is limited. The guidelines can only apply to certain existing satellites and those new satellites launched by States which adhere to the standards.

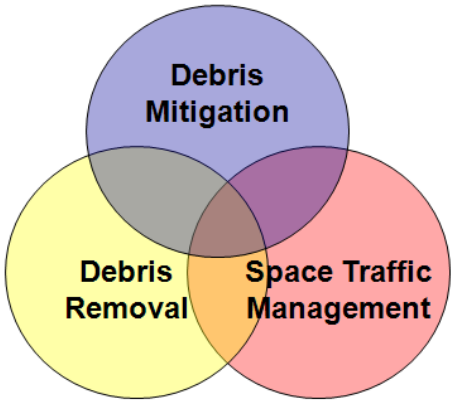

Figure 6. The three main thrusts of space sustainability.

To tackle the problem of existing space debris and uncontrolled satellites, it is necessary to develop a means to remove debris from orbit. Many methods have been proposed, including using lasers to reduce debris' altitude and hasten natural decay, "space tugs" to physically pick up pieces and large "nets" to capture debris. A long-term feasibility study on this topic was commenced at the 2006 International Astronautical Congress in Valencia, Spain. This study is due to report in mid to late 2009. However, most of the current technologies for accomplishing removal of debris are either technically or economically unfeasible and are likely to remain so for some time. This places debris removal as a solution for the future and not the present.

The third piece of comprehensive orbital safety and collision prevention is some level of Space Traffic Management. This is defined as a set of parameters and measures with the aim of maximizing sustainable use and the continued availability of orbital resources while simultaneously minimizing the risk of unintentional physical or radio-frequency interference to operational spacecraft. Within this definition, the concept of defining slots for specific orbital regimes can be an effective solution. An example of this has already been mentioned: the geostationary belt, in which slots of a certain longitude are defined and managed under international law within the International Telecommunications Union. This paper will outline a proposal for a similar slot system for the Sun-synchronous orbit regime.

IV. SSO Mission Design Constraints

To understand why the current SSO population exists in the form that it does, the underlying orbital mechanics which dictate the unique SSO utility must be analyzed, and from this analysis the mission design considerations can be understood. The primary factor in SSO orbits is commonly referred to as the SSO condition. [2] This condition is defined as an orbit who's right ascension, referenced to the First Point of Aries, precesses Westward at a rate of 0.9856 degrees per day.[2] This precession allows the orbit to move around the Earth exactly counter to the Earth's rotation around the Sun, thus preserving the Sun-Earth-satellite angle for which the orbit is so named. For low Earth orbit, the continuum of inclinations and matching altitudes which give the proper precession rate can be calculated.[2]

The second major mission design consideration is the number of revolutions before the satellite's ground track repeats. Repeating Ground Track (RGT) is a key element in the temporal resolution of a SSO satellite. The preceding continuum of inclinations and altitudes can be quantized by eliminating those altitude/inclination pairs which result in a non-integer RGT.[2] Once winnowed in this fashion, an additional key characteristic is discovered. The integer number of revolutions within which the orbit repeats is equivalent to the number of Equator crossings the satellite makes before repeating its ground track.[2] Thus, the maximum swath width at the Equator that the satellite payload must cover is defined. Figure 7 shows the 1 Day and 2 Day RGTs along with their corresponding nodal spacing.

Repeat Interval	Period (minutes)	Altitude (km)	Inclination (degrees)	Node Spacing (km)
1 Day 12 Rev	120.0	1680.78	102.96	3339.58
1 Day 13 Rev	110.77	1262.01	100.73	3082.69
1 Day 14 Rev	102.86	893.72	99.01	2862.50
1 Day 15 Rev	96	566.83	97.66	2671.67
1 Day 16 Rev	90	274.35	96.58	2504.69
2 Day 23 Rev	125.22	1912.71	104.35	1742.39
2 Day 25 Rev	115.20	1464.42	101.77	1603.00
2 Day 27 Rev	106.67	1072.19	99.81	1484.26
2 Day 29 Rev	99.31	725.58	98.29	1381.89
2 Day 31 Rev	92.90	416.66	97.09	1292.74

Figure 7. The 1 Day and 2 Day repeating ground tracks with variation in

The third major mission design consideration is the Mean Local Time (MLT) at which the satellite passes over a location on the Earth and is as function of the orbit's right ascension around the Earth's equator.[2] This feature can be of critical importance for science applications where the satellite is measuring a precise effect requiring specific solar lighting conditions, or when a satellite is designed to add to an existing data set for a particular time of the day. Because of the primary uniqueness of SSO, i.e. the constant satellite-Earth-Sun angle, the right ascension, and thus MLT, can be set at launch and over the course of time the precession of the satellite's right ascension will maintain the proper angle.

The fourth major design consideration is the so-called "frozen orbit." These are orbits which try to "freeze" one or more Keplerian elements by utilizing perturbation effects to balance out overall changes in those elements.[8] An example is the critical inclination, most well-known for its use in the Molniya orbit.[8] These highly elliptical orbits are designed so that a satellite can hang over high latitudes of the Earth for long periods, typically 8 to 9 hours of its 12 hour period. Apsidal line rotation moves perigee around the orbital plane over time, severely limiting the utility of such an orbit. However, at the critical inclination of 63.4° (and its supplement 113.5°), the J2 effects of the Earth dampen out this rotation. A similar effect can be used for SSO satellites to remove the variances in altitude using eccentricity.[2] This is desired to maintain a constant distance between the target and the satellite for consistent imaging.

Thus the orbital characteristics of any SSO satellite is largely constrained by the particular mission that designers wish it to fulfill. The altitude and inclination are prescribed by both the required SSO precession rate and the desired temporal and spatial resolution. The satellite's right ascension of ascending node is defined by the desired MLT, and it's eccentricity by the need to minimize apsidal line rotation via "freezing." The challenge in designing a slot architecture for SSO is restricting the orbits within which a satellite can be placed while simultaneously allowing for continued flexibility in the mission design. A slot architecture which removes all possibility of collision with another satellite but also constrains the utility of SSO leaves the satellite user with a safe but useless orbit.

V. Proposed Slot Architecture Methodology

The authors propose that an SSO slot architecture should initially be designed by first calculating, within reason, all the potentially usable SSO orbits. Once this set of all possible solutions is compiled, it can be crystallized into a structure within which all current and future SSO satellites can be placed. This method also has the benefit of quantizing what is in nature a continuum of orbits and thus reduces an infinite solution set to a finite one.

The first step in this process is to calculate all of the integer RGT orbits from the SSO condition continuum. B. Weeden has written a simple C software program to do this for a given minimum and maximum altitude range. A full description of the program can be found in Appendix A. Using this program and filtering for a minimum and maximum altitude of 250 km and 2,000 km, and between 1 and 20 day repeat intervals, results in 1,673 unique altitude/inclination pairs. To minimize conjunctions, the authors propose that only orbits with an altitude separation equivalent to the positional error plus a safety margin be used. As an example, if all SSO objects were known with a positional error of 4 km, a reasonable safety margin of 1 km could be added and thus SSO orbits with at least 5 km spacing would be required. Within each of these orbital, multiple satellites could be separated by mean anomaly to allow for maximum utilization of the orbit. From the resulting solution set, satellite mission planners can choose their desired RGT and swath width and thus the slot within which they need to place their satellite.

The above slot architecture would significantly decrease the chances of collisions between operational satellites by spacing them in altitude and anomaly and the resulting orbits would never cross under normal situations. However, once mission designers start to specify a range of desired MLTs, the satellites would have varying right ascension around the Equator and the opportunity for conjunctions at the poles is once again introduced. A possible solution to minimizing these conjunctions is to develop a phasing system such that satellites at the same altitude with different right ascension cross the poles at different times. This can be achieved by varying the true anomaly for each of these satellites.

Any proposed SSO slot architecture, such as the one described above, needs to be critically analyzed to determine the proper balance between increased safety and decreased utility. The key question to answer is whether or not such a system is still flexible enough to allow for the continued utility of SSO. Any design constraints such a system imposes on satellite engineering needs to also be examined.

Additionally, the effects of temporal changes in such a slot architecture, both natural and man-made, need to be examined in detail. Many SSO satellites maneuver periodically to correct for changes in altitude and inclination caused by perturbations. Such station-keeping maneuvers need to be studied carefully for their effects on critical orbital elements, such as the anomaly spacing necessary to maintain the phasing at polar crossings. This could possibly lead to the requirement for owner-operators with polar conjuncting satellites to coordinate their station-keeping maneuvers.

VI. Conclusion

The current situation in Sun-synchronous orbit, as described in Section I, presents an increasingly dangerous scenario for the long-term sustainability of this particularly useful region of Earth orbit. This danger is compounded by the current lack of structure to where satellites can be placed within these orbits and the projected growth in usage of SSO. A Sun-synchronous zoning system, similar in theory to that of geostationary orbit, can be one effective component in a solution to this problem

The difficulty in designing a SSO slot system lies in the restraints placed on the Keplerian elements due to the orbital mechanics of SSO. The addition of variability in right ascension makes this a much more difficult problem than developing GSO slots. However, there are potential solutions that would provide the right balance between safety through structure and utility through flexibility. Quantizing the feasible range of inclinations and altitudes provides for a selection of orbits with different ground repeats. Further analysis on methods of phasing polar crossing would allow for flexibility in MLT.

Appendix

This Appendix contains the source code for a simple C program written to demonstrate one possible implementation of the methodology outlined in this paper. The program takes a minimum and maximum altitude and a range of integer days. It then produces all SSO orbits with RGTs over the range of days and within the specified altitude ranges. This data is then fed to an output text file which can easily be imported into Excel as a space separated file. The source code draws upon equations listed in Ref 2 and Ref 8. The software is not intended to be used for any mission planning or engineering purposes but solely as an experimental tool to test the feasibility of this paper's methodology. Figures 8 and 9 below show screenshots of sample user inputs and program out.

Future plans are to add a function that will filter the resulting orbits for a desired altitude spacing and also the ability to calculate the right ascension to provide the required MLT. Once these functions are implemented, the software will be able to generate any one of a number of possible SSO architectures, each of which can then be run through traditional conjunction assessment software to empirically test whether or not it provides any reduction in possible collisions and added safety to SSO satellite operations.

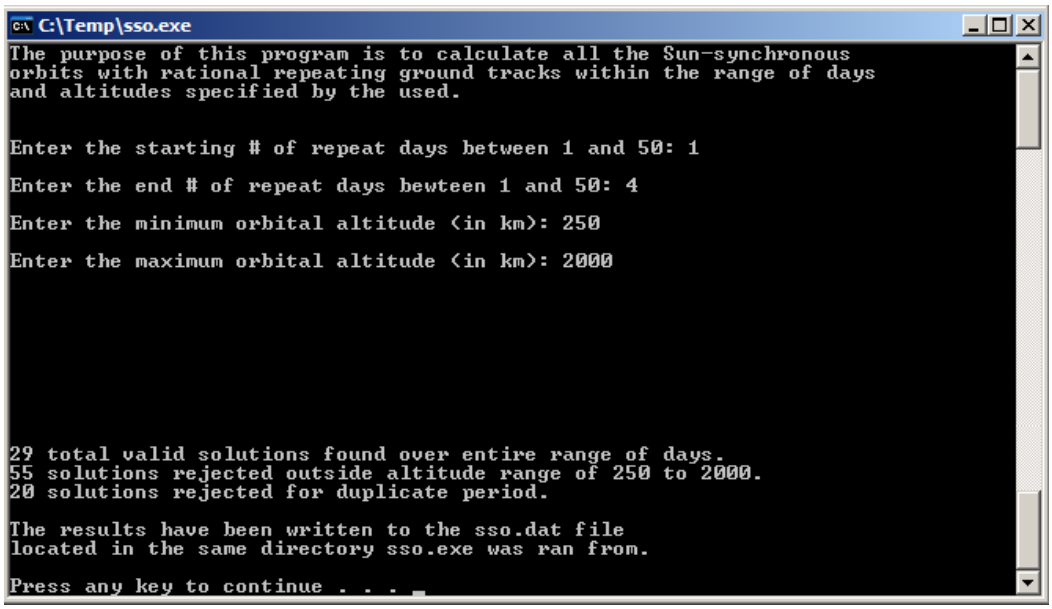

Figure 8. Sample user input and command window output for sso.exe

sso.dat - Notepad

File Edit Format View Help

DTG of Run: Fri Apr 11 14:19:13 2008

Start # of Days: 1
End # of Days: 4
Minimum altitdue: 250 km
Maximum altitude: 2000 km

Repeat	Period (min)	Altitude (km)	i (deg)	a (km)	Node spacing (km)
1D0012	120.000000	1680.782644	102.965684	8058.917644	3339.583677
1D0013	110.769231	1262.016670	100.727818	7640.151670	3082.692625
1D0014	102.857143	893.725258	99.008743	7271.860258	2862.500295
1D0015	96.000000	566.829695	97.660441	6944.964695	2671.666942
1D0016	90.000000	274.354941	96.584383	6652.489941	2504.687758
2D0023	125.217391	1912.713668	104.346910	8290.848668	1742.391484
2D0025	115.200000	1464.419396	101.769937	7842.554396	1603.000165
2D0027	106.666667	1072.186877	99.814143	7450.321877	1484.259412
2D0029	99.310345	725.580287	98.295292	7103.715287	1381.896694
2D0031	92.903226	416.660903	97.093186	6794.795903	1292.742069
3D0034	127.058824	1993.798567	104.855281	8371.933567	1178.676592
3D0035	123.428571	1833.564000	103.863587	8211.699000	1145.000118
3D0037	116.756757	1534.914858	102.150033	7913.049858	1083.108220
3D0038	113.684211	1395.473306	101.406856	7773.608306	1054.605372
3D0040	108.000000	1134.144260	100.105763	7512.279260	1001.875103
3D0041	105.365854	1011.491369	99.534521	7389.626369	977.439125
3D0043	100.465116	780.541675	98.523770	7158.676675	931.976840
3D0044	98.181818	671.661820	98.075528	7049.796820	910.795548
3D0046	93.913043	465.809784	97.275358	6843.944784	871.195742
3D0047	91.914894	368.385011	96.917496	6746.520011	852.659662
4D0047	122.553191	1794.691961	103.630803	8172.826961	852.659662
4D0049	117.551020	1570.761140	102.346796	7948.896140	817.857227
4D0051	112.941176	1361.564369	101.231382	7739.699369	785.784395
4D0053	108.679245	1165.609361	100.256287	7543.744361	756.132153
4D0055	104.727273	981.604053	99.399008	7359.739053	728.636439
4D0057	101.052632	808.423615	98.641420	7186.558615	703.070248
4D0059	97.627119	645.083847	97.968778	7023.218847	679.237358
4D0061	94.426230	490.719535	97.368974	6868.854535	656.967281
4D0063	91.428571	344.566727	96.831987	6722.701727	636.111177

Figure 9. Sample output text file for sso.exe

```
/*--------------------------------------------------------------------------------
This program has been written to calculate the orbital parameters for Repeating
Ground Track (RGT) Sun-synchronous orbits (SSO).  It is a rough outline of the
methodology outlined in the paper "Developmenet of an Architecture of Sun-
Synchronous Orbital Slots to Minimize Conjunctions" by B. Weeden and K. Shortt.

The user will specify the integer number of days to start and stop the calculation
as well as the minimum and maximum acceptable altitudes. The program will run
through all possible orbits that repeat on integer intervals with the specific days
and filter the results to include only those orbits within the prescribed altitude
ranges.  The program will also filter the results to exclude any orbits that were
already found in the previous day's solution set.

Many of the calculations are based on equations in the paper by Ronald Boain titled
"A-B-Cs of Sun-Synchronous Orbit Mission Design" presented at the 14th AAS/AIAA
Space Flight Mechanics Conference, Feb 8-12, 2004.  AAS04-108.

Last Modified: 4/4/08

Copyright (C) <2008>  <Brian Weeden> <brian.weeden@gmail.com>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

--------------------------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

#define PI 3.141592654            //PI in radians
#define CIR_EARTH_KM 40075.16     //Circumference of the Earth in km
#define A_EARTH_KM 6378.135       //Radius of the Earth in km
#define MU 3.986005E14            //mou for the Earth
#define J2 0.00108263             //J2 for Earth
#define OMEGA_SSO_DEG 0.9856      //Rate of nodal change needed for SSO in deg/day
#define NOT_FOUND -1


//Main function.

int main(void)
{
    double
```

```
    P_min,              //Period in min
    P_sec,              //Period in sec
    a_km,               //Semi-major axis in km
    alt_eq_km,          //Altitude at the Equator in km
    inc,                //Inlincation in degrees
    GT_int_deg,         //Ground Track repeat in deg
    GT_int_km,          //Ground Track repeat in km
    R,                  //Repetition Revs
    *period,            //Pointer to array of valid periods
    data[5000][7];      //Array for valid solution data

    int
    D,                  //Repetition Days
    D_init,             //Start # of days
    D_end,              //End # of days
    alt_min,            //Minimum orbital altitude in km
    alt_max,            //Maximum orbital altitude in km
    dupe=-1,            //Duplicate signal
    i=0,                //Day loop counter
    j=0,                //Rev loop counter
    k=0,                //Array loop counter
    l=0,                //Day solution counter
    m=0,                //Overall solution counter
    n=0,                //Rejections due to altitude range counter
    o=0;                //Rejections due to duplicate period counter

    char
    buffer[10],         //Array for user character input
    *pPath;             //Pointer to exe path

    FILE
    *outp1,             //output file 1
    *outp2;             //output file 2


//Intro to the user
    printf("The purpose of this program is to calculate all the Sun-synchronous");
    printf("orbits with rational repeating ground tracks within the range of days");
    printf("and altitudes specified by the used.");

// TODO (Brian#2#): Insert code to ask for desired nodal separation

    /*First we need to get the number of repeat days to start and end from the user
along with the min and max altitudes.*/
    printf("the starting # of repeat days between 1 and 50: ");
    scanf("%d", &D_init);
    printf("the end # of repeat days bewteen 1 and 50: ");
    scanf("%d", &D_end);
    printf("the minimum orbital altitude (in km): ");
    scanf("%d", &alt_min);
    printf("the maximum orbital altitude (in km): ");
    scanf("%d", &alt_max);


//Open the output file
```

```
    outp1 = fopen("sso.dat", "w");

//Insert current DTG timestamp in output file
    time_t rawtime;
    struct tm * timeinfo;

    time ( &rawtime );
    timeinfo = localtime ( &rawtime );
    fprintf (outp1,"DTG of Run: %s", asctime (timeinfo) );

    /*Writing the user inputting values to the output file*/
    fprintf(outp1,"Start # of Days: %d# of Days: %daltitdue: %d kmaltitude: %d
km",D_init,D_end,alt_min,alt_max);
    fprintf(outp1,"Repeat     Period (min)   Altitude (km)     i (deg)       a (km)
Node spacing (km)");

//Initialize the number of days to user input.
    D = D_init;


    /*We now have to dynamically allocate the memory for all the arrays.  The length
of the array will be [5][x] where x is the max limit for the number of solutions.
    Each cell will be equal to the size of a double precision floating point numer.
X is assumed to be 5000 for all practical purposes.*/

    period = malloc (sizeof *period * 5000);


//Now a little message to the user to indication the program is starting.

    printf("# of Days to repeat: %d",D);
    printf("calculations...");

//This for loop will go through each of the days between D_init and D_end.

    for (i=D_init; i <= D_end; i++)
    {

        printf("Loop counter: %d",i);
//This for loop will go through each of the revs between D*10 and D*15 and calculate
period, a, altitude, fundamental interval, and the interval between nodes.

        for (j=D*10; j <= D*20; j++)
        {
            dupe = -1;      //Reset dupe flag
            R = j;
            P_sec = 86400 * D / R;      //Period in seconds is seconds in a day times
days over revs/day
            P_min = P_sec / 60.0;       //Conversion to period in minutes
            a_km = 331.25 * cbrt(pow(P_min,2));       //Semi-major axis from Astro
Cookbook
            alt_eq_km = a_km - A_EARTH_KM;      //From Boain
            GT_int_km = (2 * PI * A_EARTH_KM) / R;      //From Boain
            GT_int_deg = 360.0 / R;     //From Boain
            inc  =  acos(OMEGA_SSO_DEG  /  (-0.013324511  *  cbrt(pow(R/D,7))))*180/PI;
```

```
//From Astro Cookbook (converted to degrees at end)

            printf("Loop counter: %d",j);
            printf("Days to repeat: %d",D);
            printf("Revs to repeat: %lf",R);
            printf("Period: %lf sec",P_sec);
            printf("Period: %lf min",P_min);
            printf("Inclination: %lf deg",inc);
            printf("Semi-major axis: %lf km",a_km);
            printf("Altitude above Equator: %lf km",alt_eq_km);

            /*We now check the calculated altitude to see if it falls within the
user-defined
            max and min.  If yes, we the enter a loop to check the new period against
all the
            previous solutions stored in the period array to ensure it is not a
duplicate.
            If it is a dupe, we exit the loop and proceed to the next rev.  If it is
new,
            it gets written to the end of the period array and the output file.*/

            if (alt_eq_km > alt_min  && alt_eq_km < alt_max)
            {
                printf("Valid altitude.");

                while (dupe == -1)
                {
                    for (k=0; k <= (m-1); k++)
                    {
                        if (period[k] == P_sec)
                        {
                            printf("Duplicate period...");
                            dupe = 1;   //set dupe flag
                            k = m;      //set k to force exit of array loop
                            o++;           //increment counter for duplicate period
rejections
                        }
                    }
                    if (dupe == -1)
                    {
                        printf("New solution found...");
                        period[k] = P_sec;  //write valid period to period array

                        //write all data to data array
                        data[k][0] = D;
                        data[k][1] = j;
                        data[k][2] = P_min;
                        data[k][3] = alt_eq_km;
                        data[k][4] = inc;
                        data[k][5] = a_km;
                        data[k][6] = GT_int_km;

                        fprintf(outp1,"%dD%4.4d       %12f     %12lf   %12f     %12lf
%12f",D,j,P_min,alt_eq_km,inc,a_km,GT_int_km);
```

```
                        l++;          //increment the counter for the valid solutions
for each day
                        m++;         //increment the counter for total valid solutions
                        dupe = 1;    //set dupe flag to exit while loop.
                    }
                }
            }
            else
            {
                n++;    //increment counter for altitude rejections
            }
        }

        /*At the end of each day we need to print out the total solutions for that
day
        and then move to the next day.*/

        printf("%d valid solutions found for %d day repeat.",l,D);
//        fprintf(outp1,"%d valid solutions found for %d day repeat.",l,D);
        D++;    //increment the number of days
        l=0;    //reset the day solution counter

    }

// TODO (Brian#1#): Add filter to search for valid answers separated by x in distance
    printf("%d total valid solutions found over entire range of days.",m);
    printf("%d   solutions   rejected   outside   altitude   range   of   %d   to
%d.",n,alt_min,alt_max);
    printf("%d solutions rejected for duplicate period.",o);
//    fprintf(outp1,"%d total valid solutions found over entire range of days.",m);
//        fprintf(outp1,"%d  solutions  rejected  outside  altitude  range  of  %d  to
%d.",n,alt_min,alt_max);
//    fprintf(outp1,"%d solutions rejected for duplicate period.",o);

// Insert printf to let user know where sso.dat is
    printf("The results have been written to the sso.dat filein the same directory
sso.exe was ran from.");


//Release any array memory.
    free (period);

//Close output file.

    fclose(outp1);

    system("PAUSE");
    return(0);

}
```

Acknowledgments

The authors would like to thank all of their fellow students from the Space Traffic Management Team Project at the 2007 International Space University Summer Session Program in Beijing, where our concept of SSO zoning was originally developed. Special thanks to William S. Marshall of NASA AMES who was our mentor and the Team Project Co-Chair. The authors are especially indebted to Ronald J. Boain from the Jet Propulsion Laboratory for his excellent paper, *The A-B-Cs of Sun-Synchronous Orbit Mission Design*, which provided much of the theoretical foundation for this paper. Finally, B. Weeden would also like to thank Analytical Graphics, Inc. for their generous donation of their Satellite Tool Kit software to help with this project.

References

[1]Anilkumar, AK et al, "Space Traffic Management," International Space University, Space Studies Session 2007, Beijing China. [online], URL: http://www.isunet.edu/index.php?option=com_docman&task=doc_download&gid=371 [cited 1 February 2008].

[2]Boain, R. J., "A-B-Cs of Sun-Synchronous Orbit Mission Design," *Advances in the Astronautical Sciences*, Vol. 119, Part 1, 2004, Pages 85-104.

[3]Committee on Space Shuttle Meteoroid/Debris Risk Management, *Protecting the Space Shuttle From Meteoroids and Orbital Debris*, National Academies Press, 1997, pp. 36.

[4]Klinkrad, H., Space Debris: Models and Risk Analysis, Birkhauser, Berlin, pp 2.1.

[5]STK, Satellite Toolkit, Software Package, Ver. 8.1.1, Analytical Graphics Incorporated, Exton, PA, 2007.

[6]NASA Orbital Debris Program Office, *Orbital Debris Quarterly News*, Vol 11, Issue 4, 2007, pp. 2.

[7]Keith, A., *Satellite-Based Earth Observation – Market Prospects to 2017*, Euroconsult, Paris, 2008.

[8]Vallado, D.A., *Fundamentals of Astrodynamics and Applications*, 3rd ed., Springer-Microcosm Press, New York, 2007, pp. 838.