N93-25967

Adaptive Laser Link Reconfiguration Using Constraint Propagation

M. S. Crone, P. M. Julich, L. M. Cook

HARRIS Corporation
GASD - 102/4844
P.O. Box 94000
Melbourne, Florida 32902
(407)729-3163

ABSTRACT

This paper describes Harris AI research performed on the Adaptive Link Reconfiguration (**ALR**) study for Rome Lab, and focuses on the application of *constraint propagation* to the problem of link reconfiguration for the proposed space based Strategic Defense System (SDS) Brilliant Pebbles (BP) communications system. According to the concept of operations at the time of the study, Laser communications will exist between BP's and to ground entry points. Long-term links typical of RF transmission will not exist. This study addressed an initial implementation of BP's based on the Global Protection Against Limited Strikes (GPALS) SDI mission. The number of satellites and rings studied was representative of this problem.

An orbital dynamics program was used to generate line-of-site data for the modeled architecture. This was input into a discrete event simulation implemented in the Harris developed **CO**nstraint **P**ropagation **E**xpert **S**ystem (**COPES**) Shell, developed initially on the Rome Lab BM/C^3 study. Using a model of the network and several heuristics, the **COPES** shell was used to develop the **H**euristic **A**daptive **L**ink **O**rdering(**HALO**) Algorithm to rank and order potential laser links according to probability of communication. A reduced set of links based on this ranking would then be used by a routing algorithm to select the next hop.

This paper includes an overview of Constraint Propagation as an Artificial Intelligence technique and its embodiment in the **COPES** shell. It describes the design and implementation of both the simulation of the GPALS **BP** network and the **HALO** algorithm in **COPES**. This is described using a Data Flow Diagram, State Transition Diagrams, and Structured English PDL. It describes a laser communications model and the heuristics involved in rank-ordering the potential communication links. The generation of simulation data is described along with its interface via **COPES** to the Harris developed **ViewNet** graphical tool for visual analysis of communications networks. Conclusions are presented, including a graphical analysis of results depicting the ordered set of links versus the set of all possible links based on the computed Bit Error Rate(BER).

Finally, future research is discussed which includes enhancements to the **HALO** algorithm, network simulation, and the addition of an intelligent routing algorithm for BP.

1. SDI BRILLIANT PEBBLES COMMUNICATIONS

During the course of the **ALR** program the space-based architecture was changed to be based on the concept of "Brilliant Pebbles (**BP**)". Each **BP** consists of a weapon system, sensor system, and a communications system. The focus of the **ALR** was on the communications system, as is that of this paper.

1.1. LINK RECONFIGURATION FOR BRILLIANT PEBBLES

The **BP** network consists of many platforms, each of which can receive simultaneously from a large number of neighbors, but which can only transmit via a laser to one other platform at a time.

This work was funded by the U.S. Air Force Rome Laboratory under contract number F30602-89-D-0096

It also runs in an open-loop fashion using simplex links, where a platform calculates the position of other platforms based on orbital predictions and periodic position updates. It then points at the selected platform only for the duration of a message. A link is never established in the manner of a typical RF architecture. There is, however, the possibility that pebbles will not recalculate links on a per message basis. Although the study does not address this possibility, the **COPES** implementation of the **HALO** Algorithm could be modified in a straightforward manner to accommodate such a change.

Given the large number of potential links from any node, a routing algorithm should not have to consider all potential nodes every time a message is sent. To work effectively it should only have to consider a subset of the links. This approach requires a database to be maintained which can effectively rate links according to constraints such as, longevity of LOS, range, probability of accurate position data for other nodes, beam-width limitations, probability of jamming, etc. Maintaining such an intelligent data base can significantly speed up the routing algorithm, and make it more robust in the face of enemy actions. The concept of link reconfiguration was redefined under the **ALR** program to mean the process of creating and maintaining such a database of rated links.

1.2. GLOBAL PROTECTION AGAINST LIMITED STRIKES (GPALS)

During the **ALR** study we simulated an initial implementation of the Brilliant Pebbles SDI architecture based on the Midcourse and Terminal Tier (MATTR) and Global Protection Against Limited Strikes (GPALS) studies. The SDI architecture has been radically altered since the BM/C^3 study, (Crone, Julich, 1990) with the incorporation of Brilliant Eyes(**BE**), Brilliant Pebbles(**BP**), and the Endo / Exoatmospheric Interceptors (E^2I). In addition, the concept of Battle Management has evolved, including both the location of Battle Managers and modes of operation. The MATTR study defined a BP-based SDI architecture which includes the midcourse and terminal phases. The GPALS study defined requirements for an SDI system which addresses a more limited size strike which may originate from any location. The GPALS architecture represents an initial but scaled-down version of an eventual phase 1 architecture, with less **BPs** and **BEs**, and without the GSTS system.

A significant difference in the mission of GPALS (as contrasted with the full scale SDS) is indicated by the name. First, the system provides *Global Protection*. The space elements of the system are intended to support defense both within CONUS and in overseas theaters. Advantages can be obtained by using a common communications architecture for the overseas theater and the CONUS implementation. Second, the term *Protection* suggests a different mission from the earlier SDIO Phase 1 architecture. The Phase 1 architecture had a primary mission of attack deterrence. Providing protection (zero leakage of attacking missiles) indicates a requirement for increased reliability and less probabilistic focus. This affects the communication requirements by placing a higher emphasis upon guaranteed delivery of messages. Third, the term *Limited Strike* indicates a smaller threat than the massive strikes considered in the Phase 1 (full scale) architecture.

The GPALS architecture is more distributed than previous architectures, with Battle Management being distributed along both regional and element lines. Weapon Target Assignments (WTA's) are generated much closer to the weapons. Control of the battle is hierarchical, however, through the use of Preplanned Response Options (PROs), Defense Employment Opportunities (DEOs), and Weapons Release Authority (WRA).

Figure 1 provides a view of the MATTR/GPALS architecture and connectivity. The legend describes the various elements involved in the battle. Control of the system is hierarchical beginning at the Command Center (CC) and proceeding through Regional Operations Centers (ROCs) and Element Operations Centers (EOCs).

In GPALS, battle management is distributed and co-located with sensor systems such as Brilliant Eyes(**BE**), Brilliant Pebbles(**BP**) and Ground Based Radars(GBRs). The **ALR** study was primarily concerned with track reports originating with **BPs** that are filtered by merge nodes as they are passed toward a GEP.

2. CONSTRAINT-BASED ALGORITHM DEVELOPMENT

Given the requirements of the GPALS simulation and the need to develop a heuristic algorithm to maintain a reduced set of potential links, constraint propagation as embodied in the **COPES** shell was chosen to accomplish both tasks. The application of constraint propagation to intelligent

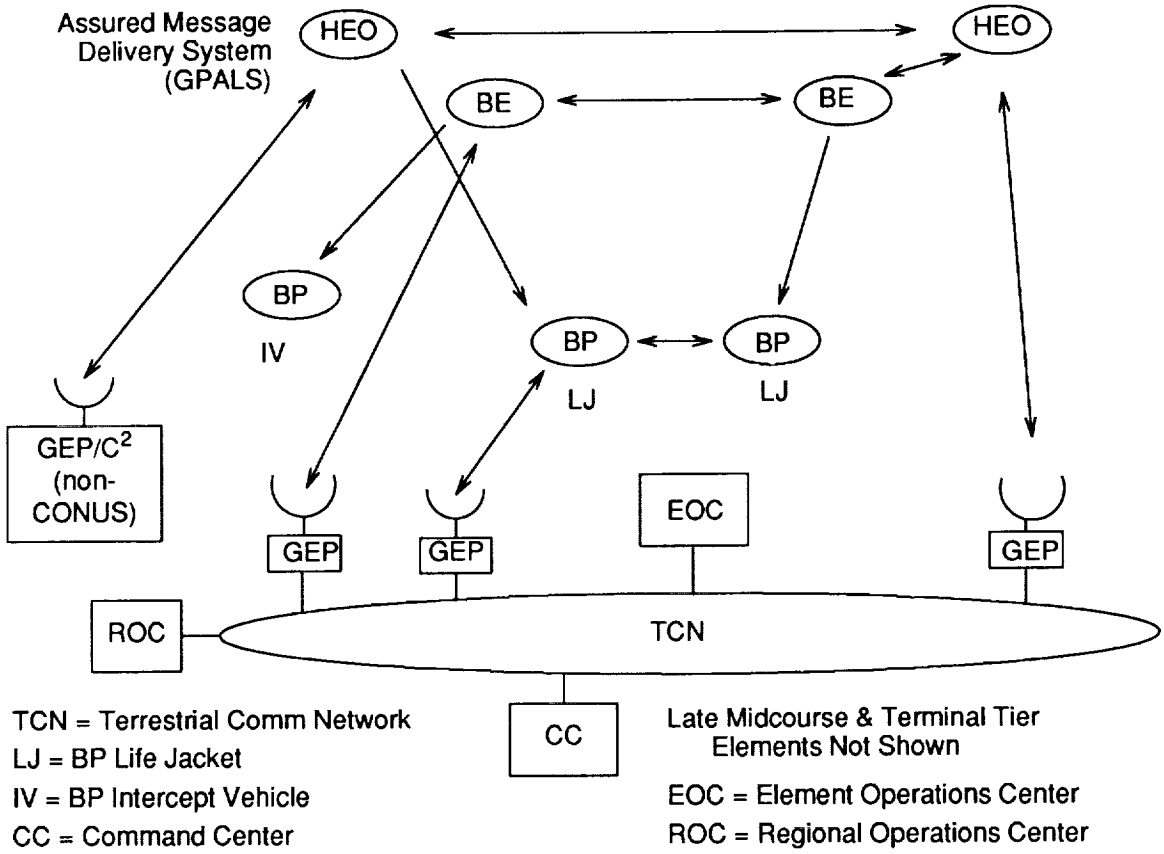

Figure 1 - Space-based GPALS Communications Architecture

problem solving was begun during the BM/C^3 program with Rome Lab.That effort resulted in the initial development of the **COPES** shell.(Crone, Julich, 1990) **COPES** was subsequently used in developing a distributed intelligent network manager in cooperation with the C Language Integrated Production System (CLIPS) rule-based language under the Distributed Intelligent Network Control (DINC) program for the U.S Army Strategic Defense Command. (Crone, Julich, 1991) This research was performed in cooperation with the Professor Ramamoorthy and students at the University of California Berkeley. The subject of this paper is the work done in intelligent link assignment for the SDI laser communications space network.

Since this work, we have used **COPES** to implement a version of the new Arpanet Shortest Path First(SPF) algorithm (McQuillan, 1980), a version of simulated annealing using **COPES** for the traveling salesman problem as a precursor to the problem of Weapon Target Assignment under Harris research, and have developed a neural network development tool. The simulated annealing technique utilized the discrete event scheduling capability of **COPES** to produce solutions to a 95 city problem which were consistently within 94% of the optimum solution. The **COPES** developed neural network tool is based on the Parallel Distributed Processing Project. (McClelland, 1988). In each case, as in all COPES development, the problem is represented in a distributed manner without a central executive process. In addition, the solution is distributed throughout the object database. For instance, in the case of the SPF algorithm, the shortest "next hop" to a particular node is maintained in the object representing the that node, as opposed to being maintained in a centralized routing table.

The remainder of this section gives the principles of constraint propagation in the context of Artificial Intelligence; its implementation in the **COPES** shell; and the current status of the shell as a basis of implementation for the **HALO** algorithm.

2.1. BACKGROUND

2.1.1. Traditional Search Techniques

A variety of search techniques exist, in the area of *combinatorial minimization* with an objective function to be minimized. (Press 1986) We have investigated some of these in the similarly complex domain of planning and have found them to be inadequate for knowledge-based problems. In particular the *Simplex Method* is a linear programming technique which can maximize a function subject to a set of constraints. This was found to be too slow and unable to provide partial schedules if all constraints could not be met. The *Constrained Minimization* technique in which a cost function is to be minimized subject to a set of constraints depends on the function being continuously differentiable. In this case standard techniques such as Penalty Function methods, and Conjugate Gradient methods could be used. Because of the discrete nature of planning and link assignment, such a function cannot be found. The concept of a heuristic cost function can be useful using a AI approach, however. Used in this way the function is used to evaluate potential link assignments and guide further refinement.

The method of *simulated annealing* is a technique which for practical purposes has solved the "traveling salesman" problem. It has been used successfully for designing complex integrated circuits to minimize interference among connecting wires in the arrangement of several hundred thousand circuit elements on a silicon substrate. These are both applications of *combinatorial minimization*. There is an objective function to be minimized over a discrete but very large configuration space. The method of simulated annealing based on the Metropolis algorithm always takes a downhill step while sometimes taking an uphill step thus avoiding being trapped in a local minima. (Press, 1986) It is applicable in cases where a simple measure of an objective function (analog of energy) can be defined. For most complex problems this cannot be defined by one function.

2.1.2. Artificial Intelligence

State space search is used in AI to move from an initial state representation of the problem(such as all potential links for each CV platform) to a goal state by the application of knowledge-based operators. (Rich, 1983) Given the nature of the initial and goal states, this search can be combinatorial. Algorithmic techniques exist such as *branch-and-bound*, and A* to reduce the search for some problems, but are inadequate for knowledge-rich problems. In this class of problem the cognitive activity of an intelligent agent involves two types of search: (1) *knowledge search*, that is, which operator to apply next, and (2) *problem-space search*, that is, search within the problem space for a goal state. (Gupta, 1983) Pruning of both spaces is crucial to reducing the search. In order to solve many hard problems efficiently, it is often necessary to to construct a control structure that is no longer guaranteed to find the best answer, but that will almost always find a very good answer. This is called *heuristic search* because knowledge is used to guide the search process. Heuristic search can be applied implicitly via the pattern matching of the rules against the problem-space data which takes place on each cycle in a production system, or explicitly via the weighting of constraints as in *constraint-directed* search in the ISIS scheduling system. (Fox, 1983) Blackboard Architectures address many of the issues of state space search and have been suggested as a control mechanism for problem solving. (Hayes-Roth, 1983)

Systems which take advantage of a great deal of knowledge are referred to as "Expert Systems", and have been shown to provide problem-solving computer programs that can reach a level of performance comparable to that of a human expert in specialized problem domains. (Barr, 1982) (Gevarter, 1983) They are in fact a form of qualitative model of both the problem space and the human problem solver. (Clancey, 1986) Expert Systems are characterized by the separation of data, rules(knowledge), and control. (Crone, 1985) They are usually rule-based, and due to the often enormous amount of pattern matching in rule-based systems, have not fared well in real-time applications. Some speed-up is predicted via parallel processing. (Gupta, 1983) Given the often autonomous intelligent activity which would be required in the link assignment problem, Expert Systems will be required, so research must uncover faster inferencing mechanisms. Rather than considering all data and knowledge in every inferencing cycle a more "object-oriented" (Stefik, 1986) knowledge representation is suggested. This is especially appropriate in cases where the problem space is in the form of a network with each node being connected to surrounding neighbors. The approach taken by this research is to use *constraint propagation* as the method of inference.

2.1.3. Constraint Propagation

An architecture which has been used with varying degrees of success, in physical reasoning, temporal reasoning, and spatial reasoning is to represent the knowledge base as a *constraint network* which performs inference by *propagating* labels. (Davis, 1987) These labels represent potential candidate values for nodes in the network.

2.1.3.1. Constraint Networks

A constraint network is a declarative structure which expresses relations among parameters. It consists of a number of *nodes* connected by "constraints". (Davis, 1987) A node represents an object which contains state and which is represented by the *value* of the instance variables of the object. A constraint represents a relation among the instance variables of the node and those of other objects it connects. As such, it is usually local in scope, but can connect all nodes in the case of a global constraint such as a heuristic weighting function. Examples of different applications of constraint propagation are numerous. (Crone, Julich, 1987,1990,1991) (Davis, 1983) (Fox, 1983) Forward inference on constraint networks, called *assimilation*, is usually done using *constraint propagation*, shown in algorithm 1. In constraint propagation, information is deduced from a local group of constraints and nodes, and is recorded as a change in the network. Further deductions will make use of these changes to make further changes. Thus, the consequences of each datum gradually spread throughout the network.

Algorithm 1. - Constraint propagation
repeat
-take some small group of constraints and nodes in some connected section of the network,
-update the information in this section of the network, given the information in the constraints and the nodes;
until no more updating occurs (the network is quiescent) or some other termination condition is reached.

In its most basic form, a set of potential labels for each node's instance variables are given, and then reduced based on constraint propagation to a unique solution or an inconsistent state.

A greater depth of knowledge concerning a particular system can be expressed in terms of constraints than is possible in a rule-based system alone. Model-based reasoning is a common application of constraint propagation where expected performance of a system is described through a set of constraints, which may contain mathematical models. Deviation from this behavior or observed similarities to expected failure modes can trigger corrective action or alter resource planning. This type of diagnostics is known as specification based as opposed to the symptom based approach used in rule-based diagnostic expert systems.

Unlike commercial AI shells such as ART, constraint propagation takes advantage of locality of information. Some of its valuable properties are:

- Forms a close analogy for systems in which physical effects propagate across connections between components.
- Constraint propagation consists of a simple control structure similar to a rule-based inference engine.
- Degrades well under time limitations; interrupting the process in the middle gives useful information already deduced.
- With assimilation, it is easily implemented in parallel, since updating can be performed all over the network simultaneously.
- Is easily expanded by adding constraints incrementally to the network.

2.1.3.2. Inferencing in COPES

In assimilation, the instance variable values for each node are represented by a set of labels which must be consistent with constraints relating the instance variable to those of other nodes. The general form of refinement is given by the following definition: **Definition 1.** Let C be a constraint on nodes $X_1, \ldots X_k$. Let S_i be the label set for X_i. Then

$$\text{REFINE}(C,X_j) = \{a_j \in S_j \mid \exists\, (a_i \in S_i, i=1,\ldots k, i \neq j) \; C(a_1,\ldots,a_j,\ldots a_k)\}.$$

That is, REFINE(C, X_j) is the set of values for X_j which is consistent with the constraint C and with all the labels S_i. A value a_j is in REFINE(C, X_j) if a_j is in S_j and it is part of some k-tuple $a_1, \ldots, a_k$ which satisfies C and all the S_i.

Applying the updating function REFINE within the constraint propagation control structure given in Algorithm 1, gives the Waltz algorithm. (Waltz, 1975) The Waltz algorithm applies constraints to nodes until no more changes occur (the network has reached *quiescence*). Algorithm 2 is an efficient implementation of the Waltz algorithm which served as the original basis for this research with many additions being added over time.

Algorithm 2. - Waltz Algorithm

```
/* The set
S_i is the current label set of quantity X_i */

REVISE refines all the parameters X_1 . . .
,X_k of a given
constraint C, and returns the set of all parameters
whose set was changed.

procedure REVISE(C(X_1 , . . . X_k ))
 begin CHANGED ← ø
  for each argument X_i do
   begin S ← REFINE(C, X_i)
    if S = ø then halt
    else if S ≠ S_i then
     begin S_i ← S
      add Xi to CHANGED
     end
   end
  return CHANGED
 end


procedure WALTZ
 begin Q ← a queue of all constraints
  while Q ≠ ø do
   begin remove constraint C from Q
    CHANGED ← REVISE(C)
    for each X_i in CHANGED do
     for each constraint C' ≠ C which has X_i in its
        domain do
        add C' to Q
   end
 end
```

2.2. APPROACH

2.2.1. Development of COPES Shell

In order to effectively apply Waltz's algorithm to a network type of problem we designed and implemented the **COPES** Shell at Harris to merge the concepts of *constraint propagation* as a method of inference, with *object-oriented programming* as a method of representation. Unlike most applications of constraint-based reasoning, the use of **COPES** provides a solution which is easily created and updated. The representation scheme allows the hierarchical definition of complex objects called classes which contain state information in the form of instance variables, and links to constraints which are applied to them. For some problems, constraints are inadequate to produce a unique state. We added a searching mechanism to the shell which allows back-tracking with or without selective pruning. An instance variable defined for a class is really represented by a complex data structure which includes its type, name, parent, etc. This allows generic functions to be developed where variable type is dynamically bound. This is similar to a Lisp/Flavors approach. Among the objectives of this work was to build the shell using the language "C" on a Unix environment such as the Harris HCX-9, and to emphasize run-time speed. Since a large part of the knowledge base is programmed directly in "C", and locality of information is considered; **COPES** offers an execution time advantage over rule-based systems.

The building of a tool to support Expert System development is compounded by the need to experiment with the tool during development to expose limitations and problems. The flexibility required for AI tool development leads one to follow the principles of object-oriented design, where possible. At Harris we have built object-oriented versions of both C and Ada to make implementation possible in a conventional environment. (Crone, Julich, 1987) (Simonian, Crone, 1989) For the sake of run-time speed, we did not use either for the development of **COPES**, but did follow many of the principles of object-oriented programming.

Object-oriented design methodologies typically start with an emphasis on the data representation. In order to support AI design, we added extensions to Entity-Relationship (E-R) diagrams which are typically used for database design. (Chen, 1976) To describe the software design, we modified the Jackson Structured Design(JSD) process model. (Cameron, 1986) The AI and software designs of **COPES** are described in detail elsewhere. (Crone, Julich, 1990)

2.2.2. Knowledge Representation in COPES

Knowledge takes two forms in **COPES**: (1) the constraint network, and (2) the constraint functions. The application of the constraints to the network is the function of the shell.

The creation of such a constraint network database is done either interactively for small problems or is read from a Unix file created by a C program. In most problems amenable to solution via constraint propagation, a network is often homogeneous with identical constraint relationships between neighboring nodes. We are currently developing a "class" language and "instantiation" commands to make the creation of such a network easier and more dynamic.

2.2.3. State of the COPES Shell

The following is a description of the current state of development of the **COPES** shell and some of the problems to which it has been applied.

- Hierarchical knowledge representation scheme using object oriented approach
  - — Metaclass, classes, subclasses
  - — Instance variables to define object state
  - — Class constraints (definition and instances)
- Waltz constraint propagation based on representation scheme.
- State saving and Restoring callable by constraint routines
- Abstract variable types with generic access methods
- Container variable types which support queues, stacks, sets, etc.
- Container variables also support distributed problem solving via TCP/IP sockets.
- Scheduled variable modification for discrete event simulation including cancellation of events.
- Variable access methods such as PUT, GET, WRITE, etc. either direct or "BY_NAME", where the class object and variable name is given.
- Database features to list class structure, constraint bindings, and error messages during creation
- A variety of tracing features for debugging
- Problems to which COPES has been Applied
  - — Distributed Intelligent Network Management
  - — Distributed Network Emulation
  - — Distributed Heuristic Algorithms
  - — Simulated Annealing WTA Research
  - — model-based diagnostics
  - — Modeling Neural Networks
  - — Distributed algorithms: A*, N-queens, SPF, TSP
  - — Dataflow-based discrete event simulation for SDI

3. HALO ALGORITHM USING COPES

This section describes the **HALO** Algorithm and its implementation using the **COPES** shell. We first introduce the Adaptive Link problem and develop a model to analyze it. Then we discuss the heuristics of the algorithm and consider their relevance to the actual Brilliant Pebbles scenario. Next, we discuss how the model is implemented using the **COPES** shell as a simulator. Finally, we discuss some results obtained from running the simulator on a typical scenario for the Brilliant Pebbles architecture.

3.1. HALO ALGORITHM DEFINITION

This section develops a model describing the **HALO** Algorithm in terms of objects and defines how these objects interact with one another. This model is defined with an Object Oriented structure which lends itself well to implementation in **COPES**. The section concludes with a definition of the heuristics of the algorithm.

3.1.1. Adaptive Link Problem

The **HALO** Algorithm considers a Brilliant Pebbles architecture consisting of a constellation (or constellations) of satellites in low earth orbit communicating with each other using laser links. The algorithm attempts to reduce the work of a routing algorithm by generating a ranked list of links ordered by the best to worst probability of successful transmission. The algorithm generates this optimally ordered list by applying a set of heuristics to the list of links such that in most cases the router would only have to consider a small set of these optimal links to make its routing decision. This is important in a laser based communication network where a large number of highly dynamic potential links exist.

Figure 2 shows a model of the data flow and objects used to implement both the **HALO** Algorithm and a simulation of a **BP** scenario. This model represents a single instance of the **HALO** algorithm running on one **BP**(referred to in this discussion as the *reference node*). In this phase we do not consider the router, consequently we are only concerned with the point of view of one pebble and how it orders its optimal set of links. Thus, Figure 2 does not consider any routing issues. In the rest of this section we develop the model shown in Figure 2 and describe the algorithm as a set of heuristics (or constraints), a set of objects, and the relation between them.

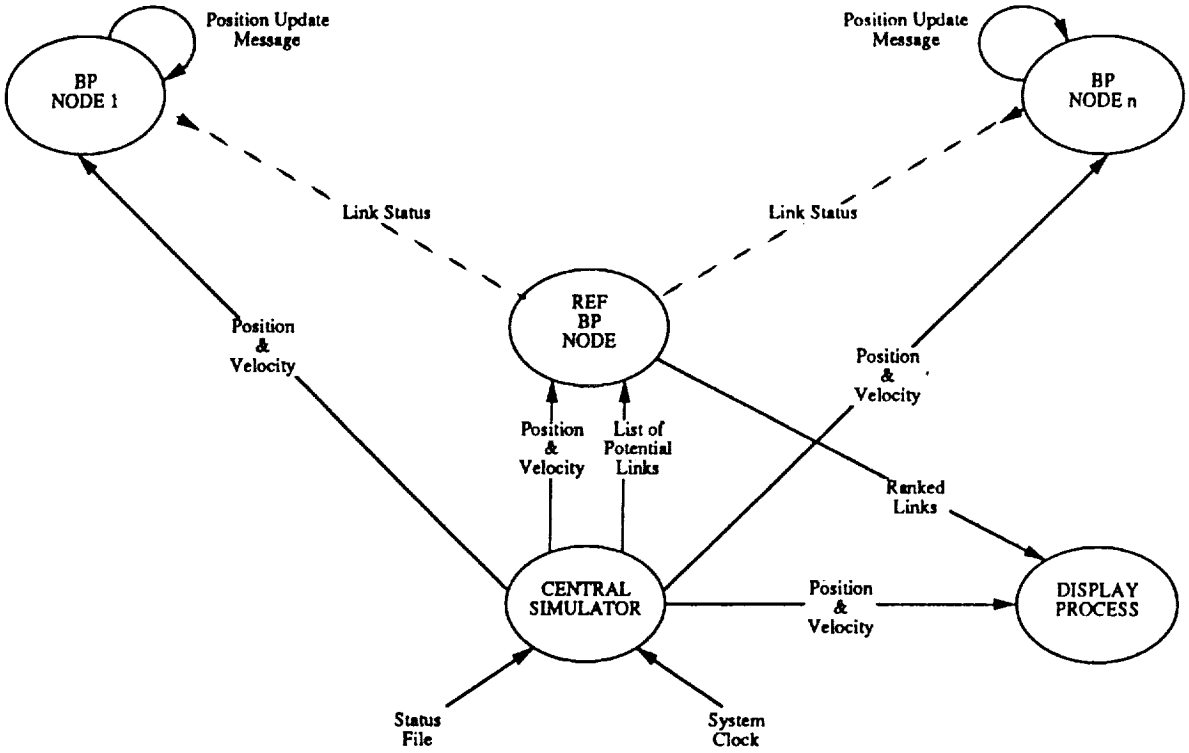

Figure 2 - HALO Data Flow

The data flow diagram in Figure 2 shows the main objects in the **HALO** Algorithm and the communications between these objects. The object **BP Node** describes parameters common to each **BP** in the constellation. It contains information such as the position of each satellite, its error term, and other instance information. This object is replicated many times within each **BP** which runs this algorithm (it is the reference node's view of other **BPs** in the network). The object **Ref Node BP** describes information unique to the reference node. It contains the ranked list of "links", some state information, and some information about the constellation. The object **Central Simulator** is not a physical object in the **BP** scenario, but contains data associated with the overall **HALO** algorithm and helps control the interactions between objects. It governs the operations of the simulation such as reading the orbital dynamics file, and starting and stopping the simulator. The object **Display Process** also is not a physical object in the **BP** scenario but is an entity which monitors the algorithm and simulator, and reports its progress for display and analysis. The lines with arrows show the communication between objects, the solid lines indicate actual messages being sent to the appropriate object(algorithmic communication, not to be confused with actual communication between physical **BPs**), and the dashed lines indicate an object viewing the data in other objects, e.g. **Ref Node BP** accesses link status from **BP** node 1.

The **HALO** Algorithm uses a set of heuristics which govern how the algorithm sorts the list of possible links. These heuristics are described below.

- LOS - This heuristic checks whether a **BP** is in LOS of the reference **BP**. For the purposes of the **COPES** implementation, this data is precomputed using an orbital dynamics package to model the satellite motion and visibility.
- Velocity - This heuristic checks whether the relative velocity between two **BPs** communicating with one another would cause a Doppler shift impairing the laser communications. An analysis of the need for this heuristic is described later in this section.
- Lasercomm Probability - This heuristic determines the probability of a successful communication between two **BPs**. It models the effects of pointing error, position uncertainty between pebbles, and other laser parameters.
- Position Error - This heuristic allows the position error to be corrected at a predetermined rate. Currently, the position error updates (which would normally be received from other **BPs**) are randomly scheduled with a period of

one satellite orbit.

These heuristics are imposed by the **HALO** algorithm and do not necessarily map into unique **COPES** constraints. Each heuristic defined above is developed further in the remainder of this section. In the next section we develop the **COPES** constraints which implement these heuristics.

3.1.2. Line Of Sight Heuristic

The Line Of Sight (LOS) heuristic determines whether a reference node can communicate with a given **BP**. This simply tests whether the specified **BP** is physically in LOS with the reference node. If it is not, then no further consideration is given to the **BP** as a potential link.

3.1.3. Velocity Heuristic

We now consider the requirements for a velocity constraint. The relative motion of two **BPs** may affect the communications between themselves due to the Doppler shift of the laserbeam. The maximum tolerable frequency shift of the laserbeam is on the order of 1 *nm* for the laser comm systems considered in the brilliant pebbles architecture. The Doppler shift of light between two bodies is defined by the following equation:

$$v' = (1 - \frac{u}{c})$$

u = relative velocity of BPs.

v = frequency of light source at rest.

v' = frequency of light due to u.

This equation is valid if $\frac{u}{c} \ll 1$.

In the brilliant pebbles simulations at typical satellite constellation altitudes the maximum relative velocity of **BPs** is about 7.5 *km/s*. Substituting the appropriate values into the above equation, the Doppler shift that the **BPs** see is approximately 20 *pm*. This is two orders of magnitude less than the laser design constraint of 1 *nm*. Thus, this constraint appears to be of limited concern in the initial analysis. There has been some discussion in the SDI community concerning a potential problem with link acquisition based on relative velocity. This remains an area of research.

3.1.4. Laser Probability Heuristic

The laser probability heuristic is a computation of the probability of successful reception of a laser transmission from the reference **BP** to a designated source **BP**. The probability is computed from a model of the laser communication link. This model is for a direct-detection receiver using multimode pulse position modulation (PPM) signaling and is currently being studied for the Brilliant Pebbles architecture.

The laser model defined for the **HALO** Algorithm has only one degree of freedom, the position error (described below). The model assumes fixed values for other parameters of the laser model. In addition controlling the transmitter beamwidth yields a better probability of successful communication over a wider range of distance between source and destination. The model assumes the beamwidth can vary from 2.5 *mRAD* to 25 *mRAD*. The model simulates this variance by keeping the following relationship:

$$\theta_b R = K$$

where

θ_b = receiver beamwidth.

R = receiver range.

K = constant.

The range of distances between the reference node and a designated node varies over the interval (200,4500) *km*. Thus, the laser model optimally sets its θ_b over this interval and then computes the probability of successful communication using the position uncertainty.

3.1.5. Position Error Heuristic

Each **BP** maintains a database of the current positions of other **BPs** in the network used for routing and link selection. As time passes, each **BP** calculates predicted position of the other **BP's**. Due to the relative infrequency of position updates, there is error associated with these predictions. This position update message has an inherent error associated with it as well.

In the adaptive link reconfiguration simulations, we model a position error as a growing sphere around the **BP** over time. Thus, we need a rate term to grow this sphere as the simulation progresses. The worst rate would result from the **BP** being at a less or greater orbit altitude than it is

supposed to be. This would cause the orbit period to be faster or slower, respectively, than a **BP** would predict it to be. Thus we will assume that the position update message has an error of a certain amount d which is in a direction greater or less than the actual orbit radius.

Assuming a spherical earth, the error will grow at a linear rate as computed below:

$$R' = R - d$$

$$v_e(t) = v(R-d) - v(R)$$

$$v(r) = \sqrt{\frac{\mu}{r}}$$

$$E_{orbit} = t\sqrt{\mu}\left(\frac{1}{\sqrt{R-d}} - \frac{1}{\sqrt{R}}\right)$$

where:

d is the error of the global positioning system.
R is the perceived radius of the BP orbit.
R' is the actual radius of the BP orbit.
$v_e(t)$ is the error velocity.
E_{orbit} is the error rate in m/s.
μ is the gravitation parameter in m^3/s^2

The adaptive link simulations assume the positioning system used in the **BP** architecture is accurate to $100\,m$. This causes the worst possible position error rate of $0.055\,m/s$ at an altitude of $550\,km$. This value is used in the simulations run. The simulator also assumes that the position updates occur at a rate of once per orbit.

3.2. COPES IMPLEMENTATION

The model of Figure 2 presents a set of objects which describe the **HALO** Algorithm. These objects are described as classes in **COPES**. Each class defines the state information of a particular object in a model. There may be multiple instances of a class such as the **node** object in the model which is duplicated for each physical node in the system. A constraint function describes the interactions between the defined objects. A constraint is bound to variables in a given class instance. A constraint "propagates" or "fires" when a variable in a class instance that the constraint is bound to changes. When the constraint "fires" it observes the state of the objects it is bound to and changes the states of these objects appropriately. A constraint may be bound to a variable in two ways. The first way is for the constraint to "fire" when the variable changes. The second way is for the constraint to ignore changes to a variable but access this variable when the constraint does "fire" from some other binding.

The **HALO** Algorithm is defined as a set of classes and constraints. The classes defined below represent the objects of Figure 2 and some additional classes required for the **COPES** shell and to support the simulator for the **HALO** algorithm. The classes are:

Ref Node : This class contains information unique to the reference node above what is necessary to describe a general node.

Node : This class contains information unique to each node.

Link : This class contains information about the laser link between the reference node and the node with which this link is associated.

File In : This class contains file status and descriptor information used by the central simulator.

Display : This class contains file descriptors and flags used by the display object.

GLOBAL : This class contains simulation parameters and flags to which every constraint has access.

In developing the algorithm, we define state transition diagrams which describe the threads of the overall algorithm flow. A sample of one of the state transition diagrams is shown for the central simulator (Figure 3). The Central Simulator is responsible for managing the simulation and reading the new position and velocity (p&v) parameters from the orbital dynamic file. It loads the p&v information into each node, waits for the current cycle to complete and then starts the next cycle. When the simulation is complete, the central simulator causes the **COPES** shell to terminate. Referring back to Figure 2, a Node performs three tasks. During the initialization cycle, it schedules a position update message to correct the position error term. During a normal cycle it receives and interprets position update messages and it responds to new position and velocity parameters. The reference node shown in Figure 2 is the node on which the software is considered to be running in the simulation. In a real system each node would be a reference node. The reference node manages the list of ranked links. As each node updates its link status parameters, the reference node updates the list of ranked links. When all links have been updated in the current cycle, the reference node

indicates it has an updated links list which starts the display process. The Display Process monitors the simulation then prints out simulation statistics and formats the ranked list for viewing with the ViewNet program at the end of each cycle.
A sample of this output is given later.

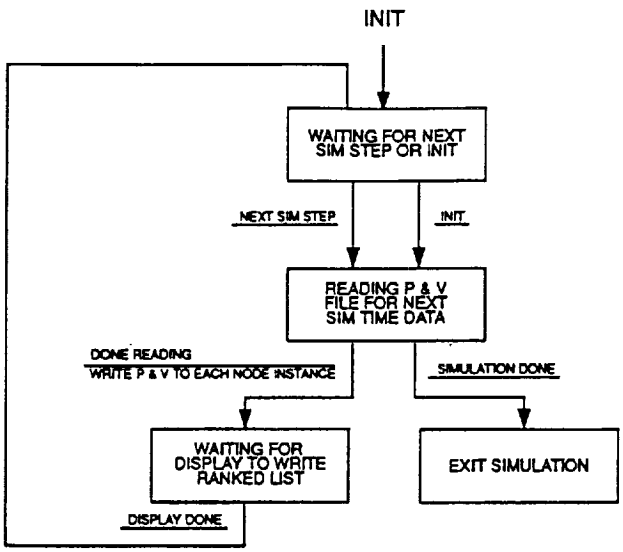

Figure 3 - Central Simulator STD

The constraints are derived from the heuristics (described in the previous section) coupled with the state transition diagrams. User defined constraints are detailed in the remaining part of this section. User constraints are defined using structured english PDL.

The first constraint is the **read_pos** constraint. This constraint performs much of the function of the Central Simulator object. It is responsible for controlling the simulations aspects of the **COPES** algorithm implementation, and reading the orbital dynamics file for the current computed satellite positions and LOS data.

This constraint has one instance and schedules itself to fire once each cycle. This constraint is bound to the instance of the class **file_in.** This class has a state variable, run_file_in, which the read_pos constraint schedules to change in the future. This allows the constraint to fire itself to run at the beginning of each simulation time cycle to read the current satellite position data. In addition to acting as the Central Simulator, the read_pos constraint represents a **BP** reference node computing the position, velocity, and LOS of each node in the constellation. It takes advantage of the discrete event scheduling capabilities of **COPES** to move the simulation, link ranking, and display through discrete phases. In this manner objects like the Display do not have to signal the Central Simulator when they are done. Their inactivity (lack of constraint propagation changes in the network) causes the removal of the next change from the schedule queue and constraint propagation continues in the next phase.

The next constraint is the **pos_update** constraint. This constraint is concerned with scheduling and receiving position update messages. It performs two functions. It initially schedules the random position update for each node. It then responds to the position update messages which cause the node to reset its position uncertainty. This constraint represents a reference node receiving a position update message from another node in the constellation. This message provides the actual position of the satellite which the reference node uses to reset its notion of that satellites position. A separate pos_update constraint is bound to each **node** instance in the **BP** scenario.

The next constraint is the **Node1** constraint. This constraint models the cumulative effect of the position error. It gets fired when a p&v recomputation event occurs (triggered by the Central Simulator) which causes the node to increase its position uncertainty. A separate instance of this constraint is bound to each class **node** instance.

The next constraint is the **comm_ber** constraint. This constraint fires when the position uncertainty parameter of a given node is modified. It then (if in LOS, meets the Doppler heuristic, and is within range) computes the bit error rate (BER) of successful laser communication. A short Structured English PDL is shown for this constraint is shown below as a design example.

```
comm_ber ()
{
  when new position uncertainty parameters for this node
   for each potential link
    if node in LOS and (Doppler and range thresholds valid) then
      compute new ber for successful tx (src to dest).
      store new ber in link.
      set link flag indicating to add/update link in ranked list.
    else if link currently in ranked list then
      set link flag indicating to remove link from ranked list.
    endif
   endfor
  endwhen
}
```

It sets this BER in the link instance for this node. A separate instance of this constraint is bound to each class **node1** instance.

The next constraint is the **rank_links** constraint. This constraint fires each time a node modifies its link quality parameters and then ranks all the links according to these parameters. When all links have been determined, the constraint sends the ranked list out to the display constraint. A separate instance of this constraint is bound to the each class **node.**

The last constraint is the **display** constraint. This constraint implements the display object. Its purpose is to take the ranked list of links from the reference node object and format it for display in ViewNet. Additionally, it outputs some statistics of the simulation for post analysis. A single instance of this constraint is bound to the class **Display Process.**

Finally, Harris developed constraint binding diagrams are produced for each constraint to show the dynamic bindings which link constraints to class variables. Figure 4 shows an example of a constraint binding diagram for the comm_ber constraint. This diagram is useful to understand how the constraints interact with the object instances. An instance of the comm_ber constraint is created for each **node** and **link** object as it is viewed from the reference node. An instance of the constraint fires when the position error of the node instance to which it is bound is modified. This causes the given **node** object to reset the concept of bit error rate for the link from the node to the reference node. The remaining variables are bound as access only and do not cause the comm_ber constraint to fire. An access only variable is indicated in the constraint binding diagram by placing an "A" on the end of the line linking the constraint to the variable.

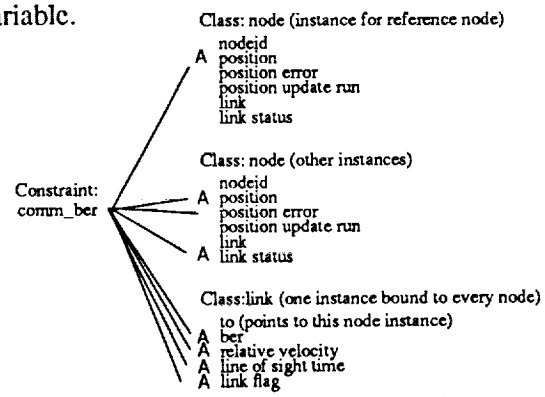

Figure 4 - Comm_ber Binding Diagram

The classes and constraints discussed in this section define the **COPES** model for the **HALO** Algorithm and simulator. In the next section we discuss the actual **BP** scenario used to test this Algorithm and the results of those simulations.

3.3. SIMULATION DATA FOR THE ADAPTIVE LINK ALGORITHM

This section presents the simulations run for the **HALO** Algorithm. The Adaptive Link simulator provides two types of output for analysis. The first is a visual display using the ViewNet tool developed at Harris. The second is a plot of the average ber rate of the top portion of the ranked list versus the ber of all possible links.

The Brilliant Pebble scenario used to test the algorithm is an unclassified network which is closer in size to a GPALS architecture. It consists of a single constellation of satellites at an altitude of 550 *km* and an inclination of 60°. The constellation contains 21 rings of 20 pebbles per ring. Only one constellation is used since the problem is not changed by multiple constellations and the implementation of the simulator is simplified. The simulation described above runs for a period of one earth day. This provides time for approximately 15 satellite orbits. The random position update messages are issued once per orbit.

The ViewNet graphical tool provides the capability to visualize the Adaptive Link algorithm in operation to gain an intuitive understanding of how it works. Figure 5 shows a snapshot in time of the ViewNet display. This figure shows the satellites in orbit around the earth, the reference node with the eight "best" links connected to the appropriate node. The actual ViewNet display is in color. Each node has a special color indicating its status as a potential link. In addition, the links are color coded from red to grey indicating their relative position in the list of ranked links. The laser probability model tends to select the closer links as opposed to the more distant links. However, it ranks extremely close links as less probable due to the fact that the position error becomes more significant at closer ranges. Observing the ViewNet display, as the links are reordered and displayed, this trend is apparent.

The second visualization of the simulation is a graph depicting the enhanced set of links available to a router versus the set of all possible. The set of all possible links is those links within LOS and within range. The ber is computed for each of these links and averaged. This plot is provided as a function of time. The **HALO** algorithm enhancement is shown by averaging the top 8 links in the ranked list of links. This average is plotted as a function of time also. The plot indicates that

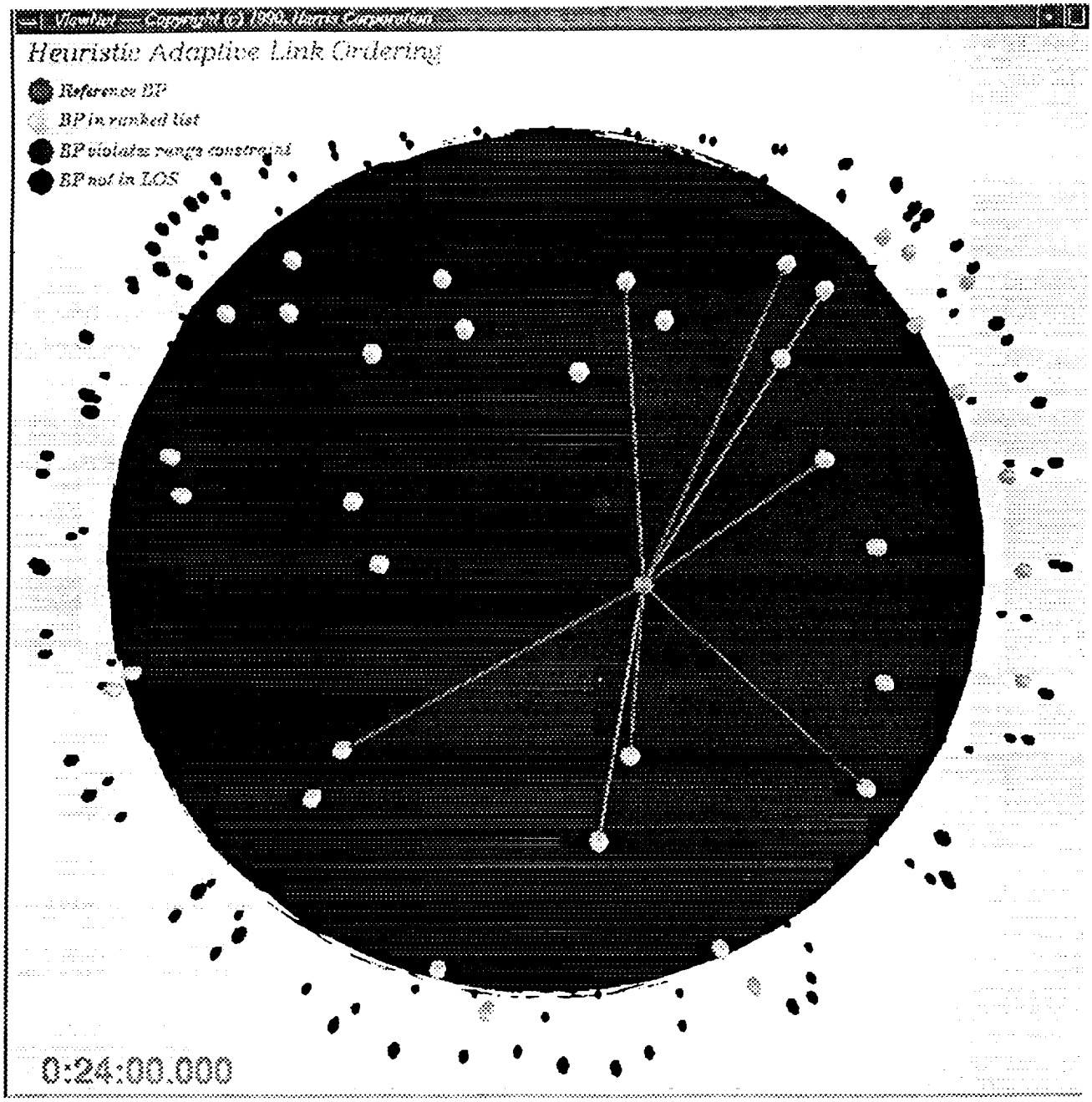

Figure 5 - Time Snapshot of Adaptive Link Viewnet Display

the top eight ranked links are an order of magnitude better probability of successful transmission over just any possible link. The plot is shown in Figure 6.

4. CONCLUSIONS

The previous section presented a successful simulation of a GPALS-based **BP** architecture using the Discrete Event Simulation capabilities of the **COPES** Shell. This simulation in **COPES** is very flexible and easily modifiable to address the **BP** network in its entirety, including any future architectural or procedural changes. The simulation of the network and results of the **HALO** algorithm were also interfaced with the Harris developed ViewNet graphics tool for network analysis on the Silicon Graphics Workstation. Also described was the **COPES** implementation of the **HALO** algorithm. A graphical analysis showed that the algorithm generates a reduced, improved, and ordered set of links for further use by a routing algorithm. The benefit of a lower bit error rate on the selected link is a reduction in the power requirement for the communications.

Given the flexibility of a constraint approach to the **HALO** algorithm written in **COPES**, changes in constraints can easily be made to, for instance, emphasize links which are more distant. This is an area for future research.

5. FUTURE RESEARCH

For the **ALR** program, the size of **BP** constellation which must be considered by a routing algorithm has been reduced to one closer to the GPALS architecture. We use an orbital dynamics program to remove all nodes which are never seen by the reference pebble we are studying. Finally, using a set of constraints defined above, the set of potential links is reduced and ordered by how well each meets the constraints. The next step is to develop an intelligent routing algorithm which would use this ordered list of potential "next hops" to choose a link or links for a particular message. The major advantages to this approach are that the set of potential links has been reduced significantly prior to the running of the routing algorithm, and the probability of successful transmission is higher. An intelligent routing algorithm might also contain heuristics to allow it to consider the first n potential links based on the situation, since the links are already sorted by how well they satisfy a set of link constraints.

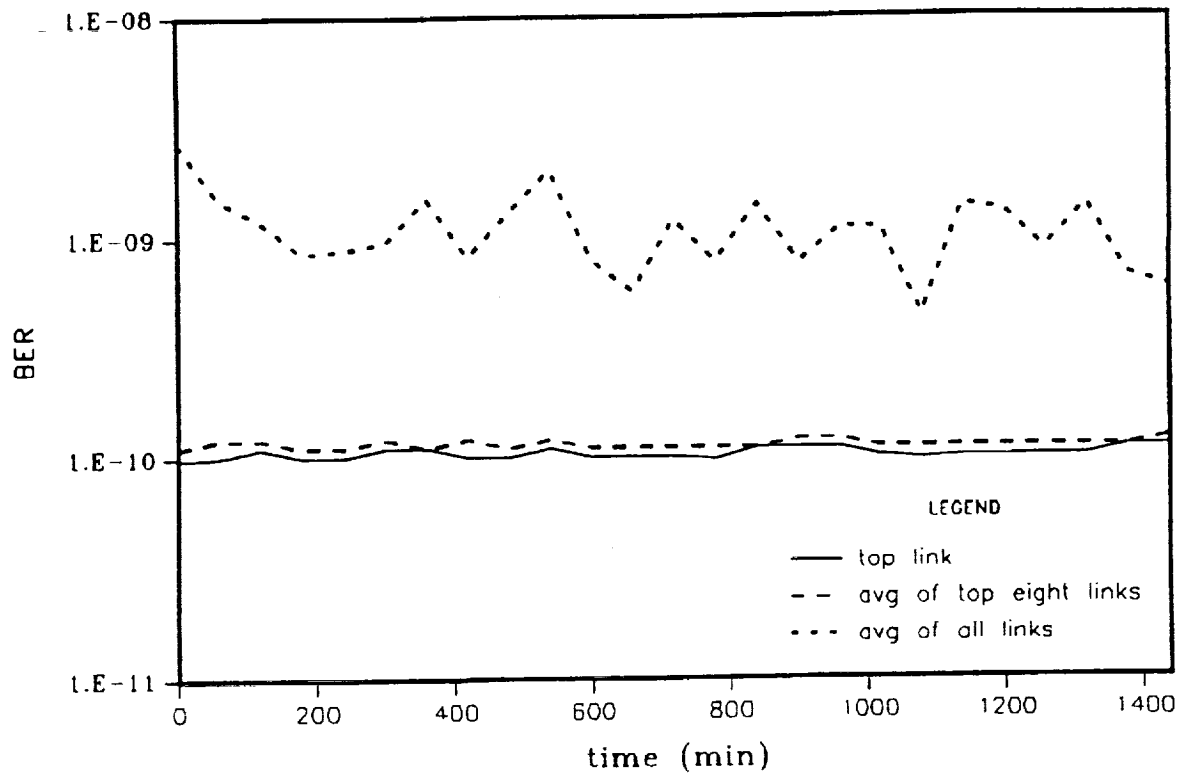

Figure 6 - BER of All Links vs Top Eight Ranked Links

6. REFERENCES

Barr, A. and Feignbaum, E. (eds.). (1982). *The Handbook of Artificial Intelligence,* HeuristicTech Press, Stanford, Cal.

Cameron, J. (1986). "An overview of JSD", *IEEE Transactions on S/W Engineering,* Vol SE-12, No. 2 (Feb.).

Chen, P. 1976. "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transaction on Database Systems,* Vol 1, No. 1 (Mar.), pp. 9-36.

Clancey, W. (1986). The Science and Engineering of Qualitative Models, *Proceedings of AAAI-86, Science Track.*

Crone, M. S., Hall, D. (1985). "Comments on the Procurement and Development of Expert Systems", *Proc. Expert Systems in Government Symposium, IEEE Computer Society.*

Crone, M. S., Julich P., Dash E., Wavering W. (1987). "Expert System Technology for the Space Station Communications and Tracking System, *Proceedings of the SPIE, Space Station Automation III, vol 851.*

Crone, M. S., Julich P. (1990). "Dynamic Network Reconfiguration Using Constraint Propagation" *Proceedings of the Society for Computer Simulation, Multiconference on Simulation and Artificial Intelligence, vol 21.*

Crone, M. S., Julich P. (1991). "Distributed Intelligent Network Management for the SDI Ground Network" *Proceedings of MILCOM*

Davis, R. (1983). "Diagnosis Based on Structure and Function", *Proceedings AAAI Conference.*

Davis, E. (1987). "Constraint Propagation with Interval Labels", Artificial Intelligence, vol 32., pp. 281-331.

Fox, M. (1983). *Constraint-Directed Search: Case Study of Job-Shop Scheduling,* Ph.D Thesis, Computer Science Department, Carnegie-Mellon University.

Gevarter, W.B. (1983). "Expert Systems: Limited but Powerful", *IEEE Spectrum* (Aug.), pp 39-45.

Gupta, Anoop. (1983). "Parallelism in Production Systems", Ph.D Thesis, Computer Science Department, Carnegie-Mellon University.

Hayes-Roth F., Waterman D., Lenat D. (eds.) 1983. *Building Expert Systems,* Addison-Wesley.

McClelland, J., Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing,* MIT Press

McQuillan, J., Richer, I., Rosen, E. May (1980). "The New Routing Algorithm for the Arpanet," *IEEE Transactions on Communications, vol.com-28, No. 5*

Press, W., et. al. (1986). *Numerical Recipes,* Cambridge University Press.

Rich, E. (1983). *Artificial Intelligence,* McGraw-Hill, Inc.

Simonian, R., Crone, M. May (1989). "True Object-Oriented Programming in Ada" *Signal Magazine.*

Stefik, M., Bobrow, D. (1986). "Object-Oriented Programming: Themes and Variations", *The AI Magazine.*

Waltz, D. (1975). "Understanding Line Drawings of Scenes with Shadows," in *The Psychology of Computer Vision,* edited by Patrick Winston, McGraw-Hill Book Co, N.Y.

Acknowledgements