2006-12-18

Multi-Objective Routing Optimization for Multiple Level Priority and Preemption in Multi-Tiered Networks

Jason Z. Farmer
Worcester Polytechnic Institute

Multi-Objective Routing Optimization for Multiple Level Priority and Preemption in Multi-Tiered Networks

By

Jason Zane Farmer

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

By

_____

October 19th, 2006

APPROVED:

_____

Prof. David Cyganski, Major Advisor

_____

Prof. Brian King

_____

Prof. Wenjing Lou

Abstract

This thesis explores techniques for improving the Quality of Service (QoS) driven routing of IP traffic in a Network Centric Military Communications System within an HC3 (High Capacity Communications Capability) tiered topology. In this specialized network various routing algorithms, including traditional, QoS-constrained search-based, and heuristic approaches, were evaluated. An automatic system for the probabilistic generation of appropriate networks and traffic was created for Monte Carlo simulation of the systems and testing of the various routing algorithms. A new algorithm we propose, based upon a hiercharical decomposition of routes about the minimum distance routes, is described and tested. These results provide both insight into this problem and demonstrate the possibility of highly optimized solutions without exhaustive search.

Contents

List of Figures

List of Tables

1.0 Introduction

This thesis will explore techniques for improving the Quality of Service (QoS) driven routing of IP traffic in a Network Centric Military Communications System based upon an HC3 (High Capacity Communications Capability) tiered topology depicted in Figure 1.1 below. The convergence layer between traditional Internet infrastructure networks and the mobile ad hoc network is the subject of primary interest in this research.

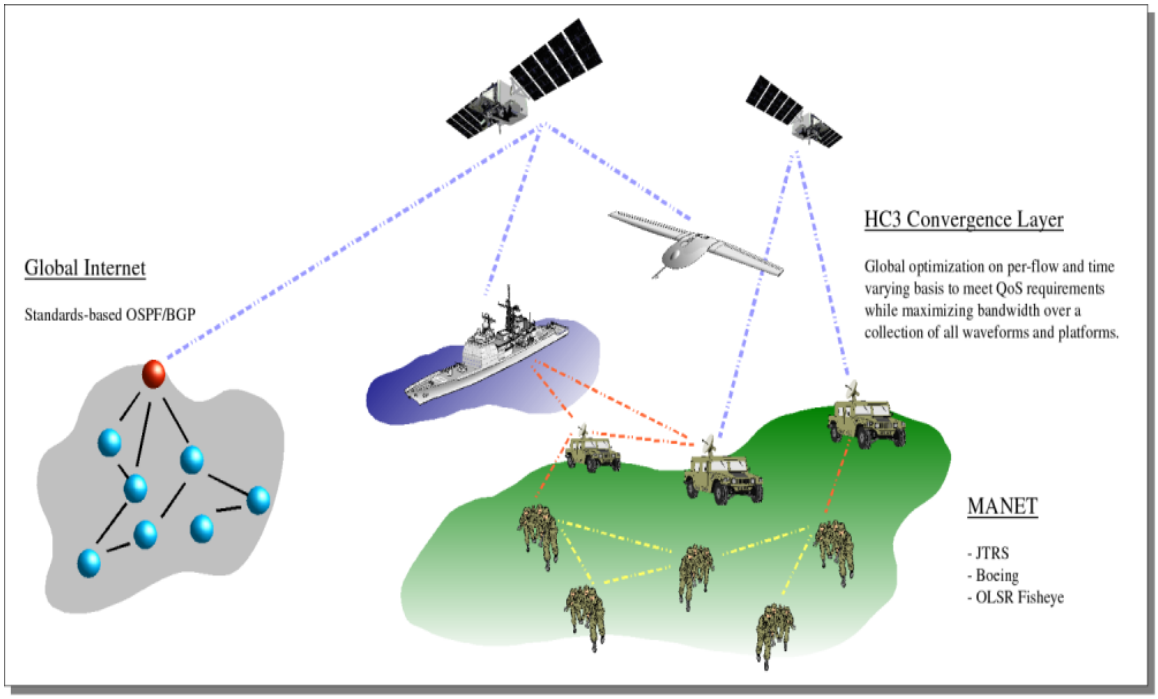


Figure 1.1 - HC3 Tiered Topology Illustration

The QoS requirements demanded by this convergence layer network application include call admission control, multiple levels of precedence and preemption, and strict bandwidth and delay bounds. A mixture of traffic will be handled through the use of Class Based Queuing (CBQ), which includes best effort TCP and QoS guaranteed Constant Bit Rate (CBR) UDP streams.

It is well known that QoS optimized routing is a computationally intractable problem and that classic infrastructure routing algorithms deliver very poor performance in MANET and tiered routing structures in which alternative routes with widely varying path costs and QoS behaviors are represented. Hence, the stated goal is to find new means, based upon multi-objective optimization, that deliver useful, timely, though necessarily sub-optimal solutions to this important problem.

1.1 Overall Project Approach and Contributions of This Thesis

The Raytheon Company sponsored project which spawned this thesis was divided into two components matched to the strengths of the two university teams engaged by the sponsor. The WPI team and this thesis work were to provide means to generate network problem models based upon Convergence Layer topologies and QoS requests with their information. The WPI team would also provide implementations of traditional routing techniques and state of the art variations, the performance of which would be compared with the North Carolina A&T's multi-objective reactive tabu search method.

The WPI team was responsible for development of the test bed and performance assessment system as well as network simulations used to prove the feasibility of solutions and to demonstrate dynamic routing techniques. The traditional routing algorithm's shortcomings will be examined, state of the art techniques implemented, and finally all solutions will be evaluated. NCA&T were responsible for the development, implementation, and details surrounding all of the search-based algorithms. Other WPI project participants identified the network simulator and added extensions including MPLS support for constraint based routing, developed the simulator frameworks, Simple Routing Problem Description Protocol (SRPDP), the graphical network viewer, and put in place the groundwork of the test bed environment. I continued their work with the implementation of all of the shortest path algorithms tested in this document, developed the blocking metric with Professor Cyganski, and implemented the evaluation tools necessary to compare the various algorithms. In addition I created a feasibility tester for the routing solutions, developed the distributed testing scripts for large-scale solution finding, and served as a liaison with the NCA&T research assistant. Finally I created the

implementation and techniques used in WPIDORA as an adaptation of the approach used by the DORA algorithm.

1.2 Summary of Outcomes

As might be expected, the outcomes of this project confirmed the general wisdom regarding the non-optimality of current QoS agnostic routing practices commonly employed throughout the infrastructure of the Global Internet and in experimental ad hoc networks and related convergence layer systems. Application of a high-level multi-objective optimization (Reactive Tabu Search) revealed significant gains in high priority connection establishment. For example, consider the outcomes of a test involving a 24 node network and 48 simultaneously-attempted connections at four levels of priority. For any problem instance, that is, at any given time of the operation of a network, an average of approximately 1.5 more priority 1 connections and approximately 2 more priority 2 connections were maintained. This goal was obtained but with an increased computational cost of over 5,000 times that of CRLSP (Constrained Routing-Label Switched Paths).

However, unexpectedly, we discovered that almost all of this performance gain could be obtained by applying a level of global optimization significantly less complex than noted above, with very modest computational loads. This result prompted the team to explore the effectiveness of various tiers of optimization and to determine the critical tier at which nearly all routing effectiveness is achieved. The important result derived from this research is that, for HC3 problems, there is an optimization strategy that is nearly optimal and yet can be computed at sufficient speed so as to not significantly degrade routing execution and to allow deployment with today's computational technology.

2.0 The Routing Problem

2.1 Infrastructure Network

There are two major architectures that comprise the HC3 network. The first of these is an infrastructure network which would be typical of many networks, and can be likened to most any hardwired network. In an infrastructure network nodes are statically located and the links are structurally determined, and do not change often, if at all. Each node is connected to a series of other nodes via switches or routers and a network, and once assembled stays static for long stretches of time. Figure 2.1 is representative of a small network where there are a limited number of interconnections, with generally static links.

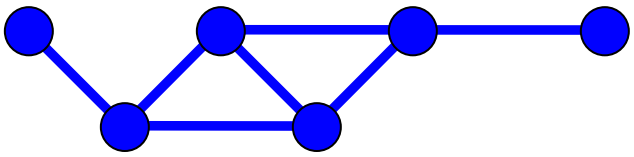


Figure 2.1 - Simple Infrastructure Network

The traffic on one link does not affect traffic on other links directly. That is to say, the presence of traffic on one link does not preclude the presence of traffic on a second link in the vicinity. In addition, links can operate as separate collision domains for the purposes of forwarding best-effort traffic. Best-effort traffic on this type of network can be routed very efficiently and effectively by using Open Shortest Path First (OSPF) routing [MOY98], in which a given node discovers neighbor nodes using a "hello" packet. The nodes can then use "Link-State Advertisements" to notify neighbors of their condition, which allows the use of Dijkstra's algorithm to determine the shortest packet routes, where "short" is measured with respect to some predetermined metric such as link delay.

Infrastructure networks may, in the case of RF links, use several non-interfering communication channels to connect different node pairs, or even the same node pairs. A given communication channel is often referred to by the jargon "waveform" and utilization of multiple waveforms implies the case of multiple non-interfering communication channels.

2.2 Mobile Ad hoc Network (MANET)

A mobile ad hoc network (MANET) is opposite in many respects to an infrastructure network, and is comprised of loosely organized nodes which arbitrarily enter, exit, and move around the network dynamically. These arbitrary changes cause rapid, near random changes to the network topology which must be addressed in the routing considerations. Figure 2.2 illustrates how nodes can even move from one network to another.

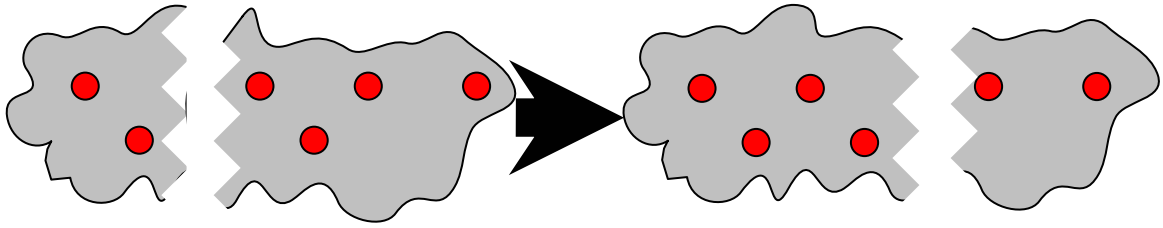


Figure 2.2 - MANET Node Movement

There are additional complexities that are encountered if the network should split, or additional networks join via node movement. Some examples of routing algorithms include ad hoc on demand distance vector (AODV), destination-sequenced distance vectoring (DSDV) [PB94], and temporally-ordered routing algorithms (TORA) [PC97].

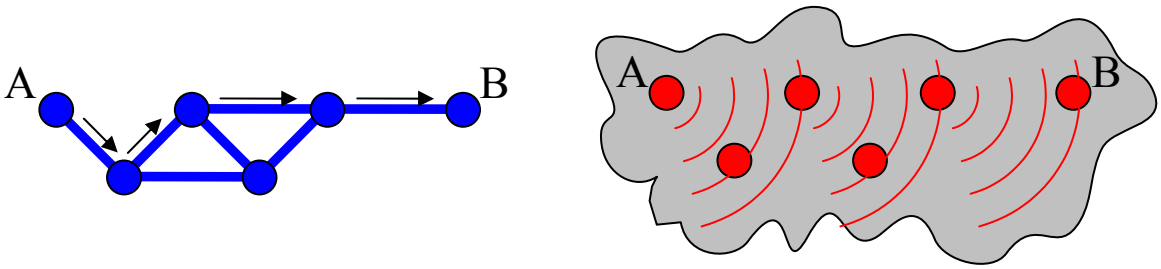


Figure 2.3 - Routing in Infrastructure Network and MANET

While routing is generally considered a solved problem for infrastructure topologies supporting best effort traffic, in the form of OSPF, this is not the case for MANETs. In the example shown above in Figure 2.3 the route from point A to B is via three intermediate nodes. The traffic affects only those links and nodes that are in the route. The second example on the right shows the same link in a MANET in which the signal is rebroadcast only twice, while all nodes in the network are affected by the traffic with regards to lost potential bandwidth utilization. This shows that a given link will also have an influence on the bandwidth in the physical region, creating obstructions and reducing the bandwidth on nearby links.

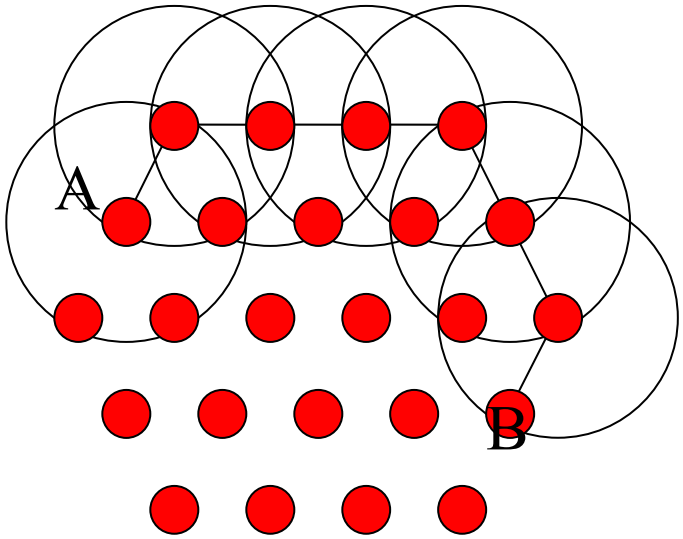


Figure 2.4 - MANET Collision Domain

The collision domain is also determined by geographical range rather than network links in a MANET. Figure 2.4 above represents a MANET with spatial division multiplexing of the collision domain. This means that the nodes transmit with a lower power, increasing hop count, but decreasing the number of nodes subject to a given collision domain. In the above example each node participates in up to 7 collision domains, including its own. MANETs also can employ different waveforms for different links having radically different data rates, latencies, packet loss rates, and maximum ranges.

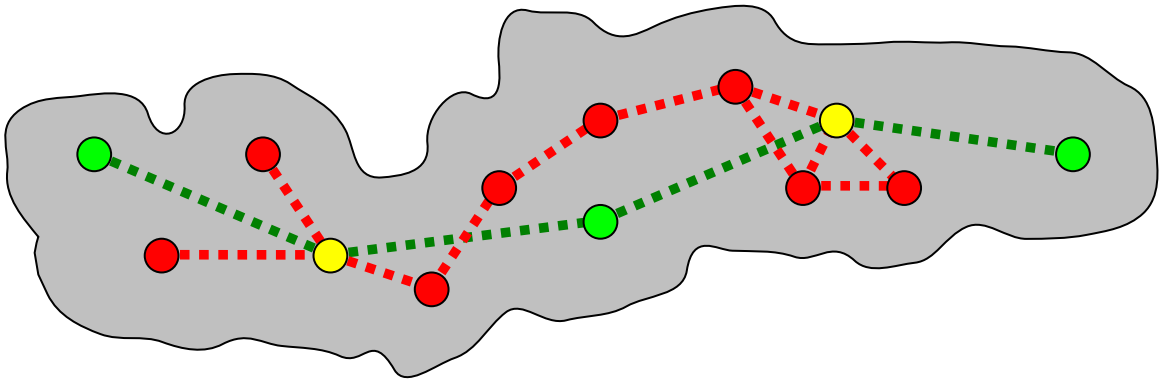


Figure 2.5 - MANET with Two Waveforms

Using multiple waveforms as shown in Figure 2.5 allows simultaneous transmission on different waveforms. In this figure the nodes in red are using waveform 1, green waveform 2, and yellow can use either waveform. This would allow for two end to end communications to happen without interference and hence a reduction in link bandwidths of geographically similar links. With two different waveforms, such as the example above, two different connections could be honored through the central network simultaneously but this freedom brings with it a much greater complexity in the routing process.

2.3 Hybrid Network (HC3 Convergence Layer)

The hybrid network is a mixture of the two preceding architectures. The HC3 network comprises the size, mobility, and highly connected properties of a MANET, but will enjoy time intervals without change in the topology intermediate to that of infrastructure networks and MANETs.

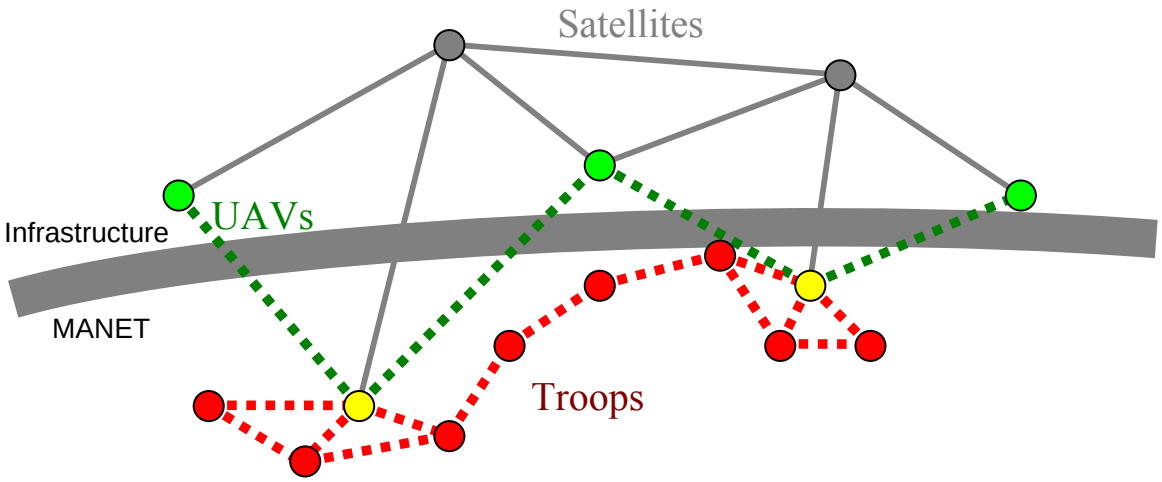


Figure 2.6 - HC3 Hybrid Network

Figure 2.6 above illustrates the mixed nature of the HC3 network. The Satellites and UAVs are infrastructure nodes in nature, while the deployed troops are representative of a MANET. The links in the HC3 network will often process higher bandwidth and the number of nodes generally will be smaller than in a typical full MANET. The general topology of the HC3 network will typically route the majority of traffic through a handful of key nodes, shown in yellow above, which will act as infrastructure gateways between the MANET style subnets and the Infrastructure networks. There will also be various waveforms used, each associated with specific platforms, including satellites in GEO and LEO orbits, unmanned autonomous airborne platforms, ground force communications, and many other possibilities.

2.4 The Routing Problem

The HC3 network topology together with demands for precedence-based access and QoS delivery requires **dynamic routing, handled on a per flow basis**. This will allow the best utilization of network resources for the highest satisfaction of user requests consistent with QoS requirements. To demonstrate this point we will examine the outcome of applying the Open Shortest Path First (OSPF) algorithm commonly used for infrastructure networks.

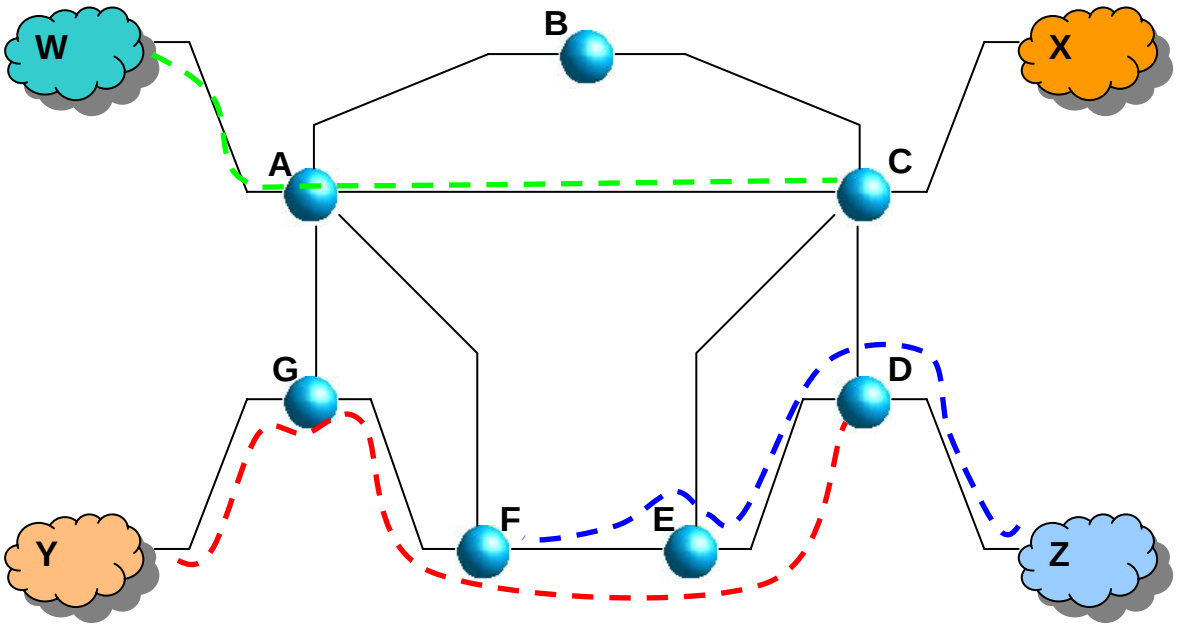


Figure 2.7 - Static Shortest Paths

OSPF will solve for the shortest paths for every pair of nodes, a subset of which are shown above in Figure 2.7, as part of a network routing table establishment process. Figure 2.7 illustrates the shortest path, using hop count as a metric, from subnet Y to node D, subnet Z to node F, and subnet W to node C. A connection that is requested from subnet Y to node D would be routed via nodes G, F, and E as per the route that was already established. This is shown in Figure 2.8 supposing that the request was accompanied by a fixed 0.7 Megabit per second (Mbps) bandwidth requirement as part of its QoS demand.

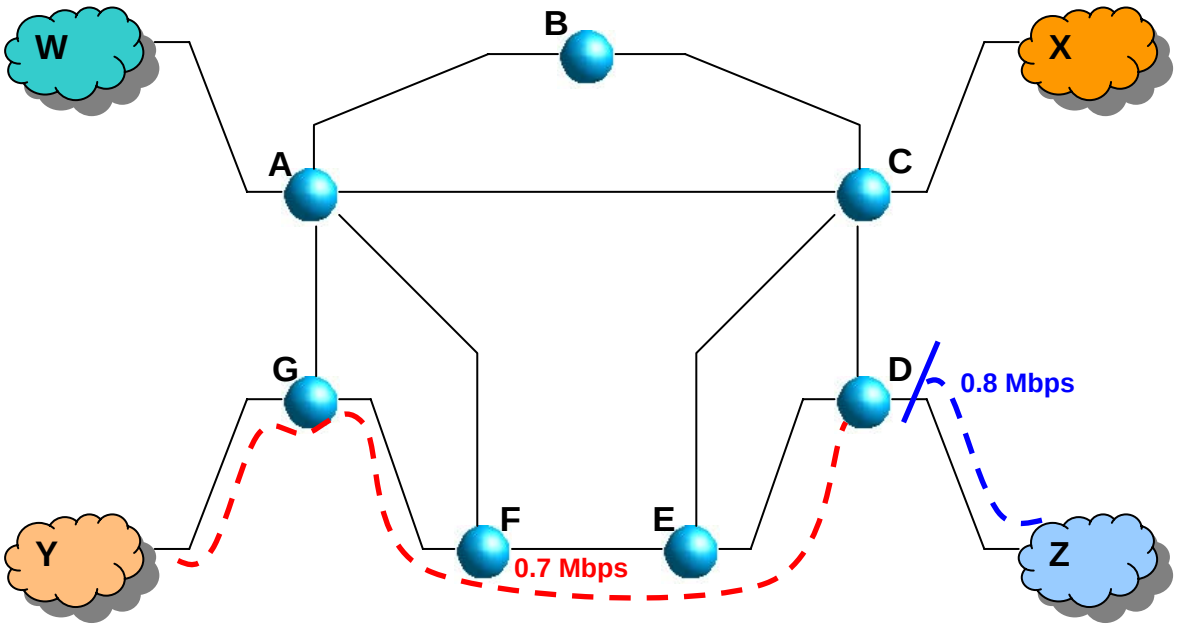


Figure 2.8 - OSPF Routing with Denied Connection

After the establishment of the Y-D connection with a 0.7 Mbps guarantee, a second connection request from subnet Z to node F is made for 0.8 Mbps. Since the links only have 1 Mbps bandwidth there is only 0.3 Mbps bandwidth available through links F-E and E-D given the OSPF route established earlier. This connection from Z to F would attempt to reserve the route that lead through nodes D and E, but there is not enough bandwidth available for the 0.8 Mbps connection, so it fails. If there were a dynamic routing algorithm at work it could rapidly re-route the existing connections to accommodate the new requests being made given that an alternative exists. Figure 2.9 below depicts such a solution.

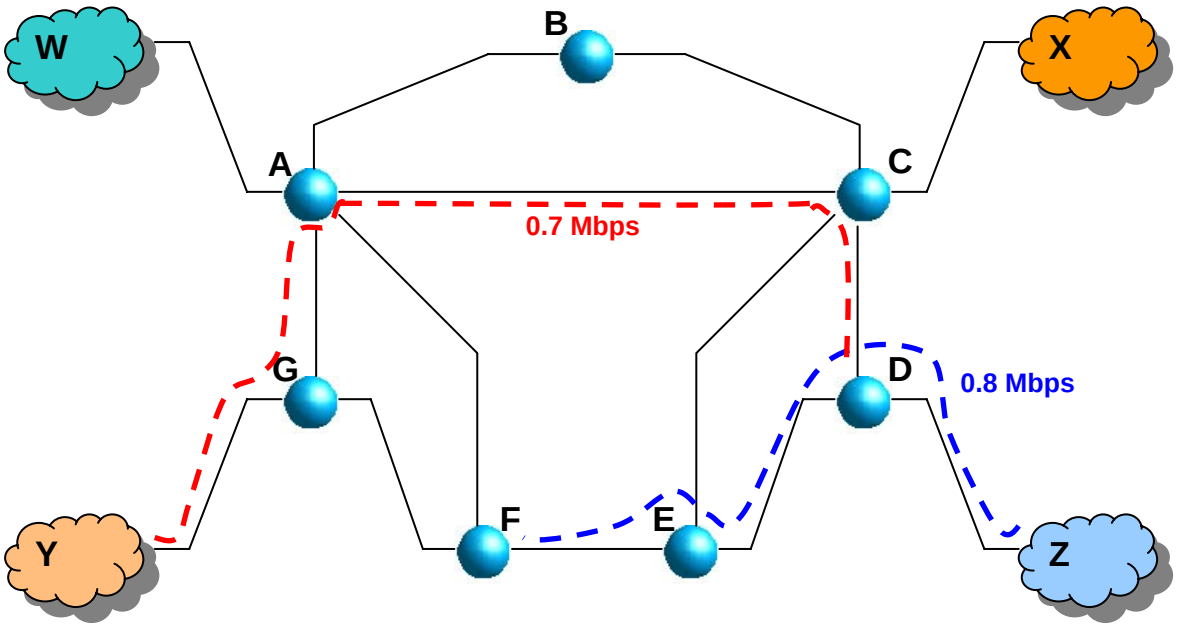


Figure 2.9 - Dynamically Routed Network

With the receipt of a Z-F connection request, the existing Y-D connection was moved to allow the bandwidth of the F-E link to be utilized for the second connection. This solution yielded two routes which maintained the low hop-count of the Y-D connection by moving it through a different area of the network. If a third connection was requested the network might again be rapidly re-routed as necessary. Furthermore, precedence of requests must be respected in a multi-level priority network. Connection requests with a higher priority would be granted network bandwidth preferentially, with lower priority connections routed around them as possible and otherwise terminated, in other words, pre-empted.

A MANET routing algorithm capable of handling waveform diversity would appear on the surface to accommodate some level of infrastructure networking in a small network and hence would be applicable to the Hybrid convergence layer network. But MANET algorithms are well known to be inefficient for infrastructure networks, and a transition must be made at some point as we approach the level of large-scale infrastructure routing domains.

Since a Hybrid network must find the best route in this time-varying subnetwork for packets between the infrastructure and highly variable mobile ad hoc networks, it needs quick convergence of its ad hoc-like routing.

2.5 Quality of Service

When connection requests have one or more attributes of Quality of Service (QoS) that must be satisfied, the usual, fully local, forwarding schemes based upon a single pre-computation of shortest paths, such as provided by OSPF, are no longer viable. At one extreme, this is easily visualized: In highly multiply-connected networks, minimizing "hop counts" as a metric for route pre-computation is not viable because it leads to routing most data through the network's center, thereby congesting any routers at the middle and compromising most notions of QoS (low drop rate, fixed bandwidth, low delay, etc.) Thus, it is obvious that, in contradiction to the usual goals of an OSPF or similar routing algorithm, sometimes routing packets via longer paths may increase throughput.

Of course, such a policy must be within the bounds of satisfying absolute QoS restraints such as those on minimum guaranteed bandwidth and maximum allowed delay. Furthermore, an element of QoS that enters in the context of military networks is the need to respect the priority of connection requests. In fact, there may be multiple levels of priority, and pre-emption of existing connections is appropriate if required to establish a new higher priority connection. Thus, the necessary routing algorithm must be stateful, that is, it must maintain the identity and QoS attributes of committed connections and be able to choose routes with respect to the remaining resources, pre-empt previous commitments when required by priority, and even re-route committed connections to reorganize the use of resources for the purpose of increasing the number of admitted connections.

In summary, we need a routing algorithm that will

- quickly converge after link state changes

- maximize global network throughput through a global, stateful optimization
- satisfy diverse and competing QoS requirements
- guarantee QoS commitments within the context of priority based pre-emption
- fairly share available bandwidth among non-priority users
- interface with existing IP networks
- dynamically re-route existing connections when necessary to increase admissions

Because of the several levels of competing elements of optimization and constraint, the problem can only be optimally solved via some form of global and multi-objective optimization. Unfortunately, it is well known that such problems are almost always computationally intractable, being of a complexity level belonging to the class NP-complete or harder. That is, the computational requirements for optimal solution grow at least exponentially with problem size, and these optimization methods offer no possibility of solution in real-time for problems of the size that comprise useful instantiations of HC3 convergence layer sub-networks [TGP01].

2.6 Optimization Requirements

A solution to a network problem consists of a series of paths that connect the source and destination node and meet the requirements set forth by the QoS constraints. The optimization that is performed must strive for multiple objectives which include that higher precedence connections are satisfied first, and if multiple paths are available choose such that bandwidth utilization is maximized across the entire network. This is in addition to the requirements that a path must fall within the given delay bounds, and have at least as much bandwidth as requested. All paths must be feasible, meaning that the utilized bandwidth of a link or node cannot exceed the total available bandwidth for that link or node.

For the purposes of mathematical optimization the above precedence and QoS rules can be represented as the equation shown below in Figure 2.10.

$$F = \sum_{m=1..M} a_m \sum_{c_k \in \{P_m \cap QoS=1\}} Q(c_k, p_k) +$$
$$a_{M+1} \sum_{c_k \in \{QoS=0\}} (FR(p_k) + BR(p_k)) + a_{M+2} \frac{(\sum_{c_k \in \{QoS=0\}} FR(p_k) + BR(p_k))^2}{\sum_{c_k \in \{QoS=0\}} (FR(p_k) + BR(p_k))^2}$$
$$a_1 \gg a_2 \gg \cdots \gg a_{M+1} \approx a_{M+2}$$

Figure 2.10 - Optimization Equation

The above equation contains the information necessary to represent all of the relevant, although simplified, aspects of the QoS connections in this context. This equation is a simplification of a multi-objective problem of mixed integer-continuous type, but will suffice for the current project.

For purposes of brevity, arrays of nodes, links, connection requests and paths will be identified *n*, *l*, *c*, *p* and specific elements of these arrays denoted by n_i, l_j, c_k, p_k. Let p_k be a path linking the source and destination associated with connection request c_k. The goal is to find an optimum set of paths with source and destination nodes that match those requested in c_1, ..., c_N such that the above function in Figure 2.10 is maximized. The weights, a_1, …, a_{M+2} must be selected such that each dominates the value of higher index weights. This is what accounts for a priority one connection having greater influence than several priority two connections, etc.

The function $Q(c_k, p_k)$ is 1 if:

1. The forward and backward rates requested by c_k can be reserved on the path p_k given the reservations made on each link in p_k which is shared by another path,
2. and likewise if these bandwidth requests are not in conflict with the bandwidth that can be provided by a node through which these links pass.
3. If the sum total delay imposed by each link in the path and by each node in the path does not exceed the delay requested by the connection request.

The function *Q* is 0 otherwise [NACP06].

Continuing, *M* is the largest precedence index that appears among the connection precedences. $FR(p_k)$ and $BR(p_k)$ are the forward and reverse bandwidths that have been

allotted on the k^{th} path for use by the k^{th} connection (for which there is no lower limit in the case of the best effort connections that make no explicit QoS requests.) The last two terms in the above equation represent the total bandwidth allotted to each best effort connection and the fairness with which it is distributed among them [NACP06].

It was the responsibility of the NCA&T team to formulate and implement optimization algorithms that find local optima of this function, with the intent of finding the global optimization or at least a local optimum near to its optimality.

2.7 Our Approach

Another worthy goal is to explore the efficacy of suboptimal approximations of full global and multi-objective optimization. Because of the inherent complexity of such problems, there is little hope that this or any other investigation will ever realize true optimality in real time. However, it is not generally understood how far simple solutions are from optimality. Likewise, previous research in multi-objective optimization has yielded suboptimal solution processes that provided adjustable optimization algorithm complexity – hence the question arises as to whether there is a computationally feasible solution that comes sufficiently close to full optimization as to make this HC3 QoS routing problem accessible. A major component of the project effort that took place at WPI and which is the subject of this thesis, is the creation of the tools necessary to perform this evaluation within the context of HC3 networks and solution methods that bridge the gap between full multi-objective optimization and simple shortest path algorithms.

The elements to our approach to answer this question are outlined here:

- Define the problem such that multi-objective optimization can be applied to the convergence layer routing problem.
- Seek a centralized and omniscient solution.
    - Solution obtained assuming all information about all nodes, connectivity, and present requests, then information is distributed to nodes for implementation.
    - Problems related to timely gathering of this information or dynamics of passing new requests were not considered.
- Compare results with existing routing methods (OSPF, DiffServ, IntServ).
- Establish exact bounds for smaller problems to enable absolute benchmarking of new solutions.
- Seek new distributed and dynamic routing methods.
- Build a platform for automated large scale testing of convergence layer problems.
- Evaluate newly proposed solutions from the literature.

3.0 Global Multi-Objective Optimization

In this section we will outline briefly the various forms of optimization that were utilized or investigated in this research.

3.1 Branch and Bound

The branch and bound algorithm is capable of finding the true optimum solution of a routing problem. On beginning this project our intention was to construct a branch and bound solution for a subset of the network problems which would stand as an absolute measure of the optimality of other solutions. To do this, a branch and bound routing solver was constructed by NCA&T which operates by selecting every possible first link in a route in the network and then selecting all choices for a next link until all feasible solutions have been tested. As an algorithmic speed-up, partially constructed paths are compared to a previously found best case situation. If the partially constructed path is incapable of exceeding the optimization previously found, the branch can be abandoned and other branches explored.

This nearly exhaustive procedure is guaranteed to find a best solution – but is computationally infeasible for large problems. To test if it was feasible for small problems a branch and bound implementation for network problems was constructed and applied to hundreds of route optimization problems. After a long weekend it was discovered that, in all but one case, it did not complete a solution within the cutoff time which had been set to 36 hours for each problem. Thus, unfortunately, there was no possibility of using branch and bound solutions as a baseline for comparison for any significant number of cases. Of the 100 problems attempted only one problem finished, and it required only 192 minutes. This particular problem was found to be a degenerate case, where the generated network topology and generated connection requests quickly

filled common links in the network and immediately starved all later connections of bandwidth.

3.2 Search Based Algorithms

The following subsections review the work in search based algorithms performed by North Carolina A&T. While implementation of these algorithms was outside the scope of this thesis, the evaluation of these algorithms was central to this work.

3.2.1 Local versus Global Search

While a full global optimization is not computationally feasible, one can in general implement an optimization procedure which includes global measures and avoids trapping in some local optima. To sharpen this notion, a distinction between local and global searches is necessary. A local search finds the optimal solution for a neighborhood of the solution space. Local optima are easily found based upon a gradient-based search in continuous parameter optimization problems and by small variations of route paths in routing algorithms. A global search yields closer to the optimum solutions for the entire solution space by using local search within neighborhoods while introducing mechanisms to traverse neighborhoods. The most difficult aspect of global search, and the most significant factor in limiting its performance, relates to the escape from the basin of a local optimum. The search and escape mechanisms usually depend upon meta-heuristics that are based upon some demonstrable aspect of the problem or are based upon experimentally confirmed ad hoc strategies.

3.2.2 Tabu Search

Tabu Search (TS) is an example of a meta-heuristic, a general recipe for coming up with good, but not necessarily optimal, solutions to problems that are generally too complex to solve with exact algorithms. TS searches for a solution across the entire space of feasible solutions (a global solution). It does this by using a local search (LS) strategy that searches sets of closely related problems for local optima. “The basic principle of TS is to escape LS whenever it encounters a local optimum by allowing non-improving moves;

cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search …" [Gen03]. In implementing TS for a given problem, one must devise an LS technique appropriate for that problem. The major drawback in TS is that the tabu list size is fixed, and so it may permit moves that result in lower-quality solutions and prohibit moves that would result in higher-quality solutions.

3.2.3 Reactive Tabu Search (RTS)

In reactive tabu search (RTS), TS is combined with a feedback scheme that automatically adjusts the tabu list size. In essence, RTS still maintains the structure of TS in using the set of temporarily forbidden moves within a tabu list. What RTS provides is "a fully automated way of adapting the size [of the tabu list] to the problem and current evolution of the search, and an escape strategy for diversifying the search when the first mechanism is not sufficient" [BT94]. Using a hash table, all previous solutions are recorded along with their frequencies of occurrence. RTS, because of the escape diversification technique and the exploitation of fast memory structures, does not require an initial choice of the tabu list size and at the same time provides robust and efficient convergence.

3.2.3.1 Adaptation of RTS for Routing Problem

To apply RTS to the routing problem, a number of choices and adaptations were necessary. We summarize here the main elements of that adaptation: [CEH06]

- Set size of hash table for recording different solutions found
- Adjusted parameters to effectively search possible solutions
- Formulated initial solution for each problem to be solved based on bandwidth available, delay constraints of routing requests, total bandwidth used for best effort requests, and bandwidth shared among best effort requests
- Used Dijkstra's algorithm to find possible paths
- Used Breadth First Search to find possible paths
- Formulated a fitness function to find the best weighted solution for each problem

- Quality of Service requests met, remaining bandwidth shared among best effort users
- Adjusted weights for best effort terms to contribute more significantly to the fitness
- Removed links for forward and reverse paths for chosen connection requests
- Adjusted tabu list size by adjusting RTS parameters
- Incorporated data structures to control repetition of partial solutions, giving better coverage of the solution space
- Found feasible solution for each problem

An important decision in implementing a global search solution, such as RTS, is the initial solution chosen to be the starting point of the search. In the course of the project we tested results in which Dijkstra's algorithm provided the initial solutions and later used solutions provided by OCRLSP and finally the DORA algorithm (both of which are discussed below).

3.2.3.2 RTS HC3 Implementation Details

In our implementation, requests are sorted by priority, with requests of like priority arbitrarily ordered. For this order, best-effort requests are considered to have the lowest priority. The initial solution is generated by allocating paths (both forward and reverse) for the requests in the sorted order. For each path, we use Dijkstra's shortest-path algorithm with link bandwidths as edge labels and requested bandwidth and delay as constraints. After a forward and reverse path is allocated for a request, the bandwidth of each link on either of these paths is reduced by the requested bandwidth. The resulting residual bandwidth is used for the edge labels in finding paths for the next request. After generating an initial solution, NCA&T's implementation of RTS will perform the following: [CEH06]

1. Record the current solution in the hash table.
    a. If the solution occurrence frequency meets the threshold value, perform an escape (follow steps 2-5 while disregarding tabu status).

2. Randomly choose one already satisfied request. Requests later in the sorted order are more likely to be chosen than those earlier in this order.
3. Generate a move and record it in the tabu list.
    a. Generate a random number between 1 and the smaller of (i) the number of links in the forward path solution for the request and (ii) ¼ the number of links in the network; remove that many randomly chosen links from the forward path.
    b. Follow a similar procedure to remove links from the reverse path solution.
    c. Place these forward- and reverse-path links, associated with the chosen request, into the tabu list as a record of a single move.
4. Reallocate forward and reverse paths for the chosen request (without using the removed links).
5. Allocate forward and reverse paths for all requests following the chosen request in the sorted order using all possible links, including those made tabu for chosen request.
6. Repeat steps 1-5 for a specified number of iterations or until stopping criterion is met.

Note that each iteration finds a complete feasible solution. Paths (both forward and reverse) are allocated using Dijkstra's shortest-path algorithm as in finding the initial solution as described above.

3.3 Shortest Path Algorithms

3.3.1 Open Shortest Path First

Open Shortest Path First (OSPF) [MOY98] is the default algorithm in use throughout the Internet today. It is based on use of Dijkstra's algorithm for shortest path identification, and develops a routing table of static shortest paths without bandwidth constraints. The cost matrix for Dijkstra's algorithm in our implementation used the link delays of the unloaded network to determine path cost.

However, OSPF routing alone does not satisfy the needs of a QoS supporting network. Rather, routing must be supplemented by a reservation request and stateful admission control system, such as IntServ. The static paths obtained via OSPF form the routes explored for reservation. Static paths in this case mean these paths obtained when the entire "unloaded" network is solved for shortest (lowest delay) paths between any two points, once and only once. Then, as each connection request comes in, an IntServ reservation system attempts the reservation of the required connection bandwidth on that static path. If the reservation fails, the connection is not admitted and reservation of the next connection request is attempted. This is obviously suboptimal (as will be amply demonstrated in the results section) since no information about previous reservations is used to attempt to find other paths than the one initially associated with the entrance/exit node pair. Furthermore, no attempt at pre-emption of older, lower priority connections is attempted.

3.3.2 OSPF+

OSPF+ is a variation of OSPF developed at WPI as part of this thesis work, and is a non-standard attempt at what is called a "traffic engineered" network solution. This approach is based upon execution of the OSPF algorithm on the unloaded network many times, but in each case using only links that can handle some prescribed level of bandwidth. As before, all entry and exit pairs are computed for the list of connections required. When considering an entry/exit pair the cost matrix is pruned to only include links that can support the requested bandwidth at a minimum. This will remove the paths that OSPF will find which would not support the connection even on an unloaded network. With the traffic engineering enhancement all connections would be possible on an otherwise empty network. In effect, this implements a DiffServ network in which connections have code-point attributes that direct connection reservation to pre-coded network paths associated with that bandwidth grade.

As will be seen in the results, simply engineering bandwidth-graded solutions produces a significant improvement in achievable performance as measured by total connection

admissions.

This traffic engineered solution and the other traffic engineered solutions to be described below were actually implemented via MPLS label switching in the protocol simulator that will be described later.

3.3.3 Constrained Route Label Switched Path (Unordered)

Constrained Route Label Switched Path (CRLSP) routing algorithms have been developed for use in conjunction with MPLS networks [AC00]. Such routing techniques use label distribution protocols (CR-LDP) [AC01] to distribute connection specific paths to MPLS nodes, providing a level of dynamic control that far exceeds the static nature of OSPF and DiffServ type networks.

The variation of CRLSP we implemented uses a bandwidth and delay constrained solver with priority pre-emption developed for MPLS networks. CRLSP will, on a per flow basis, solve for the shortest path based upon all previously allotted bandwidth given to higher precedence users, pre-empting others.

This algorithm uses Dijkstra's algorithm with a cost matrix based on link delay, just as the previously described algorithms. This diverges from OSPF by the fact that it also uses the bandwidth available on the network at the time of the connection request rather than the unloaded link bandwidth. In the example network shown below in Figure 3.1 there are two currently existing connections using 0.7 Mbps from subnet Y to node D and 0.2 Mbps in the reverse direction. The second connection uses 0.3 Mbps from subnet Z to node F and 0.8 Mbps from node F to subnet Z. These asymmetric bandwidth requests are typical of HC3 connection requests in which one direction requires a much greater bandwidth than the other owing to client/server relationships.

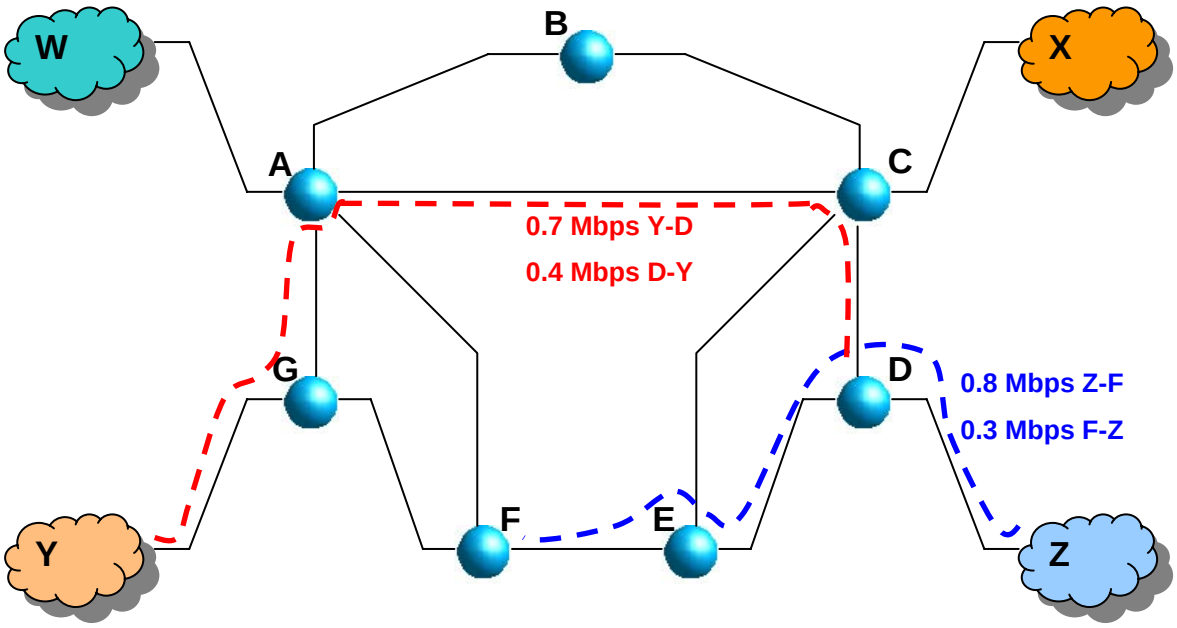


Figure 3.1 - CRLSP Example Network

For this case we will examine the outcome when CRLSP seeks an additional route from subnet X to node F with 0.7 Mbps in the forward, subnet X to node F, direction and 0.2 Mbps in the reverse, node F to subnet X, direction. This connection format will be used throughout this document: with the forward direction being from the first stated node to the second node, and the reverse being the opposite direction.

When CRLSP seeks to route an additional connection, all links that will not support, at a minimum, the requested bandwidth, are pruned from the network. The remaining links are used to find the shortest delay path from the entry node to the exit node. This pruning can be seen in Figure 3.2, below, for the forward direction.

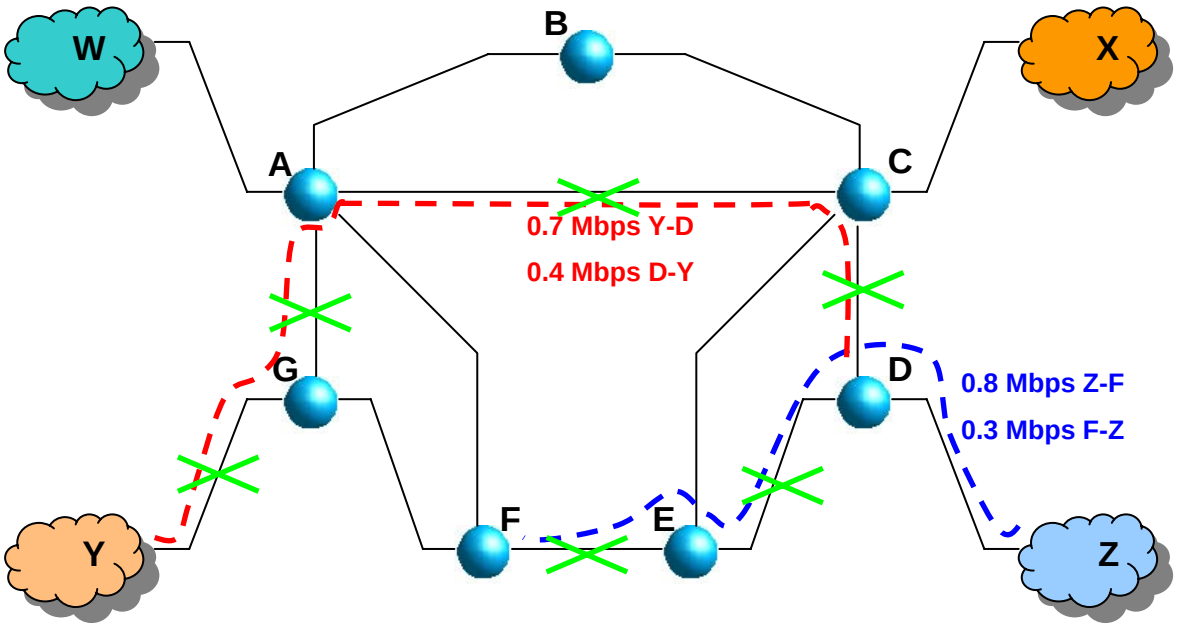


Figure 3.2 - CRLSP Pruned Network

While the OSPF routing algorithm would choose either X-C-E-F or X-C-A-F, neither of these routes would work resulting in a denied connection. Using a dynamic routing technique the feasible forward route left after the above deletions would be X-C-B-A-F, a slightly longer route but the only route with enough bandwidth left. If the route found exceeds the maximum delay constraint for the QoS connection the reservation will fail, but if the path found is within the delay constraint a reservation will be tentatively made. The reverse route will then be calculated in the same manner using the reverse bandwidth constraint. In this case, the reverse route would be F-E-C-X which has sufficient bandwidth for the 0.2 Mbps reverse path. Since both directions have a feasible solution, a reservation is made for both paths at once. If either direction fails to be feasible both reservations are aborted. The network with this additional connection is shown in Figure 3.3, below.

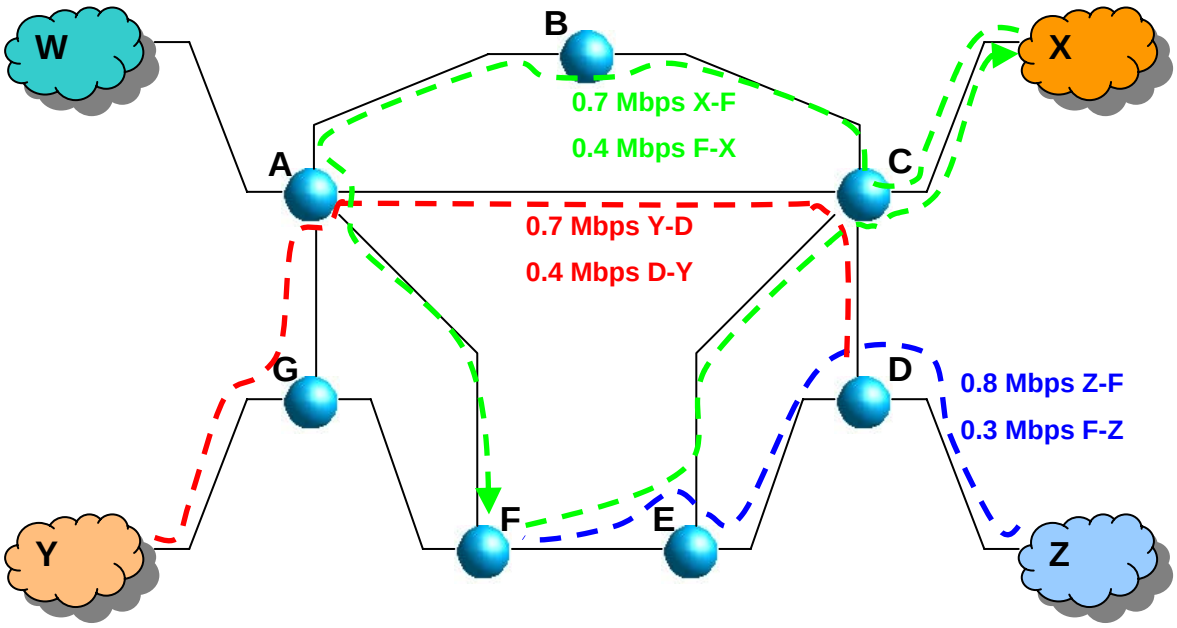


Figure 3.3 - CRLSP with Additional Connection

With this dynamic per flow routing algorithm more connections can be satisfied than with static route based reservation protocols and as connections are added they are routed around existing connections as possible. This algorithm, as stated, will not respect precedence, however, and makes reservations based on the order in which requests were made.

3.3.4 OCRLSP (Ordered CRLSP) Routing

Ordered CRLSP applies the CRLSP routing algorithm as described above, however, it first orders all requests from highest to lowest priority. It then solves for available paths beginning with the highest priority requests. When implemented in a system with ongoing requests, this translates into the need to resolve for all current paths and the newest request each time a request is made. Thus, OCRLSP constitutes a significant increase in required processing from the other shortest path algorithms. It further requires the ability to actively re-route existing connections based upon new solutions. This latter capability is inherent in the CR-LDP protocol by which it is supported. These costs in increased processing however support the precedence respecting and pre-emptive behavior that is necessary in a multi-level network such as the HC3 topologies.

3.3.5 Dynamic Online Routing Algorithm (DORA)/(WDORA)

While the benefits of search-based algorithms such as TS and RTS might be significant in that they explore much broader opportunities for optimization than shortest path algorithms, the larger execution time is potentially prohibitive compared to shortest-path algorithms such as CRLSP and OCRLSP. It remains an open question with any optimization problem as to whether a heuristic specific to that problem might not unlock most or all of the improvement to be had by a search-based algorithm. Searching the literature for such a heuristic algorithm revealed one which had been proposed that "made engineering sense" and was practical with respect to processing requirements: DORA, or Dynamic Online Routing Algorithm [BSI02]. A single evaluative study of this approach was available [MAB05], from which it appeared that performance was excellent for it and a newly proposed extension, WDORA.

DORA is a novel routing algorithm that places paths with reserved bandwidth evenly across the network, which allows more future paths to be accepted in the network and allows balancing of the traffic load. DORA was claimed to have produced results that are at least as good as those produced by algorithms that are much more complex [BSI02].

The main goal of DORA is to avoid routing a path over links that have high potential to be part of some other path and have low residual available bandwidth. WDORA uses the same basic principles in route determination as DORA. The major difference is that, in calculating routes, if there is more than one possible route to satisfy a request, the route that has the greatest bandwidth is chosen. Hence, the term "widest" (which supplies the "W" in "WDORA") relates to the widest bandwidth being chosen. Its benefit is that "WDORA selects routes having the largest residual capacities which results in routes having a greater probability of being still feasible for future requests" [MAB05].

3.3.6 DORA2

The DORA2 algorithm developed as part of this thesis work is an adaptation of the DORA/WDORA mindset that exploits knowledge about the network topology. An

additional search is implemented in DORA2 to find alternative routes to free up the most popular links for later use. This expansion on traditional shortest path algorithms yields paths that are not the shortest path, but are still "short enough" to meet the QoS delay constraints as we might like to use these if that leaves open shortest paths that might be essential later and shouldn't be wasted when longer paths will do. By selecting a waypoint on the network that is not on the shortest path a conjunction of shortest paths, which are less optimal, can be found. If additional waypoints were specified, first two, then three, until the available number of nodes are exhausted then all possible paths would be found. In this sense, our single waypoint solution set comprises a first order approximation of the set of complete paths, while maintaining the computational feasibility to run in a real-time network.

The requested connections are first ordered by priority with connections of the same priority ordered by the time order in which they were requested. This step maintains the preferential precedence order while respecting connections that came earlier, allowing them to stay connected if possible. The theory behind DORA2 was that if the network traffic were spread out across the network, and popular links were used last it would tend to use the "outside edge" of the network before congesting the central nodes. As an initial setup for the routing solution, the network was analyzed and each link was given a path potential value (PPV). Given the network in Figure 3.4, all feasible paths between Y and D are shown below. These are the paths that meet the delay constraint in the QoS request made for this connection.

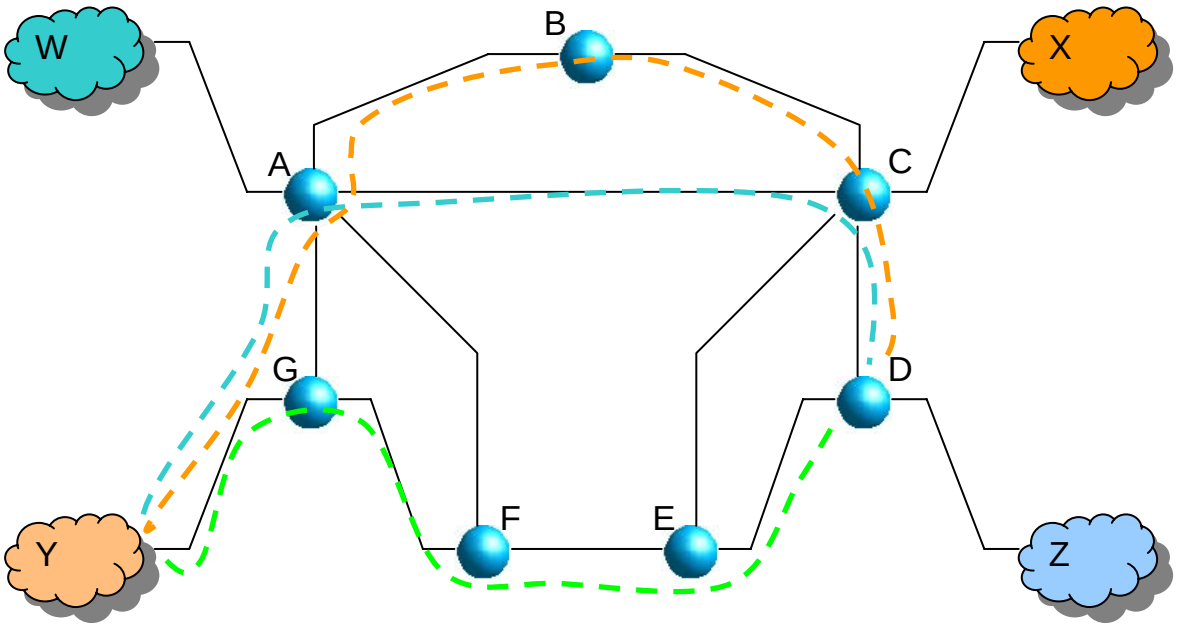


Figure 3.4 - All Feasible Paths from Y to D

The path potential value is calculated as the number of times a given link is found in the exhaustive set of possible paths for all connections. For the connection from Y to D the path values would be given a score of 3 from Y to G. The link between G and A would receive a PPV of 2 from this connection, similarly the link between F and E would receive a score of 1. These scores would be summed for each link across all entry and exit nodes that have a connection request. In this way each link will have a PPV that represents its popularity among all connections.

The novel approach described above was used to quickly find all possible paths different than the shortest path by a single waypoint. Given a connection request the network would first be pruned of all links that did not meet the bandwidth requirement in a fashion similar to what takes place with CRLSP. Dijkstra's algorithm was then executed twice, first using the entry node as the root of the minimum spanning tree. For the example to follow all links will have the same delay. In Figure 3.5, it can be seen how the shortest path from subnet Y is found to each of the nodes in the network. If a path was found from the entry node to the exit node, Dijkstra's algorithm was executed a second time, with the exit node as the root of a second minimum spanning tree.

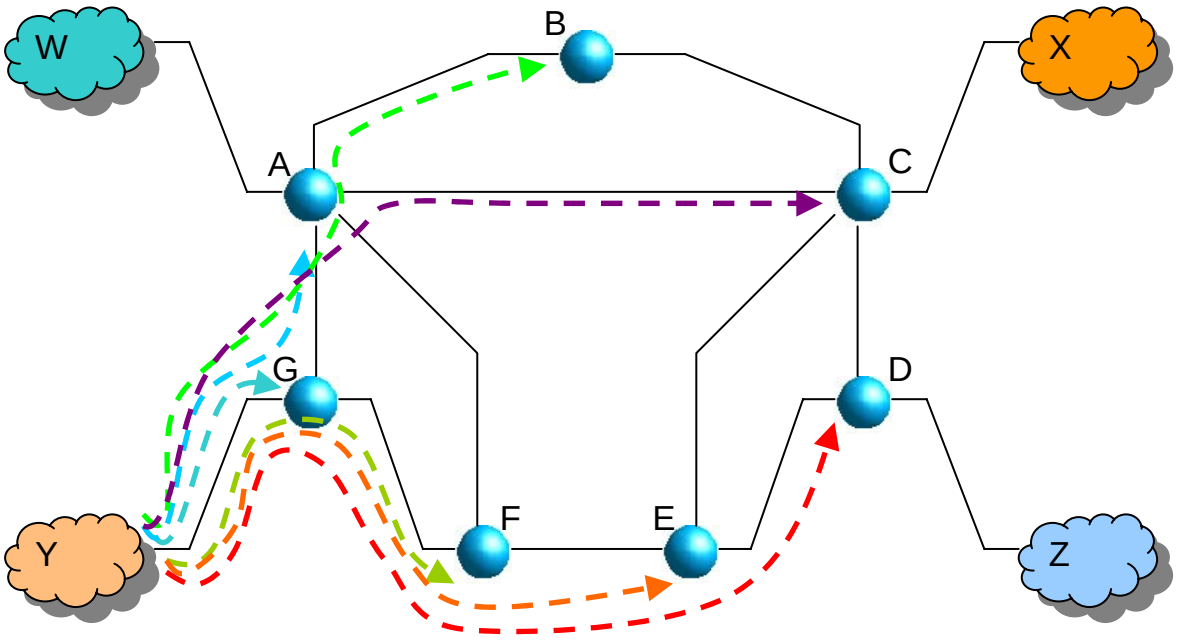


Figure 3.5 - Dijkstra's Algorithm from Entry Node

Using the current available bandwidth values for the forward direction, the bandwidths as seen from subnet Y outward, Dijkstra's algorithm is executed from node D to find the shortest paths to alternative nodes. Figure 3.6 illustrates that the shortest path between the entry and exit node is the same, since the bandwidths and delays were identical in both cases. It is possible to remove from further consideration any nodes along the shortest path, since the path to that node is always along the shortest path already selected. This exploration of the network from both sides exposes rapidly all additional paths different by a single waypoint.

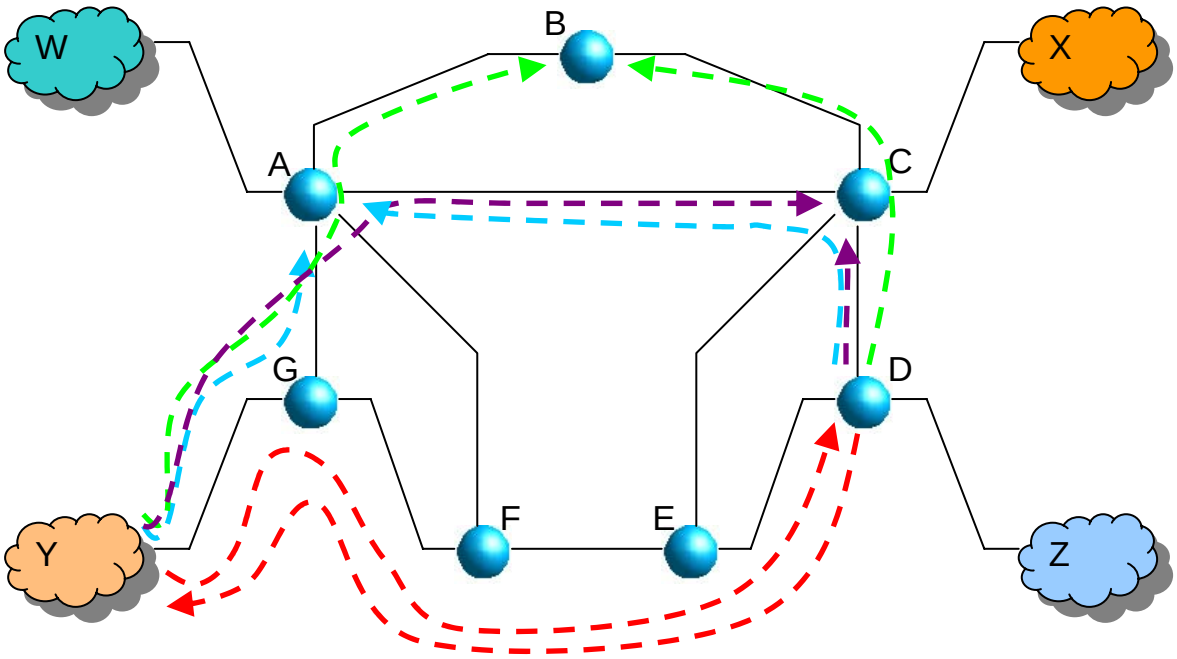


Figure 3.6 - Bi-Directional Dijkstra's Algorithm to Find Alternate Paths

The other path of note is that looking at the shortest paths to node A (blue arrows) and node C (purple arrows) will yield a final path that is identical, namely Y-G-A-C-D. This would be pruned as well, to form only one route. In the end there are three potential paths, Y-G-F-E-D, Y-G-A-C-D, and Y-G-A-B-C-D, for this connection pair.

While it is known that these three paths meet the bandwidth requirements, they have not yet been tested for end-to-end delay. The next step is to determine which, if any, potential paths meet the QoS delay requirement. To obtain the end-to-end delay just requires adding the two values computed by Dijkstra's algorithm for each of the two partial paths. If this delay is within the QoS requirements, we add the route to the list of possible routes. This procedure would then be followed a second time to determine the reverse route as if it were a second unidirectional connection request. This split-route technique maximizes usage of the asymmetric nature of link and connection request bandwidths towards finding means to route as many connections as feasible. Routes with the smallest individual PPV scores are then chosen first, with the sum of all PPVs used to break ties.

3.3.7 Hybrid Method

We also implemented a fusion of DORA and RTS to obtain even better results than obtained by virtue of heuristic optimization in the vicinity of the improved starting point found by DORA2.

4.0 Evaluation Method

To evaluate the performance of these various algorithms, we undertook the development of a software toolset comprising a test-bench based on Monte-Carlo statistical performance evaluation. What was needed were tools to automatically generate large numbers of routing problems that shared common characteristics of all convergence layer networks while attributed with randomly selected topological details, connection attributes, and QoS demands. We also sought not just to evaluate these methods in a sterile mathematical simulation but to implement a full event-based packet-level simulation of the routing to verify feasibility of solutions and to directly capture performance features such as best effort traffic bandwidth utilization and delay.

In summary, we sought to generate:

- A suite of problems auto-generated with a given network complexity, fixed node, and connection request numbers
- Probabilistically generated connection requests distributed across the QoS priorities and best effort traffic.
- Probabilistically assigned bandwidth and delay constraints for QoS.

For every routing algorithm, the same suites are evaluated in order to avoid any unfairness of comparison and so that we may actually compare on a problem-by-problem basis the outcomes of the alternative solutions.

A protocol for problem representation and exchange (between the two university working groups), and two applications to support automatic statistical evaluation were created. These will be described in the next sections.

4.1 SRPDP – Simple Routing Problem Description Protocol

Simple Routing Problem Description Protocol (SRPDP) was used to exchange information between the members of the research team and between components of software developed by the teams. The SRPDP reduces the description of the nodes, links, and connections that comprise a routing problem and paths that define its solution into a single Matlab structure of arrays with only numerical entries.

To support realistic problems, SRPDP can represent:

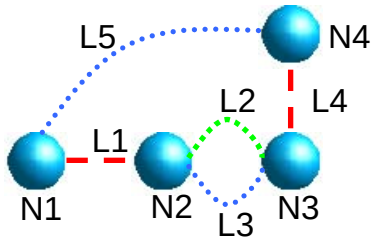
- multi-layered networks (many links between two nodes)
- multi-waveform networks
- asymmetric flows
- connection precedence
- mixed traffic, including QoS requesting and best effort (BE) traffic
- QoS with rate and delay constraints
- Convergence Layer architectural features

4.1.1 SRPDP Structure and Entities

A problem is comprised of a set of matrices stored in Matlab. Nodes, links, and connections make up the problem, while a series of fields for paths hold the solutions of the different algorithms. A final field holds calculated results for each algorithm such as number of connection requests granted and bandwidth granted. Figure 4.1 below shows a sample of a problem structure.

Nodes

ID	Delay	Rate	Vector
1	10 ms	1000 Kbps	{1,10,1000}
2	5 ms	1500 Kbps	{2,5,1500}
3	5 ms	1500 Kbps	{3,5,1500}
4	10 ms	500 Kbps	{4,10,500}



Links

ID	Src	Dst	FR	BR	Delay	Cost	Waveform	Vector
1	N1	N2	1500 Kbps	1500 Kbps	150	0	0	{1,1,2,0,1500,1500,0,150}
2	N2	N3	1000 Kbps	350 Kbps	250	0	1	{2,2,3,1,250,1000,0,350,
3	N2	N3	600 Kbps	800 Kbps	400	0	2	{3,2,3,2,600,800,0,400}
4	N1	N4	750 Kbps	750 Kbps	300	0	2	{4,1,4,2,750,750,0,300}
5	N3	N4	1500 Kbps	1500 Kbps	50	0	0	{5,3,4,0,1500,1500,0,50}

Connection Requests

ID	Src	Dst	Prec	FR	BR	Bucket	Delay	Vector
1	N1	N3	1	750 Kbps	10 Kbps	0	300 ms	{1,1,3,1,750,10,0,300}
2	N1	N3	2	600 Kbps	15 Kbps	0	400 ms	{2,1,3,2,600,15,0,400}
3	N1	N3	2	600 Kbps	15 Kbps	0	400 ms	{3,1,3,2,600,15,0,400}
4	N1	N3	2	1200 Kbps	300 Kbps	0	50 ms	{4,1,3,2,1200,300,0,50}
5	N2	N3	3	800 Kbps	20 Kbps	0	300 ms	{5,2,3,3,800,20,0,300}
6	N3	N2	1	500 Kbps	10 Kbps	0	450 ms	{6,3,2,1,500,10,0,450}

Figure 4.1 - SRPDP Problem Sample

This flexible structure allows networks of much greater complexity than required to be represented with ease. Each node has a bandwidth and processing delay, links can have asymmetric properties, and connection requests comprise any arbitrary set of entry and exit nodes. If additional constraints were necessary, at a later date, then the required modification would only be a matter of adding an additional column to the relevant matrix.

4.2 Problem Generator

To test the routing algorithms, an automatic, probabilistic problem generator was needed. We created a random topology generator, *routeprob* (later updates were given the more unwieldy name *hc3_generate_problemset*), to support Monte Carlo testing and comparison of various QoS routing solutions. The topologies which are generated are multi-tiered (hierarchical in architecture) and multi-layered (multiple links with different attributes may connect a given pair of nodes).

The Matlab-based topology generator automatically creates network-architecture, connection-protocol, and routing-problem description files in SRPDP format. Parameters passed to the Matlab function allow specification of the number of nodes of each type, number of connection requests and other relevant parameters.

The resulting networks mimic HC3 convergence-layer style networks consisting of highly connected sub-networks (e.g., ground-based ad hoc nodes and terminals) multiply connected by a small number of resource limited routers (e.g., satellite and UAV resources) with access to the Global Information Network (GIN).

To generate a problem, the number of nodes and connection requests must be given. There were two network sizes used in our testing, one with 12 nodes, and one with 24 nodes. These nodes were broken into three subnets, which were connected to one another via satellite links. In Figure 4.2 is a representation of one such network. There are three subnets circled in blue, with connections to the satellites in green. The trapezoid nodes labelled 13, 14 and 15 represent satellites with the Global Information Network being the double circle labelled as 16.

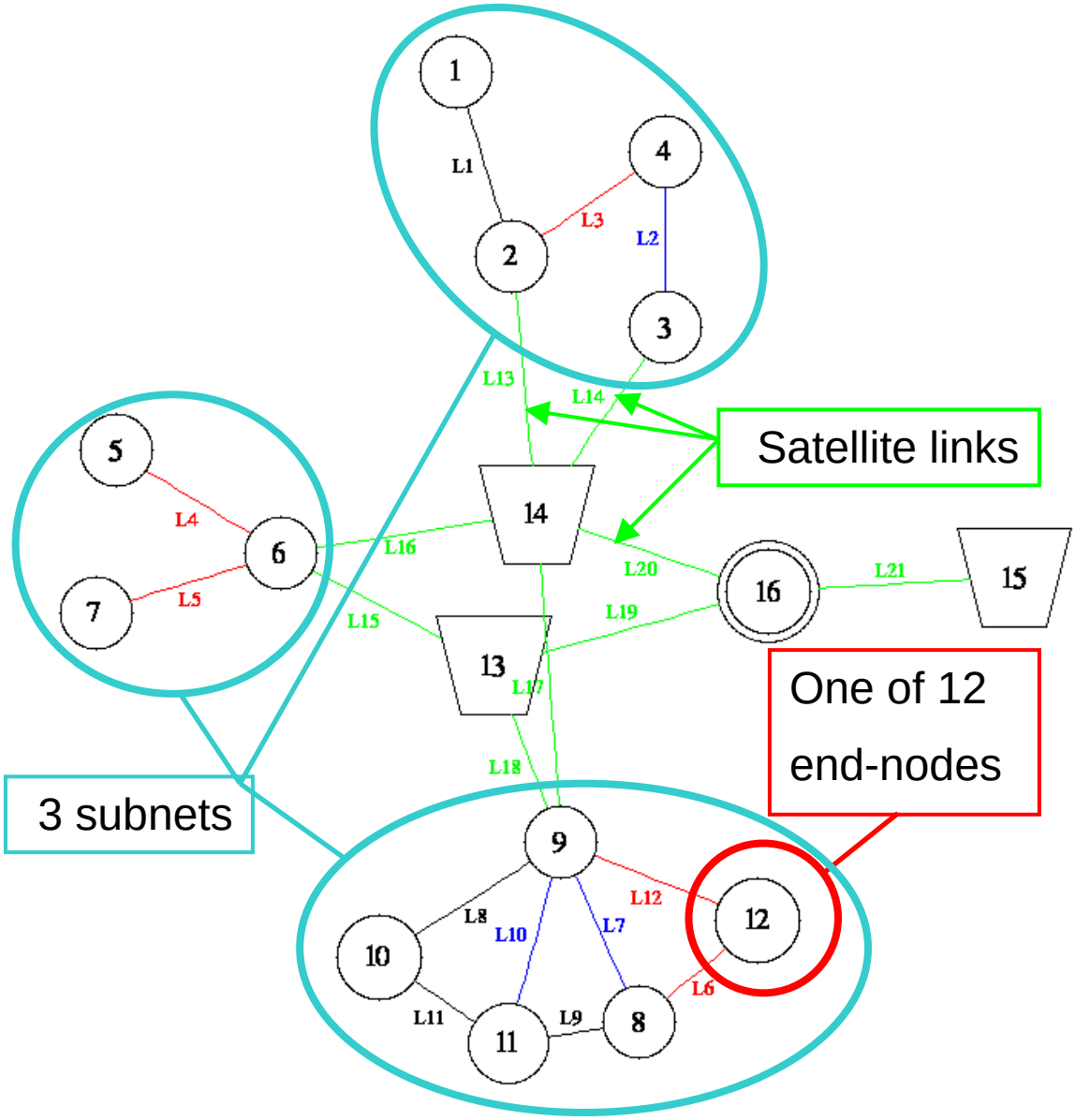


Figure 4.2 - Sample Network

The link properties were generated using the built-in random function from Matlab. The satellite link delays were uniformly random between 200 and 500 seconds. Similarly the bandwidths were Gaussian random variables that were then rounded to the nearest multiple of 64 Kbps between 100 and 1500 Kbps. This represents the standard bandwidth of a voice channel as defined by the telecommunications industry, and is a common unit of allottable bandwidth in satellite traffic allocation. The connection request delays were also generated using the same technique with the square root of the

number of nodes as a multiplier. This allows for a scaling effect; as the network size grows the tolerance for delay is similarly increased, though at a smaller rate. Bandwidths for the connections were generated from a uniform distribution between 10 and 500 Kbps in multiples of 64. The connections were also given a precedence, which was uniformly random between 1 and 4, and a QoS status of either 1 or 0 with equal probability. With this technique approximately half the connections would be QoS connections, the remaining being best effort connections.

4.3 Blocking Metric

We shall begin with a general introduction to the means for comparison of MLPP (Multi-Level Precedence and Pre-emption) routing algorithm outcomes that was developed during this project. Because of the multiple levels of precedence involved, a direct comparison of the number of overall successful QoS connections would ignore an important component of the optimization – that high precedence connections bear overwhelmingly more weight than low precedence connections. As will be seen, we introduced the notion of a "trumping or blocking metric" to capture this distinction. That is, an additional high priority solution trumps any number of lower precedence connections.

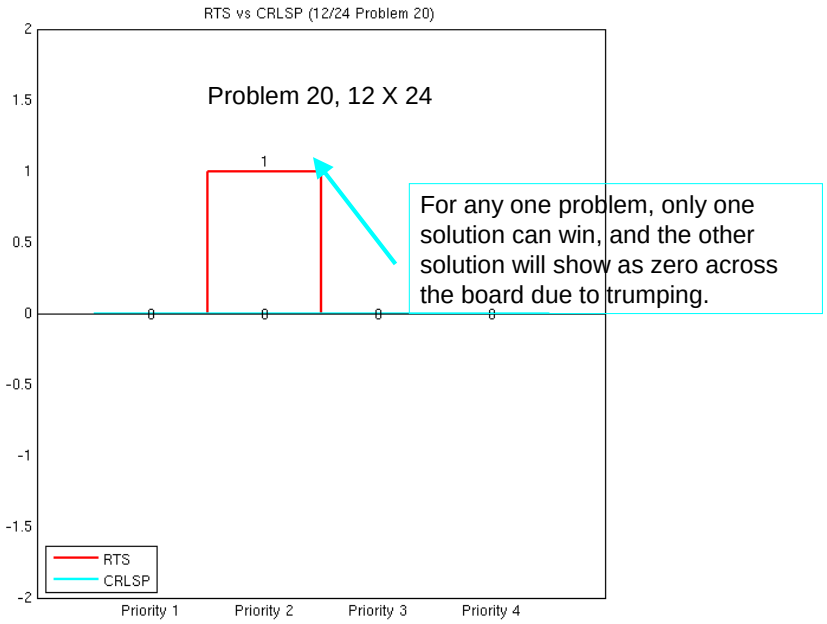


Figure 4.3 - Graphical Blocking Metric

This notion of trumping should be clearly reflected in the metrics applied to summarize the performance comparisons of various routing algorithms. We developed a graphical metric depiction strategy that performs this function in an intuitive fashion. Figure 4.3 above shows the results graphically, from problem 20 of the 12 x 24 problem set, depicting the number of connections of the highest priority by which RTS outperformed CRLSP in this problem. Even if CRLSP generated more connections of lower priority in this problem, these wins are trumped by the loss at the higher priority and hence are not shown. In this case you can clearly see that the RTS solution, shown in red, had one additional connection request granted at the priority 2 level.

We created a program *hc3_generate_winloss_matrix* that parses all the data from a Monte Carlo batch execution generated by *hc3_autosim* and reduces it to a numerical evaluation of the entire problem suite that describes the overall performance differences between two or more QoS routing algorithms. The blocking metric is calculated using the total connections granted at each priority level, and respecting priority finds how much "better" one algorithm performed over another. These numerical comparisons were then passed into *hc3_draw_bargraph* which would turn the numerical data into an easily comparable graphical representation.

- Problem #20, 12 nodes, 24 connections

RTS found 1 more 2nd priority connection it could make by alternative routing of higher priority connections

As a result, bandwidth utilization by priority 2 traffic has increased by 384 kbps

Request Priority	1	2	3	4
Number of Requests	1	3	8	1
CRLSP granted	1	0	3	0
RTS granted	1	1	3	0
BW Requested	138	1280	3850	448
CRLSP BW	138	0	1098	0
RTS BW	138	384	1034	0

Figure 4.4 - Example of Trumping

The table in Figure 4.4 shows the values of the numbers of connections and bandwidth (BW) obtained for the 20th problem of the 12x24 test suite and shows that there were one priority 1, three priority 2, eight priority 3 and one priority 4 connections requested. The third row of the table shows that the classical routing algorithm was able to grant four of the connections. The fourth row of the table shows that the Reactive Tabu Search algorithm was able to find routing for an additional priority 2 connection. The final two rows show the raw bandwidth granted. Because of the additional connection, RTS was able to increase the overall network bandwidth utilization by 332 kbps, through an increase of priority 2 traffic and a reduction of priority 3 traffic by 64 kbps.

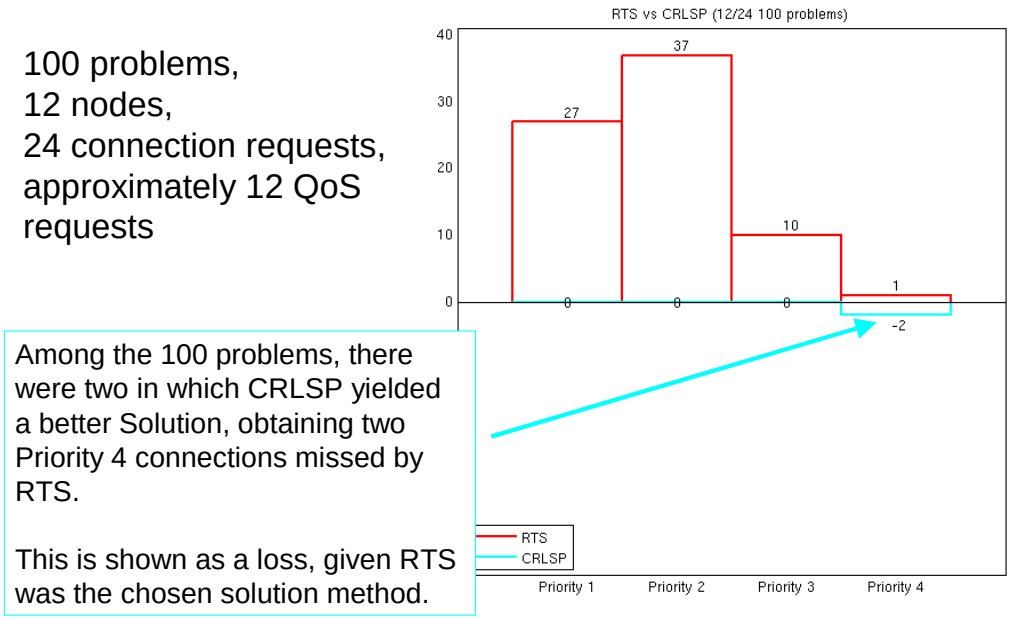


Figure 4.5 - 12 x 24 Suite Performance RTS vs. CRLSP

When this trumping technique is applied to an entire test suite of 100 problems, we can construct a summary figure such as Figure 4.5, and a pattern of behavior can be found. For these 100 problems RTS was able to find 27 additional priority 1 connections. In addition to these increased priority 1 connections RTS was able to grant connections at the lower priorities as shown with the 37 priority 2, 10 priority 3 and one additional priority 4. These connections were over and above whatever connections were already granted by CRLSP, giving an indication of how much "better" one algorithm did than the other. The algorithm on the top half of the graph is compared with the algorithm on the bottom, and it is indicated as shown by the blue line that two priority 4 connections were missed by RTS that were granted by CRLSP. This gives a graphical cost-benefit comparison between any two algorithms. RTS was able to provide a benefit of all those additional connections having missed optimizations seen by the other algorithm that would have yielded an additional two priority 4 connections.

4.4 Automated Graphics Generator Route_to_graph

Another Matlab program was created to help visualize the networks being generated. Route_to_graph was the name of the script that generates graphics files depicting the

networks specified with SRPDP structures to aid visualization and review of problems. The bandwidth and waveform support attributed to each link is shown. The output which is generated is in the format required by the graphviz tool, which is an open source tool for graph visualization available from Lucent Technologies. Route_to_graph will generate an output file similar to that shown in Figure 4.6, which is the input to graphviz and produces a graphical representation of the network as Figure 4.7.

```
graph G {
overlap = false;
center = 1;
edge [fontsize=10];
node [shape=circle];
1 -- 2 [label="L1" color="red" fontcolor="red" len=1.5];
2 -- 3 [label="L2" color="black" fontcolor="black" len=1.5];
2 -- 3 [label="L3" color="blue" fontcolor="blue" len=1.5];
1 -- 4 [label="L4" color="blue" fontcolor="blue" len=1.5];
3 -- 4 [label="L5" color="red" fontcolor="red" len=1.5];
```

Figure 4.6 - Route_to_graph Output File for graphviz

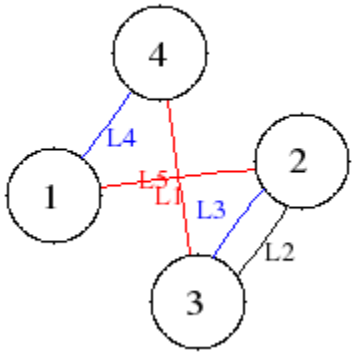


Figure 4.7 - graphviz Generated Network

4.5 Simulation Software

As described earlier, construction of an event-based simulator for the network and protocols under study was integral to this project. Implementation of a full protocol emulating simulator provides several benefits, which we shall present in this section.

Written to support this testing, Matlab scripts were implemented that would match a set of frameworks, for the network simulator ns [AC01], that were created previously in the project by research assistants Nick Sherwood, Darius Kazemi, and Professor David Cyganski.

A mechanism to implement rapid path specific routing needs to be supported by the network simulator used to apply the flow-based optimization of routes obtained by the algorithms explored in this project. Furthermore, rapid re-routing of flows without interruption of the flows is also required. Lastly, topology information must be rapidly collected from the network to provide necessary information for the execution of the route optimization program. These capabilities are beyond those supported by most segments of the Global Internet. Since the HC3 subnetwork is a relatively small network constructed and controlled by coordinated projects, it is possible to instantiate advanced network architectures that support the required capabilities. However, it should be demonstrated that such an architecture exists (with support from the network development community) and that it has border interface compatibility with existing (OSPF) technology. In this project we implemented, tested, and validated such a complete implementation based upon MPLS [SM00] components implemented in a TCP/IP context using the CR-LDP protocol for topology discovery and route distribution and CR-LSP for route optimization. This implementation is a network-ready demonstration of an advanced instantiation of the best-of-class recommendations for the Transformational Communications Architecture (TCA) network [AC01].

Another reason to implement a complete protocol level simulator is to check the proper operation of the routing protocols. Though every effort was made to write error-free software from well considered mathematical models for the optimization of network routing problems, software will be software! By using a simulator with a long history of testing and evaluation by the industry and universities, described and validated in hundreds of publications, we introduced an independent test of our solutions. Ultimately, this effort proved its worth as indeed many irregularities in the operation of proposed routing algorithms were identified using this independent evaluation and then corrected.

Finally, the protocol emulation provided means to independently validate the TCP/IP communications with per-flow routing in an MPLS network. This was accomplished with the equivalent of DiffServ-style class based queueing (CBQ) applied on a flow precedence basis. This is sufficient to guarantee the bandwidth protection and delay constraints required by the individual-flow QoS demands. Again, we were able to demonstrate and validate this architectural approach.

We created our simulation environment based upon ns (version 2.26) with extensions to support traffic engineering (RSVP-TE), [ACG] to support centralized QoS routing and constraint based dynamic MPLS route discovery (MNS version 2.0), and to support distributed QoS routing. Several required patches and other modifications of the source code were integrated into these packages to obtain a version that would function in a current generation Debian Linux based system. We also wrote and inserted additional code hooks into the ns simulator for purposes of monitoring, troubleshooting and implementing multi-precedence pre-emption processes.

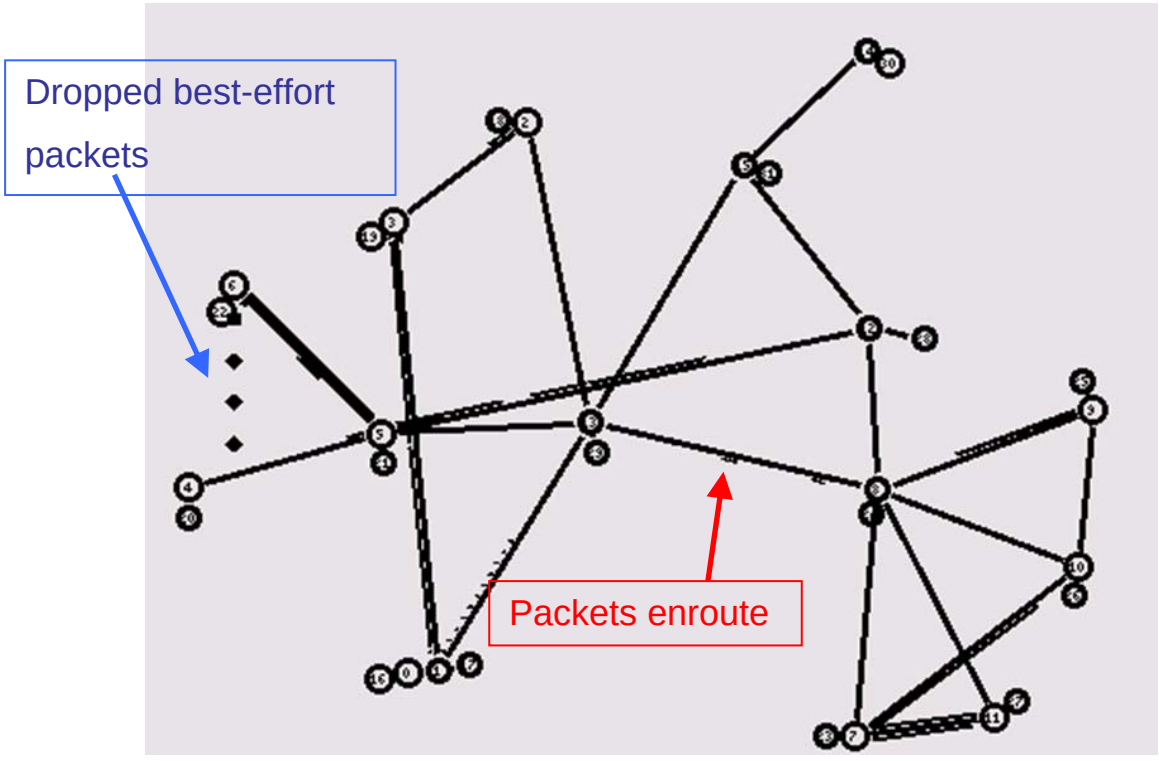


Figure 4.8 - ns Simulation Visualization

Above in Figure 4.8 is an example of a 12 node problem being simulated. It is possible to observe and track the packets moving through the network, and visually see that only best-effort traffic is dropped. This is further verified by Matlab scripts that we wrote to parse the verbose output logs that ns generated with our additional code hooks.

4.6 MPLS Traffic Engineering

The following list summarizes the capabilities possessed and techniques demonstrated by our ns implementation and validated in this project:

- Route forwarding implemented by applying MPLS
- LSR nodes populate entire network
- CR-LDP distributes explicit solution routes with constraint based checks on bandwidth, queue buffer allocation
- Class Based Queues separate guaranteed bandwidth data flows from best effort flows
- Simulation checks the feasibility of the route optimization algorithms by monitoring dropped packets and measuring actual bandwidths achieved
- Best Effort performance is directly measured
- Fail-over paths can be assigned
- Automated Flow Aggregation to reduce the number of CBQs and MPLS flow identifiers required

Several software tools essential for this implementation and analysis were written by the team for this project and are summarized below.

4.7 Network Simulation Trace Analyzer

The outcome of an execution of the ns simulation is a trace file, that is, a record of every event involving the transmission or the reception of any packet at any node in the network. It is from such enormous and detail-specific data that overall evaluations of network performance and constraint compliance must be extracted. To do so, we wrote several tools using Awk scripts and the Matlab system that parse these trace files and

form digests of information that are directly related to dropped packets and best effort traffic delivery information from which performance data may be generated.

4.8 Monte Carlo Test System

We created the *hc3_autosim* Matlab script, which automates the following tasks that are required for every test:

- Generation of routeprob problem structures
- Execution of all routing algorithms under test (OSPF, CRLSP, RTS, etc.)
- Simulation of each solution network
- Matlab based evaluation of route feasibility
- Parsing of simulation trace data for statistics and simulated feasibility
- Generation of data structures with requests granted by each algorithm for each request priority and a report on the bandwidth requested and granted
- Generation of numerical and graphical algorithm comparison
- Parallel execution of routing algorithms in *autosim* is supported by a Matlab-based inter-computer network communications system that distributes routing jobs by round robin scheduling to a given group of computers and gathers the outcomes for further processing

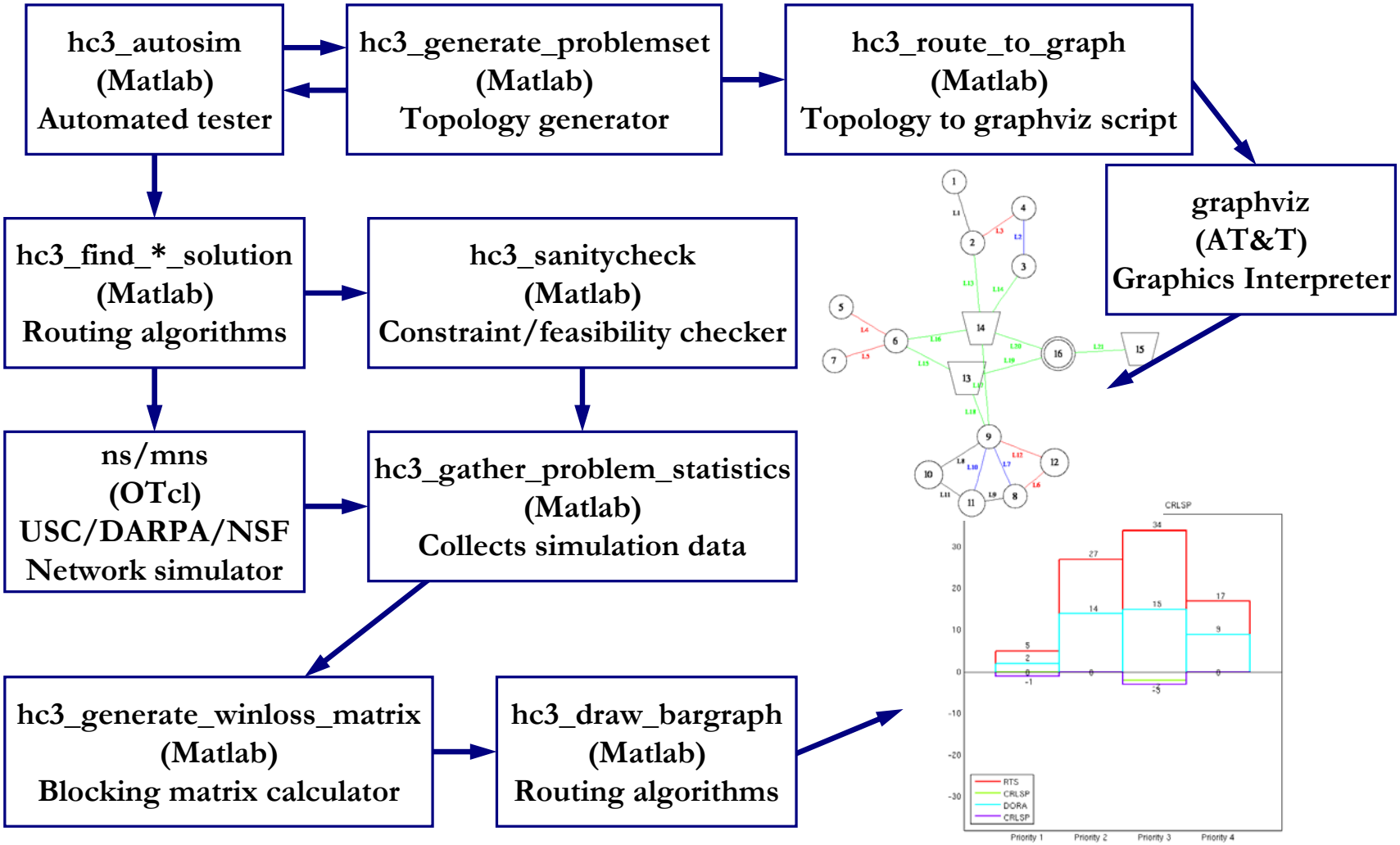


Figure 4.9 - WPI Testbed System

The completed system diagrammed in Figure 4.9 allowed parallelization by distribution to many computers on our network and thus the utility to run evaluations of a large number of networks to provide a measure of statically relevant results in a useful timeframe. The use of a Matlab based feasibility checker provided near instantaneous verification of valid routing solutions, while the event-based protocol level simulator provided independent confirmation. This allowed for fast failure detection during debugging and development, as well as quickly generating comparison results for analysis in minutes, while the simulations would provide verification when they finished execution.

4.9 Traffic Engineering Generator

We created an ns command script (in the TCL language) to instantiate fixed label switched paths (LSPs) in an MPLS network with traffic engineering. By defining LSPs, we are able to force packet flow to conform to optimized routes as determined by either an external and centralized routing algorithm (such as implemented by NCA&T) or local mechanisms such as found by distance-vector or OSPF routing algorithms.

A summary of operational capabilities and components includes:

- LDP distributes explicit solution
- Class Based Queues separate guaranteed bandwidth data flows from best effort flows
- Flow Aggregation reduces the label burden automatically
- Simulation checks the feasibility of the route optimization algorithms
- Dynamic rerouting supported
- Capability to assign fail-over paths in future work

4.9.1 CRLSP-Based Centralized Constrained Routing MPLS Module

We created a constrained route label switch path (CRLSP) routing algorithm module for MPLS based upon the CR-LDP (constrained route label distribution protocol). Our implementation of CRLSP implements link-bandwidth constrained routing with

admission control and bandwidth reservation with precedence-based resource preemption. The admission control and reservation aspects emulate IntServ. Our admission and reservation system handles each flow direction independently so that asymmetric connection requests are appropriately allocated and refused as units if blocked by admission control. We also implemented a centralized first-to-last priority scheduling rule (functioning under the name OCRLSP as described above) rather than defaulting to priority-based preemption so as to optimize route solution convergence time, but this can be subsequently relaxed. Connectivity and resource information is gathered dynamically via CR-LDP messages, and proposed routes are computed via a distance-vector algorithm.

4.10 Summary

There were a great number of Matlab scripts created over the course of this project. The development of a description language for networking problems allowed the creation of a generator for that protocol which in turn created the need for a graphical problem representation. In addition, numerous solution generators were created, at least one for each algorithm tested and shown here, and several more implementations which we did not pursue to full testing due to obvious flaws or lack of obvious advantage. The network simulator was an invaluable tool in providing a third party verification of our work, and was necessary to provide insight into where a bug in the solution could be located. The full implementation details of the many generators, analyzers and simulators written for this project can be gleaned from the source code package that was developed.

5.0 Results

As indicated earlier, tests were performed using custom programs written for and executed in the Matlab environment and using the ns simulator with MPLS modules as modified by WPI to support CRLSP-based route optimization with constraints. Tests were conducted for many networks, but these were restricted to be of two sizes. The restriction to these two cases was for the purpose of exhaustively testing some cases rather than obtaining only anecdotal results about many. The smaller network type consisted of a 12 node, 24 connection-request problem, which on average had 12 QoS connection requests and 12 best-effort connection requests. The larger network type comprised a 24 node, 48 node connection-request problem which involved 24 QoS connection requests on average. These will commonly be referred to as 12 x 24 and 24 x 48 problem sizes. The problems were generated using the randomized topology generator, described previously, with the priorities assigned randomly during the generation of a problem.

The one thing that is not known well about these problems is the maximum number of connections possible (with priority respected) for a given problem, that is, the optimal solution performance. Finding this theoretically optimum solution requires an algorithm such as Branch and Bound, which considers all potential solution routes and hence can identify the global optimum. Such an algorithm yields the best case solution but at a great computational cost. Looking at the 12 x 24 problem 3 as shown below in Figure 5.1, we see that the branch and bound solution was able to find one additional priority 1, thereby trumping all other solutions.

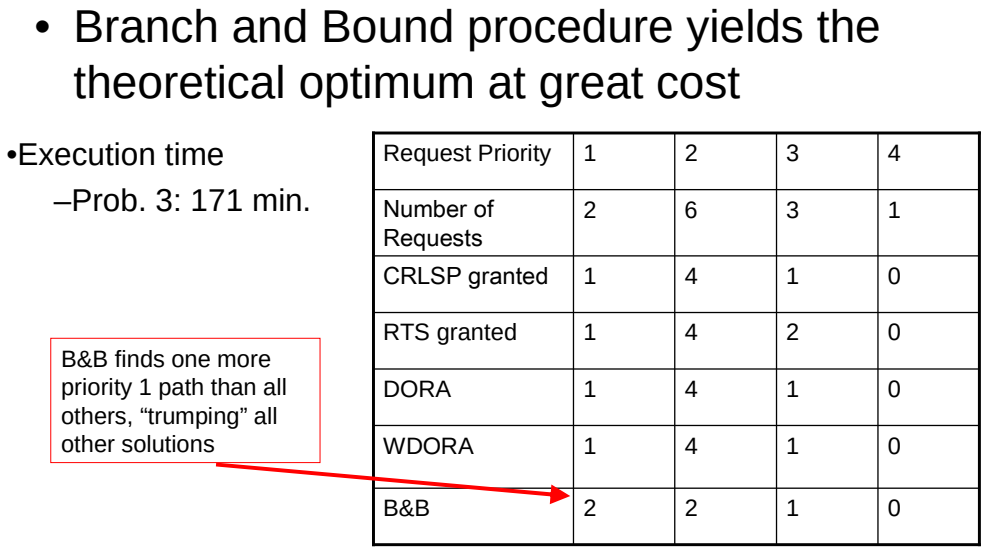


Figure 5.1 - Branch & Bound Solution

The cost to compute this solution was 171 minutes, impractically long in terms of real-time network routing. Some other problems in our test set had branch and bound run times that could be on the order of years, due to the complexity of the larger 24 x 48 networks. Since it was impossible to compute the optimum solution for each problem to be used as a reference for performance, classical routing was used as the baseline for comparison in the study that follows.

This optimal solution, given above, also serves to illustrate the fact that not all connection requests can in general be fulfilled for a given problem, and hence choices must be made among the connection requests to be granted. The routing algorithm should be designed to choose connections with a higher priority first. Thus, making an additional priority 1 connection overrides the benefits of making any number of the other connections. That is, high priority connection request fulfillment should **trump** low priority connections.

12 x 24 Test Suite Performance

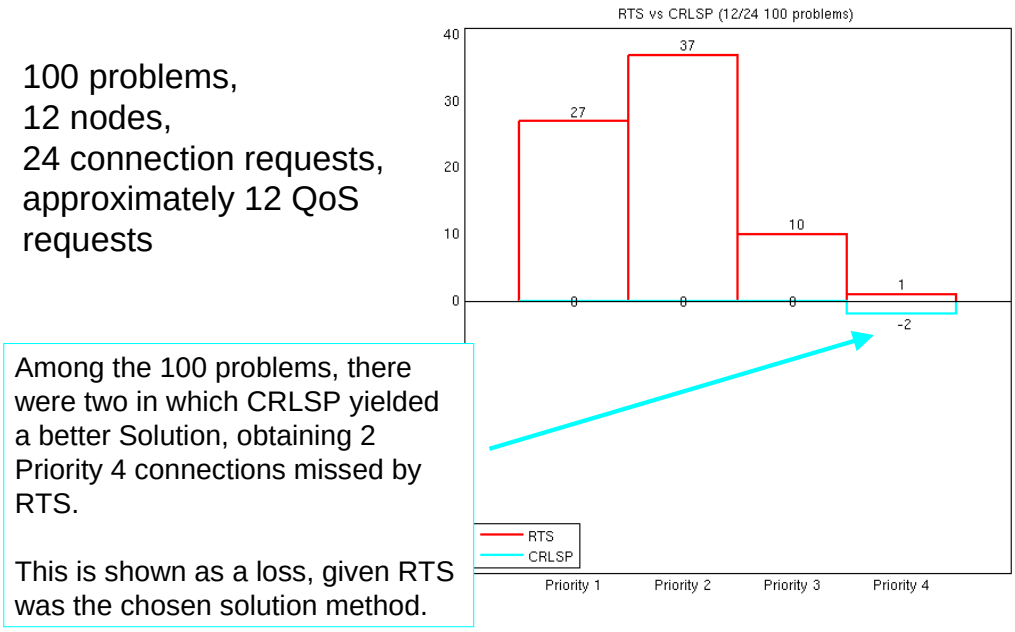


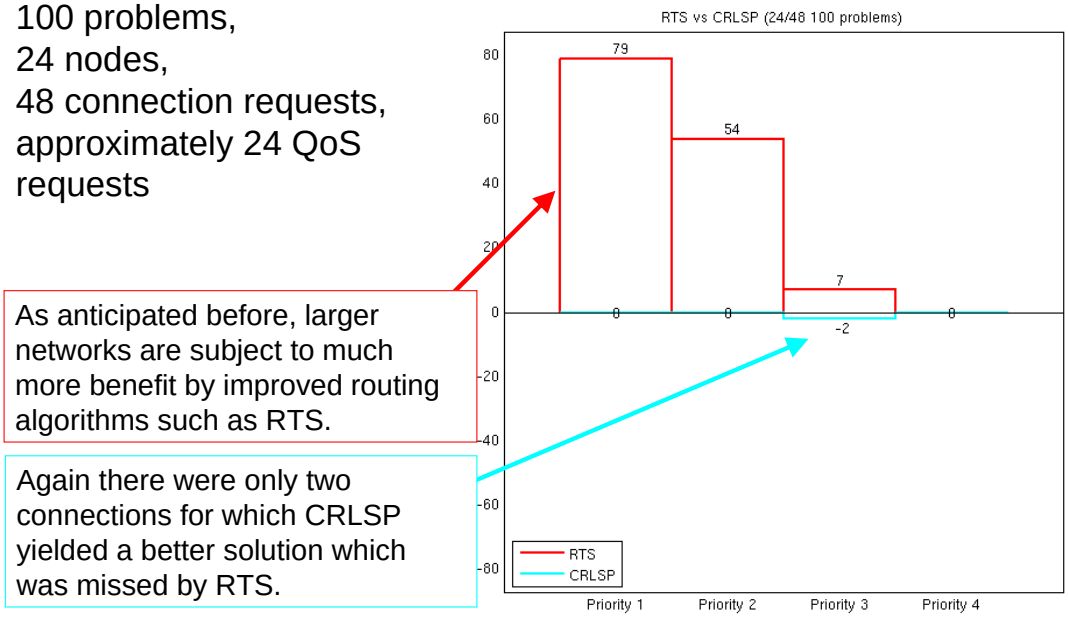
Figure 5.2 - 12 x 24 Suite Performance RTS vs. CRLSP

Applying the graphical blocking metric over an entire test suite, such as depicted in the graph in Figure 5.2, allows a meaningful comparison of algorithms and summary of performance tradeoffs. This is the graphical metric for the 12 node, 24 connection problem for a set of 100 problems. RTS results are represented in red. This graph shows that RTS finds a significant number of additional connections beyond those found by CRLSP, shown in blue. Looking at the numbers above the RTS plot, it can be seen that using RTS results in a gain of 27 priority 1, 37 priority 2, 10 priority 3 and one priority 4 with the cost being that two connections of greater priority were found by CRLSP in some test or combination of two tests that should have been found by RTS.

Similarly, the 24 node, 48 connection problems were evaluated using the same technique (Figure 5.3). It was expected that the benefits of using RTS over CRLSP would follow a similar pattern with the gains being higher given the larger number of connection attempts made.

24 x 48 Test Suite Performance

100 problems,
24 nodes,
48 connection requests,
approximately 24 QoS requests



As anticipated before, larger networks are subject to much more benefit by improved routing algorithms such as RTS.

Again there were only two connections for which CRLSP yielded a better solution which was missed by RTS.

Figure 5.3 - 24 x 48 Suite Performance RTS vs. CRLSP

The benefits for RTS were high, with 79 additional priority 1 connections being made. The tradeoff for using RTS rather than CRLSP was a loss of two priority 3 connections – a loss with no weight in the rating of performance in that these are of lower priority than any of the connections gained.

The other consideration when choosing a routing algorithm is the performance and computational complexity (hence computational resources) involved. Some of the highlights in the complexity versus performance comparison of RTS and CRLSP are:

- RTS execution time for one 12 x 24 problem: 5.6 minutes
- RTS execution time for one 24 x 48 problem: 12 minutes
- CRLSP execution time for one 12 x 24 problem: 0.056 seconds
- CRLSP execution time for one 24 x 48 problem: 0.184 seconds

- In 12 x 24 problems, CRLSP lags RTS by approximately 0.75 connections of any priority per problem or 6.25% per connection request.
- In 24 x 48 problems, CRLSP lags RTS by approximately 1.40 connections of any priority per problem or 5.83% per connection request.

The execution time for RTS was approximately linearly related to the size of the problem. With the smaller 12 x 24 connection problems, RTS was completing in an average of 5.6 minutes. The larger problems with approximately double the connections to route were computed in 12 minutes on average.

CRLSP executes on average in 0.056 seconds for the smaller problems and 0.184 seconds on the larger problems. This is dramatically faster than RTS, with slightly fewer connections routed successfully. CRLSP performance actually lags RTS by approximately 0.75 connections per problem or 6.25% per connection request. For the larger problems, CRLSP lags RTS performance by 1.40 connections per problem or 5.83% per connection request on average. Why can a simple solution such as CRLSP be this good?

> Most networks pose routing problems that are not as difficult as NP-complete: "Conditions that impact the complexity of QoS routing," F.A. Kuipers and P.F.A. Van Mieghem, IEEE Transactions on Networking, August 2005, pp. 717–730.

One could argue that with a well chosen approach mapped to a specific problem, it is possible to generate good solutions with much less complexity than a full global optimization, and that the above result indicates that CRLSP is quite close to the algorithm that does this for us.

5.1 Comprehensive Performance Comparison

Next, we will examine and analyze the outcomes of tests that span seven different routing algorithms (OSPF, OSPF+, CRLDP, OCRLSP, DORA, RTS, HYBRID), two network

problem sizes (12 nodes, 24 connections and 24 nodes, 48 connections) and from 100 to 5000 cases of each. In every case, each connection request was assigned a minimum allowed bandwidth and maximum allowed delay.

The baseline with which all other algorithms were compared was OSPF, which is the most common algorithm in use in the Internet today. When OSPF was used, the usual shortest path routes were found for each request based upon the pre-computed connectivity matrix and cost functions that were taken as the delay of each link. Reservations are then attempted (in order of the randomly generated requests) based upon remaining (previously unreserved) bandwidth. This again replicates the behavior of today's predominantly OSPF-based, priority-agnostic, RSVP/IntServ-based system for QoS reservation on QoS-aware segments of the Internet.

In Table 5.1 we show the results upon applying the OSPF-with-reservations algorithm to 100 cases each of our two problem types. The total number of connections at each priority level that were granted are given in this table.

Problem Size	Priority 1	Priority 2	Priority 3	Priority 4
12/24	70	127	143	76
24/48	114	241	259	128

Table 5.1 - Connections Satisfied

The blocking metric, which will show important (that is, passing the trumping criterion introduced earlier) improvements as various levels of optimization are introduced, is shown in the following figures.

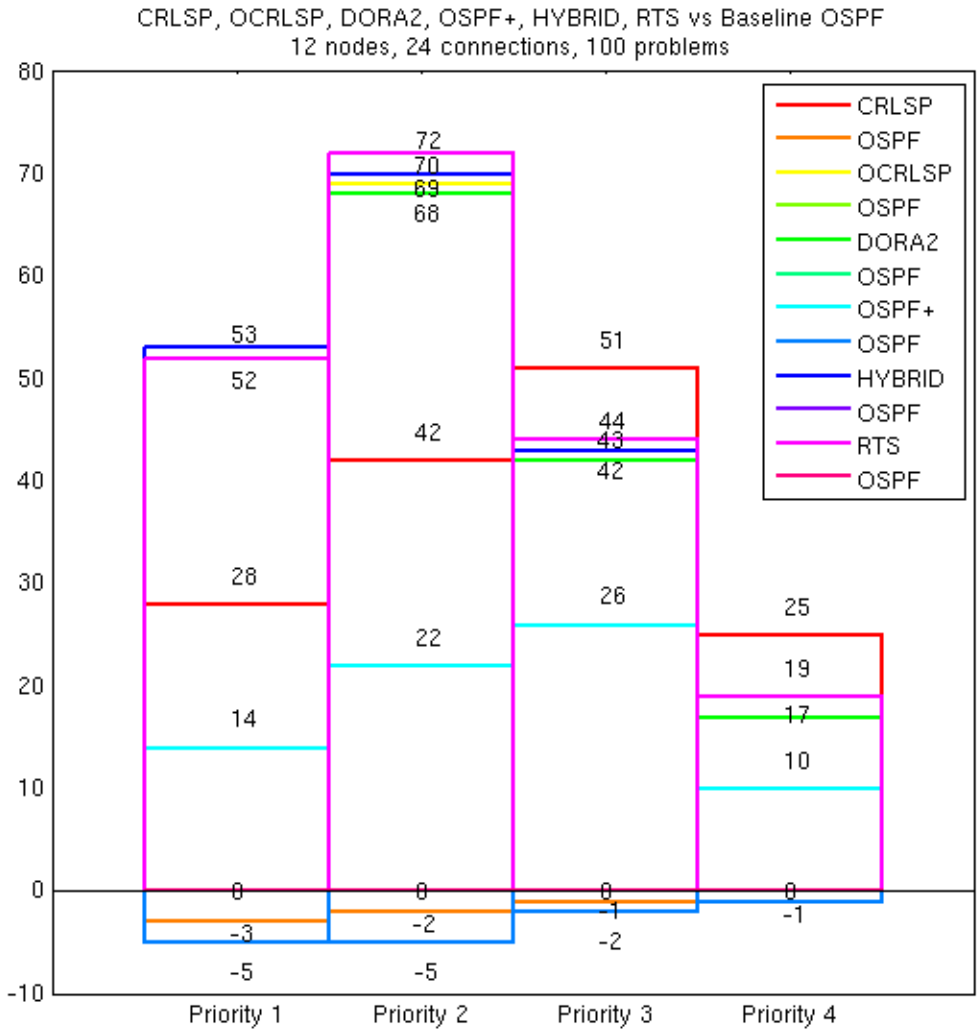


Figure 5.4 - 12 x 24, 100 Problems, All Algorithms

There are clear increases in utilization as higher levels of optimization are introduced. The lowest level of optimization introduced is to be found in OSPF+, which uses a DiffServ-based approach to apply some knowledge of the network link bandwidth to determine which of several bandwidth-qualified shortest paths to use. This additional information regarding bandwidth increases the number of priority 1 connections by 14 for the 12 node, 24 connection problem set and, furthermore, does this at no tradeoff for lower priority connections. At all levels, a total of 72 connections are added to the 416 established under OSPF, for an inprovement of 17.3% overall, including the very

important increase of 20% more priority 1 connections. **DiffServ can play an important role in capturing improved MLPP QoS performance.**

The red bar above the OSPF+ bar in the above graph represents CRLSP, which recalculates the shortest available route for each connection based upon remaining bandwidth. It will preempt lower priority traffic connections previously committed and hence will route around congestion in the network to a limited degree. This increases the number of priority 1 connections made over OSPF by 28. Thus, 40% more priority 1 connections are made, twice the improvement of OSPF+. **Thus, at the cost of maintaining global distribution of network state or, equivalently, of using a centralized routing host, a very significant improvement over DiffServ performance can be realized.**

To read the next few graph values, one must be aware that the graphs were painted in the order of opimization level, from lowest to highest. Thus, if a color appears to be missing, it is under that of the next higher priority level result shown. Hence, the yellow line indicating OCRLSP is under the purple line of RTS as is also DORA2 in the priority 1 column of the above graph.

The next level of optimization considered, Ordered CRLSP, uses some global optimization in the form of a complete recomputation of all paths with each new connection request to increase the effectiveness of CRLSP. OCRLSP applies sorting of all connections to be made by precedence followed by recomputation via the CRLSP method. It adds no additional burden to the distribution of information or centralization, but adds to the computational burden involved with each connection establishment and to the communication burden of having to inform the entire nework of the required routing changes. This strategy could not be applied on a global scale but is practical for implementation to address a HC3 convergence layer due to its small size. OCRLSP found means to provide 52 additional priority 1 connections, an improvement of 72%, nearly doubling the performance increase obtained by CRLSP. **Thus, at a cost of**

greater per-route computational burden and additional inter-router communication for each route change, yet another large performance advantage can be realized.

DORA2 is a two-pass algorithm that first solves the routing problem for each connection without competition, gathers statistics on the usage of links, and then resolves the problem for each connection (using priority ordering) with a metric that reflects the usual constraint information and the statistics found in the first pass. Though not dramatic, this does present an increase in computation over OCRLSP. As can be seen in the figure, no advantage over OCRLSP was found at priority 1, and it fell slightly behind OCRLSP at the priority 2 level. **There appears to be no advantage to the DORA2 scheme, a two pass solution, over OCRLSP.**

RTS introduces a significantly higher level of global optimization. Using a solution found with Dijkstra's algorithm (the same as used by OSPF) as an initial routing solution, it performs an extensive local optimization of local path alternatives. As can be seen in the graphs, there is no advantage over OCRLSP for priority 1 traffic. A small improvement is obtained at the priority 2 level. As will be seen in the concluding section, the cost of RTS is very high compared to the preceding algorithms and is not compensated by a significant performance improvement. **There appears to be no reason to apply costly local path exploration optimization in HC3 networks.**

The final optimization technique considered is the Hybrid method. This uses the two-pass DORA2 solution as the initialization for an RTS optimization. The notion here is that RTS is more likely to explore the correct locale of the solution space if initialized by an already nearly optimal solution. Results do in fact show in this case that a single additional priority 1 connection is obtained. Hence, the Hybrid method achieves a 75.7% improvement over OSPF for priority 1 connections versus a 74.28% improvement obtained by OCRLSP. However, this small improvement was obtained at a cost of 3450 times greater computation time! **It appears that almost all optimization in HC3 convergence layer networks is achievable at low cost with a remaining small improvement achievable at unwarranted cost.**

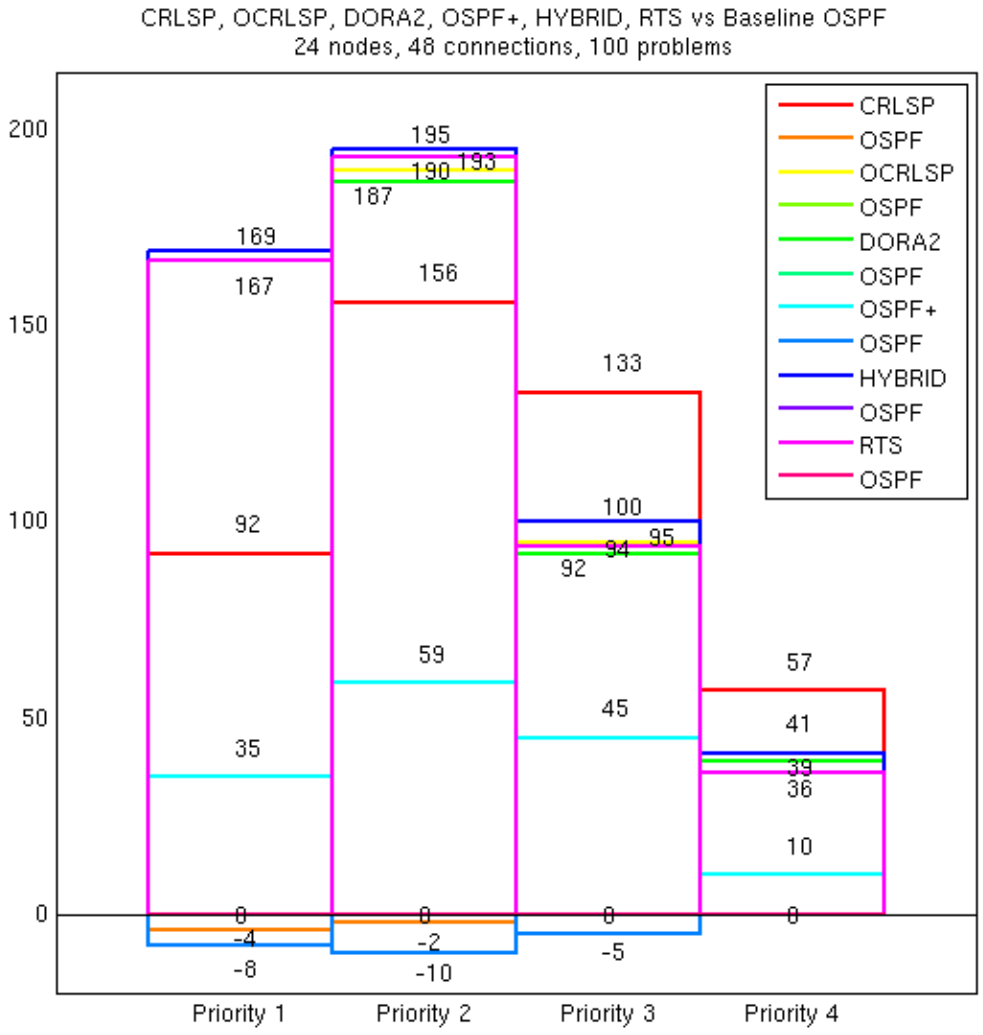


Figure 5.5 - 24 x 48, 100 Problems, All Algorithms

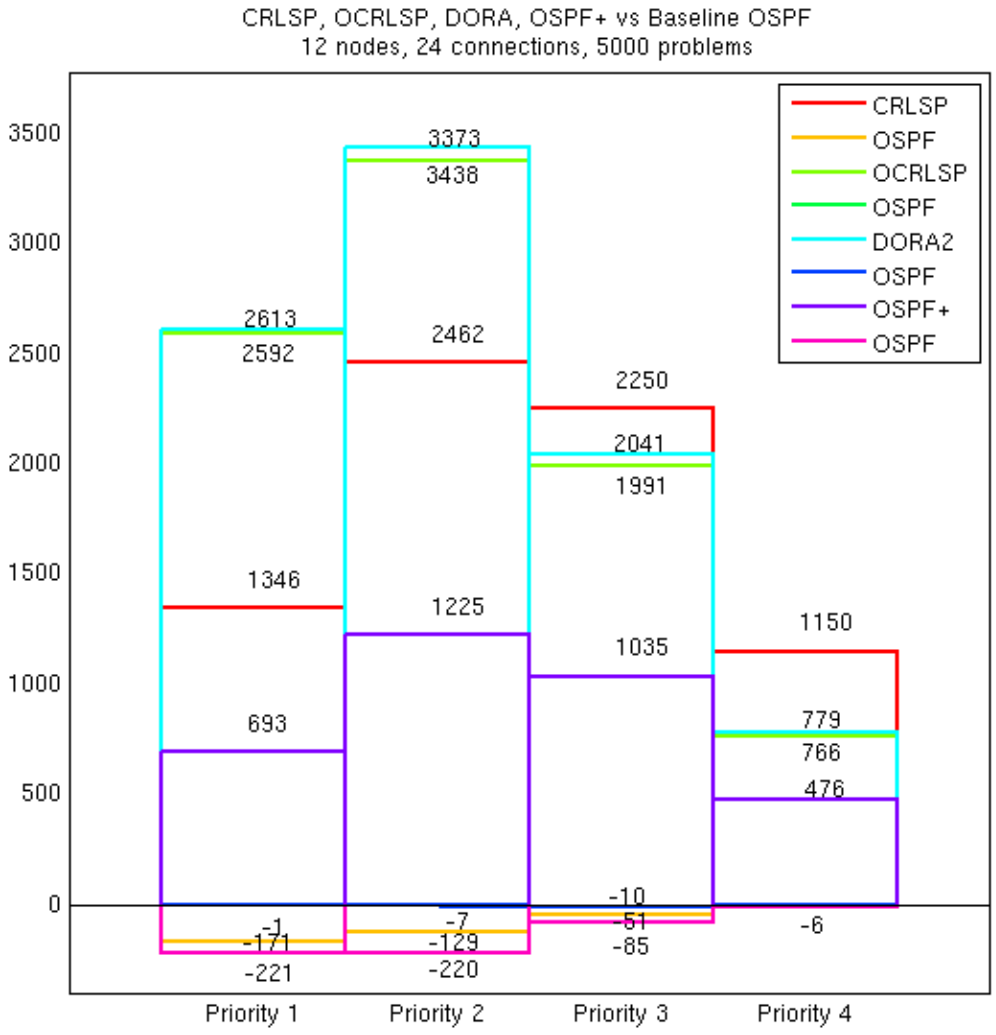


Figure 5.6 - 12 x 24, 5000 Problems

Several more tests were conducted to obtain evidence that our observations were statistically sound and that they are not restricted to this specific problem size. The next graph indicates our results by applying all but RTS and Hybrid methods to 5000 problems of the 12 x 24 size. The latter algorithms were not introduced into this test owing to the impractical time of execution of a test of this size given their computational burden. As can be seen, there is no qualitative difference discernable between these results and those we have been discussing for 100 cases. **There is strong evidence that our conclusions are statistically significant.**

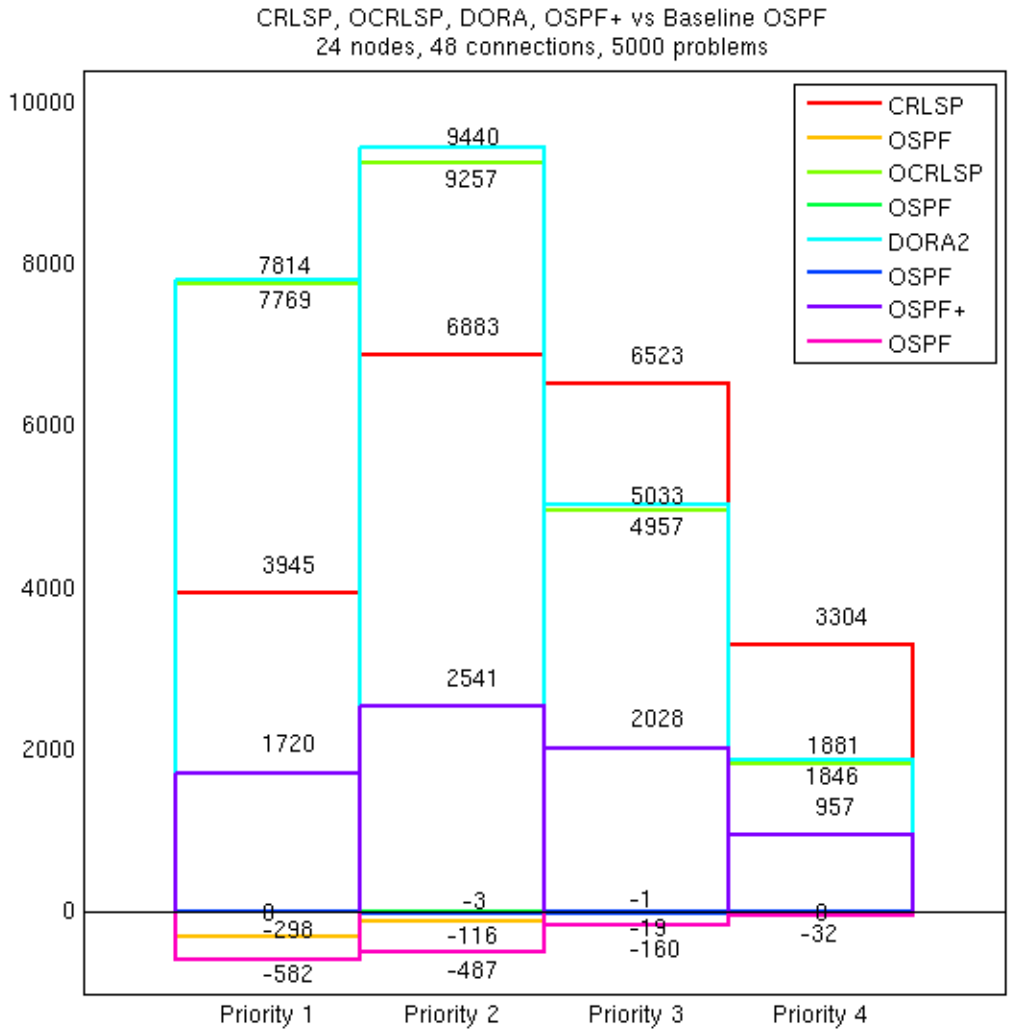


Figure 5.7 - 24 x 48, 5000 Problems

Next, we tested the behavior of these solutions with respect to network size. The following graphs depict the trumping metric evaluations obtained upon doubling the network size. Once again, 100 cases were executed for all algorithms, and 5000 cases for those for which it was computationally feasible. In both of these results we see again no qualitative differences in the overall behavior (ranking, and proportional performance gains). **Thus, our conclusions are robust with respect to problem size in the vicinity of network sizes that might be encountered in an HC3 convergence layer network.**

6.0 Interpretation of Results

OSPF is the default routing algorithm used in the Internet today. It is based on use of a static (that is, routes are assigned on the basis of network topology and assigned metrics and not dynamically on the basis of the state of admitted connections) shortest path algorithm with no bandwidth or delay constraints. QoS-based connection requests are evaluated and admission is granted by a process that tests admissibility of these static routes given the current admission state. In the following, we will denote this as a "Stateless, Non-Preemptive, Static Routing" algorithm.

The optimal (and impractical) routing algorithm would be a "Stateful, Preemptive, Globally Optimized" algorithm. That is, for each new connection request, it would re–solve for the best path assignments. This solution would take into account all possible path assignments (global) for all the currently granted connections (the connection state) and the newly requested connection. It would then dynamically reassign paths as necessary and pre-empt lower priority connections to optimize the blocking metric while maintaining the bandwidth and delay constraints of the QoS parameters.

Practical multi-objective optimization involves relaxing the global nature of the optimal solution. Our evaluation ultimately spanned a set of seven levels of optimization, ranging from the Stateless, Non-Preemptive, Static Routing of OSPF to techniques that explore many, but by no means all, path assignments, seeking a local optimum in the vicinity of solutions that arise from heuristics intended to quickly identify reasonable solutions. These heuristics involve the use of precedence-sequential optimization (that is, assign all highest precedence paths first, followed by each lower precedence class) and two-pass approaches in which the metrics determined by bandwidth and delay constraints are

supplemented by information about the "popularity" of certain links given optimum routing of each connection in the unoccupied network.

The algorithms which were tested, as described in the previous section, can be described by their major routing attributes and are seen to form a hierarchical set. As listed below, this hierarchy is

1. Stateless, Non-Preemptive, Static Routing (OSPF)
2. Stateless, Non-Preemptive, Multi-Static Routing (OSPF+)
3. Stateful, Path-Preemptive, Time-Sequential Routing (CRLSP)
4. Stateful, Preemptive, Precedence-Sequential Routing (OCRLSP)
5. Stateful, Preemptive, Two-Pass, Precedence-Sequential Routing (DORA2)
6. Stateful, Preemptive, Limited Non-Sequential Routing (RTS)
7. Stateful, Preemptive, Two-Pass, Limited Non-Sequential Routing (Hybrid)

The cost of the various levels of optimization varies greatly, with the greatest computation costs entering upon the introduction of limited non-sequential routing, that is, the operation of local optimization of the path assignments about the initial path solution. Hence, there is great value in discovering the gains in routing performance to be obtained at each level.

Table 6.1 shows the amount of time taken, in seconds, to execute a set of 100 routing problems of the given size. Note that the variance of execution time due to local processor conditions (background jobs) exceeded the differences in computation time of the OSPF, OSPF+, CRLSP, and OCRLSP algorithms. Yet, as we saw in the previous section, over this span of algorithms, nearly a doubling in successful high priority connections established was achieved.

Execution Time (100 problems) sec.			Performance Summary (100 problems) % Increase in Priorty 1 QoS Connections	
Algorithm	12 / 24	24 / 48	12 / 24	24 / 48
OSPF	5.88	21.11	Baseline	Baseline
OSPF+	5.66	19.86	12.86 %	23.68 %
CRLSP	5.63	18.38	35.71 %	77.19 %
OCRLP	5.43	18.91	74.28 %	146.49 %
DORA2	26.03	107.70	75.71 %	148.25 %
RTS	34035.00	72342.00	74.29 %	146.49 %
Hybrid	18734.00	38486.00	75.71 %	148.25 %

Table 6.1 - Execution Time and Performance Summary

We will summarize the significant performance and execution time properties of these algorithms.

DORA2 required approximately five times more computation than the preceding algorithms but demonstrated an advantage in finding priority one connections. Other algorithms would find additional lower priority connections, which would balance out the total connections if not for the trumping metric. DORA2 also showed great promise as an initialization routing solution for the Hybrid method, allowing it to find additional routing solutions that were missed in the RTS solutions.

The RTS and Hybrid algorithms are seen to present computational burdens that are unsupportable with today's routing processors. But our previous results also showed that the gains to be captured by increases in optimization are small.

Our results have shown that almost all performance advantage due to optimization in HC3 networks is attained upon the introduction of Stateful, Preemptive, Precedence-Sequential Routing, that is, with the use of OCRLSP, or level 4 in our hierarchy. Thus, for a very small increase in routing complexity, results that approach the highest

otherwise attained performance can be obtained. This result has immediate impact on the design of routers for HC3 applications as it places the problem of obtaining solutions nearly as optimal as that obtainable with impractical levels of computation for QoS constrained routing within the realm of feasible implementation.

A question raised by this result is whether it has a theoretical basis. Very recently, a published work demonstrated that this is indeed the case. The paper "Conditions that impact the complexity of QoS routing" (F.A. Kuipers and P.F.A. Van Mieghem, *IEEE Transactions on Networking*, August 2005, pp. 717-730) provides theoretical evidence that most networks pose routing problems that are not as difficult as NP-complete. **It appears that our results, for HC3 topology networks, confirm this overall observation that, while some network problems are very difficult, most are not, and we identify the specific level of optimization needed to access the practically achievable performance improvement.**

7.0 Conclusions

7.1 "Grand Challenge" of Network Centric Warfare

Upon delivering our final report presentation to our sponsor, the WPI team noted that in a recent editorial in the *IEEE Communications Magazine*, the following gauntlet is cast:

> From a development perspective, there is no underlying "network theory" for multihop large-scale heterogeneous tactical mobile networks on which to pin our network designs. We are quite literally designing networks and components in the dark with respect to understanding how their performance compares to a theoretical limit. There is a reluctance … to invest in large-scale, scientifically sound (i.e., repeatable, calibrated, scalable) experimental testbeds where real platforms are outfitted with experimental components and a tractable assessment of their performance is scientifically studied.
>
> Until these challenges are seriously addressed by the military community, little or no real progress will be made in achieving the vision of ubiquitous voice, video, and data being available to the war fighter in the field.
>
> "Network-Centric Military Communication," Guest Editorial, C.A. Nissen, T. Maseng, *IEEE Communications Magazine*, Nov. 2005, pp. 102–104.

We further noted that this project took direct aim at the problems identified in this challenge and made significant contributions to our knowledge about them. In doing so,

we also created a suite of tools and techniques that substantially speed such investigations as may be carried out in the future.

7.2 Outcomes

The significant outcomes of this project may be summarized as follows:

- Created a versatile, scalable, reusable testbed and simulator for evaluation and comparison of alternative routing algorithms.
- The testbed reflects, validates and demonstrates TCA best-practice combination of QoS routing and MPLS traffic engineering implementation.
- Created a suite of tools for the automated generation of random networks of a given topology and link bandwidths and delay randomly assigned. Similarly, random connection requests with priority of connections and required QoS parameters were also drawn from given random processes. Other tools provide for the automated execution of large scale Monte Carlo tests of these networks and problem sets.
- Thoroughly evaluated best-of-class QoS constrained shortest-path routing methods, multi-objective optimization methods and recent fast heuristics against each other and the Internet-standard OSPF algorithm.
- Identified OCRLSP as viable for practical convergence routing.
- Confirmed recent theory that, while optimal QoS aware routing solutions are in general NP-Complete and hence not practically achievable, nearly optimal solutions are easily within reach of current processor capabilities for limited span networks (such as the HC3 convergence layer) given acceptability of higher communication burdens owing to the need to maintain global state information.

7.3 Future Investigations

The results of this research immediately suggest the following areas for fruitful future research.

7.3.1 Optimality of Distributed Dynamic Routing

A paper delivered at INFOCOM ("Joint optimal scheduling and routing for maximum network throughput," E. Leonardi, et al., March 15, 2005) offers the following theorem:

> Combined behavior of dynamic routing and scheduling algorithms based upon link state information, with no knowledge of the average traffic pattern can achieve the same network throughput as optimal centralized routing and scheduling algorithms with complete information on the traffic pattern.

If what is claimed in this paper is true and extends to the case of QoS constrained traffic, then there would be means to achieve the same gains that we obtained in this work through application of the OCRLSP routing algorithm without as large a communications burden and without centralization.

This project included a small investigation of stigmergy-based distributed, dynamic route optimization. However, it was found that current work in distributed agent-based routing does not address QoS constraints. It appears to us that stigmergy-based QoS routing may be a viable approach to achieve distributed dynamic routing for convergence layer networks and other larger networks.

7.3.2 Ad Hoc Network Extension

The HC3 Convergence Layer network typically interfaces with a mobile ad hoc network. It would be fruitful to implement an ad hoc network mobility test component for our testbed and then investigate the possibility of extending the OCRLSP technique into the ad hoc network itself. Ad hoc networks suffer from many of the same routing problems as the convergence layer with respect to difficulties induced by QoS and MLPP

constraints. Hence, our success at the HC3 layer offers the possibility of improved MANET performance at a similar low routing cost.

8.0 References

[ACG] Adami, D., Callergari, C., Giordano, S., Mustacchio, F., Pagano, M., and Vitucci, F., "Overview of the RSVP-TE Network Simulator: Design and Implementation," Dept. of Information Engineering, University of Pisa, Italy.

[AC00] Ahn, G., and Chun, W., "Design and Implementation of MPLS Network Simulator Supporting LDP and CR-LDP," *icon*, p. 441, Eighth IEEE International Conference on Networks (ICON'00), 2000.

[AC01] Ahn, G., and Chun, W., "Design and Implementation of MPLS Network Simulator (MNS) Supporting QoS," *icon*, p. 694, 15th International Conference on Information Networking (ICON'01), 2001.

[BT94] Battiti, R., and Tecchiolli, G., "The Reactive Tabu Search," *OSRA Journal on Computing,* Vol. 6, No. 2, pp. 126–140, 1994.

[BSI02] Boutaba, R., Szeto, W., and Iraqi, Y., "DORA: Efficient Routing for MPLS Traffic Engineering," *Journal of Network and Systems Management*, Vol. 10, No. 3, Sept. 2002, pp. 309–325.

[BSI02A] Boutaba, R., Szeto, W., and Iraqi, Y., "Dynamic Online Routing Algorithm for MPLS Traffic Engineering," *Networking 2002*, LNCS 2345, pp. 936–946, 2002.

[CEH06] Cyganski, D., Esterline, A., Homaifar, A., Clay, L., and Farmer, J., "Multi-Objective Routing Control and Optimization for HC3 Networks." Final report submitted to Raytheon Corporation upon project completion, May, 2006.

[Gen03] Gendreau, M., "An Introduction to Tabu Search," *Handbook of Metaheuristics*, Norwell, MA: Kluwer Academic Publishers, 2003. Ch. 2, pp. 37–54.

[MAB05] Maalaoui, K., Abdelfettah, B., Bonnin, J., and Tezeghdanti, M., "Performance Evaluation of QoS Routing Algorithms," *Computer Systems and Applications*, 3rd ACS/IEEE International Conference, Cairo, Egypt, January 2005, pp. 66–69.

[MOY98] Moy, J., "OSPF Version 2," RFC 2328, April 1998.

[NACP06] "Network Architecture and Connection Protocol – 1.1 Routing Problem Description Language for NCAT/WPI Research." Unpublished internal document related to Raytheon sponsored HC3 project, May, 2006.

[PC97] Park, V., and Corson, M., "A highly adaptive distributed routing algorithm for mobile wireless networks," *Proceedings of INFOCOM'97*, April 1997, pp. 1405–1413.

[PB94] Perkins, C., and Bhagwat, P., "Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers," *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, August 1994, pp. 234–244.

[SM00] Singh, A., and Mittal, G., "QoS and Traffic Engineering: MPLS, DiffServ and Constraint Based Routing," Project report, Department of Computer Science & Engineering, Indian Institute of Technology, May, 2000.

[TGP01] Trimintzios, P., et al. "Engineering the Multi-Service Internet: MPLS and IP-Based Techniques," Proceedings of IEEE International Conference on Telecommunications, Bucharest, Romania, June, 2001.