Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

CONFIGURATION SYNTHESIS FOR A HETEROGENEOUS BACKBONE CLUSTER AND A PC-INTERFACE NETWORK

Joseph Betser
Alberto Avritzer
Jack W. Carlyle
Walter J. Karplus

September 1988
CSD-880074

Configuration Synthesis for a Heterogeneous Backbone Cluster and a PC-Interface* Network†

Joseph Betser, Alberto Avritzer‡
Jack W. Carlyle, Walter J. Karplus
UCLA Computer Science Department
Los Angeles, CA 90024

August 5, 1988

Abstract

In this paper we address the design of Locus[1] [PopeWalk85] family networks rendering PCI service. Given are the expected user workload, the hardware costs and the performance constraints. The workload consists of three classes of users: *Interactive, High-Interactive, and Communication-Intensive.* The number of PC's is given and is equal to the number of users. We show that the problem can be reduced to a discrete Capacity and Flow Assignment (CFA) problem. The backbone capacity assignment is inspired by the Lagrangian Decomposition Approach [Fox66]. It starts from a backbone capacity assignment which matches the initial flow and chooses a backbone server upgrade that gives the greatest response time reduction per Dollar. At this point we apply the flow deviation method and iterate until the constraints are met. We present several computational examples of network configuration synthesis, that emphasize the significance and generality of the results obtained.

*PC-Interface (PCI) is a trade mark of Locus Computing Corporation (LCC)

†This work was sponsored in part by the UCLA-IBM Joint Study D850915

‡Sponsored in part by CNPq-Brazil grant 200366/86

[1]Locus is the predecessor to the forthcoming IBM AIX/370

1 Introduction

Locus is the predecessor of the newly announced IBM AIX/370. It is a Unix compatible distributed operating system that supports a large variety of architectures. Some of the mainframe architectures supported are 4300, 9370 and the 3090. A set of such backbone servers is called a "Transparent Computing Facility" cluster (TCF cluster). Personal workstations such as high-end PS/2 and PC/RT are supported as well, and they can be used to access a TCF backbone cluster.

The UCLA School of Engineering and Applied Sciences Network (SEASnet) is the primary computing facility of the school. SEASnet also serves as a pilot/demonstration center, as it is the most experienced production environment for pre-AIX/370. SEASnet demonstrates a high degree of heterogeneity, as it consists of two 4361s and a 4381 as backbone servers (as well as a number of DEC VAX/750s SUNs FPS and others as in Figure 1). On the client side, SEASnet users use primarily IBM PC/ATs running DOS to access the system (ascii terminals and PC/RTs are also available).

In this work we shall concentrate on the PC to backbone-cluster connection. This connection is accomplished by an operating system bridge called PC-Interface (PCI). PCI allows a PC running DOS to transparently access an AIX or Unix file system through the so-called network drive (Drive E:).

The problem of optimal synthesis of such heterogeneous systems is both compound and difficult. There are several subparts to this general problem, namely - *component measurement and characterization, load and system model construction, and lastly, the sub-optimal configuration synthesis algorithms.* The major contribution of this paper is the last among the three - the configuration synthesis. This contribution is very general, as the presented algorithms can be applied to many resource systems which need to be optimized.

We begin this paper by providing the necessary context that inspired this work. We shall describe the system that was used to measure and characterize the components of the synthesized configurations in the computational example we present. Hence, further system description and workload characterization will be provided in section 2.

We then proceed with an exposition of our top-level approach in section 3. We describe how the backbone cluster synthesis problem can be mapped into an extension to the Capacity and Flow Assignment (CFA) problem.

In section 4 we present the extended sub-optimizing CFA algorithms we developed for the synthesis problem. Both Capacity Assignment and Sub-Optimal Session Assignment algorithms are outlined.

In section 5 we apply the algorithms to three different configuration sets. We derive sub-optimal session assignments, and ultimately present the cost/performance curves that readily yield the configurations of choice.

In section 6 we outline analysis insights and provide general conclusions for this work. In section 7 we suggest additional directions and further extensions to our work.

We proceed with the description of the system that inspired this work.

2 System Description and Workload Characterization

SEASnet is an Ethernet based network, serving the UCLA School of Engineering. Figure 1 illustrates the general layout of SEASnet through the school. Our focus for this work is in the PCI networking option. PC-Interface allows any of the 100+ PCs in the school to access any of the backbone machines. Figure 2 shows the part of SEASnet on which we focus, and from which we derive the computational example of section 5. The backbone servers provide extensive file service for class instruction, as well as research work. We present here algorithms that optimize resource utilization, as well as synthesize systems with more (or fewer) available resources.

Extensive measurements were carried out on SEASnet in order to parameterize its resources and construct a queueing network model. Most of these measurements are reported elsewhere [Betser87, BeLaCaKa87]. An important benchmark was the delay incurred in a 1 Mega-Byte file transfer. This is a crucial yardstick for a file service oriented system.

Based on our study of the SEASnet user behavior [Betser88], we introduced three classes of users in our characterization:

1. Interactive (Int) : Short interactive commands interleaved with 1 min think time.

2. High Interactive (HI) : Short interactive commands in succession with 1-2 sec think time.

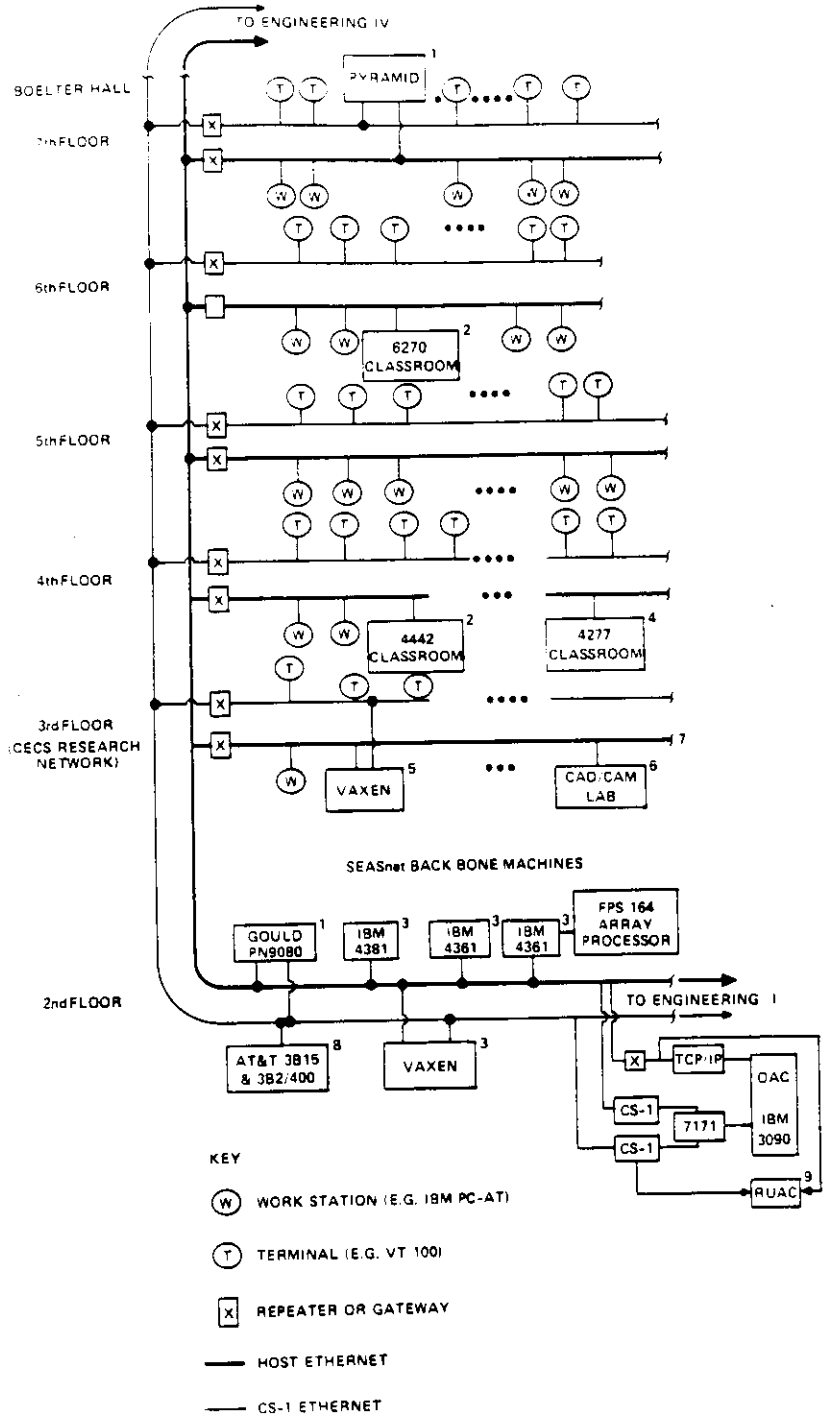


Figure 1: SEASnet - General Layout within the School of Engineering

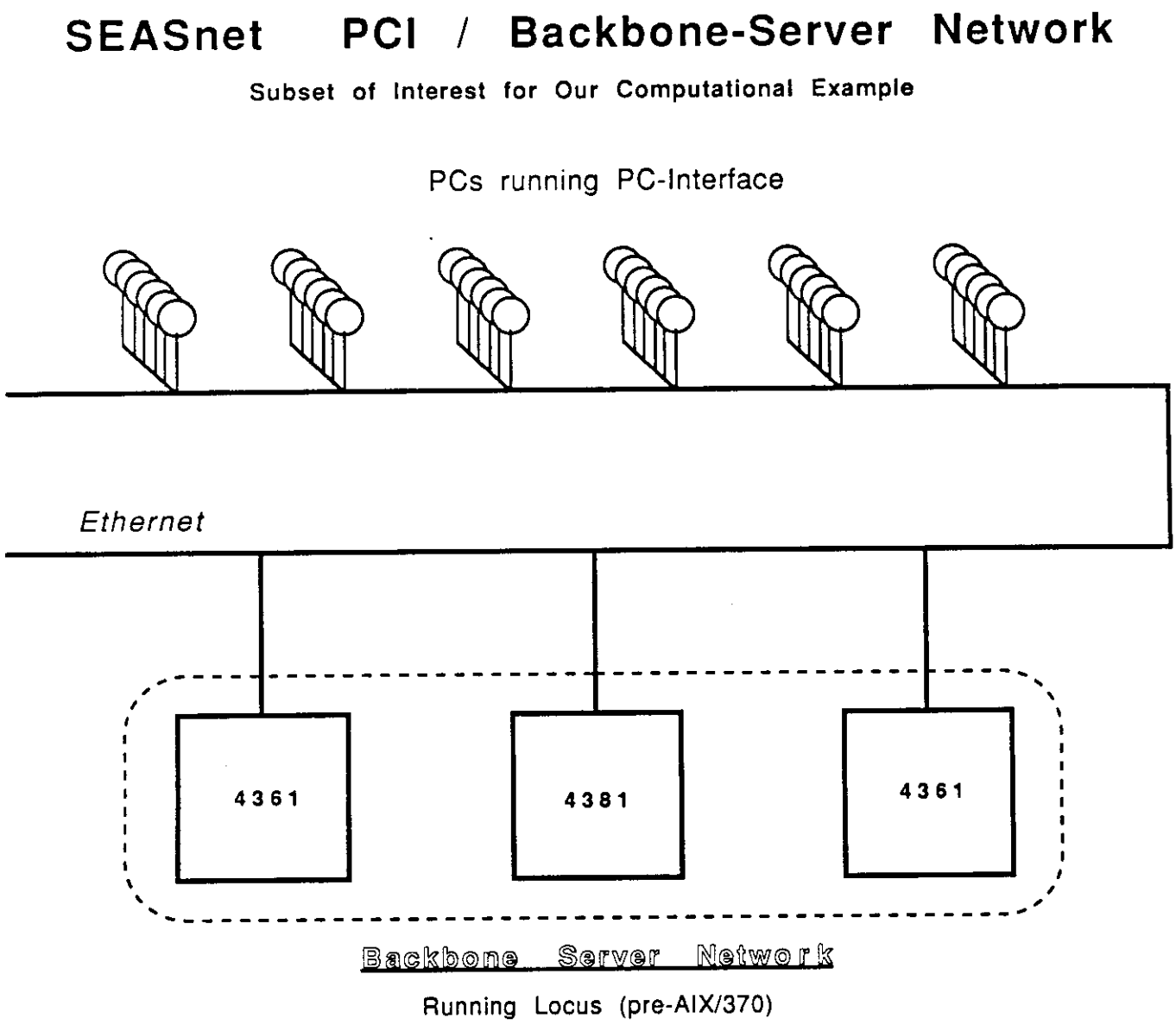


Figure 2: SEASnet - PCI / Backbone-Server Network

3. Communication Intensive (CI) : Massive continuous file transfer from backbone servers to PCs.

The performance figures for workloads with customers of different classes were derived through measurements and simulation. With the aid of many such experiments, under various configurations and loads combinations, we have identified the contention points for SEASnet. We then constructed the queueing network model shown in Figure 3. This model describes a Locus backbone server supporting a variety of PCI sessions.

We model the primary governing parameters of the Locus-PCI operation paradigm. It is important to note that we present the contention points as the CPU speed and I/O capacity of the backbone servers. We also present a bottleneck at the PC end, in the way of the Ethernet interface speed. We have observed that these are the primary resource contention modules within this architecture[2].

The tuning and validation of of this model was a long iterative process, and is reported in detail in [Betser88]. We present in Figure 4 some recent comparisons between measured results and simulated results generated by the queueing network model.

Our model was used to simulate many of the configurations that we reported in the computational example of section 5. These extensive simulations were used to construct the delay space tables. These tables were used as input to the optimization and synthesis algorithms we subsequently conducted.

Having provided this motivation, we proceed to the description of the optimizing algorithms.

3 Reduction to the Capacity and Flow Assignment Problem

For a given a user load, the *synthesis of heterogeneous backbone clusters for PCI networks* consists of two major decisions :

[2]The Ethernet is an example of a non-bottlenecking resource. We have never gotten even close to full Ethernet throughput during our extensive experimentation

Queueing Network Model for a Server and PCI Sessions

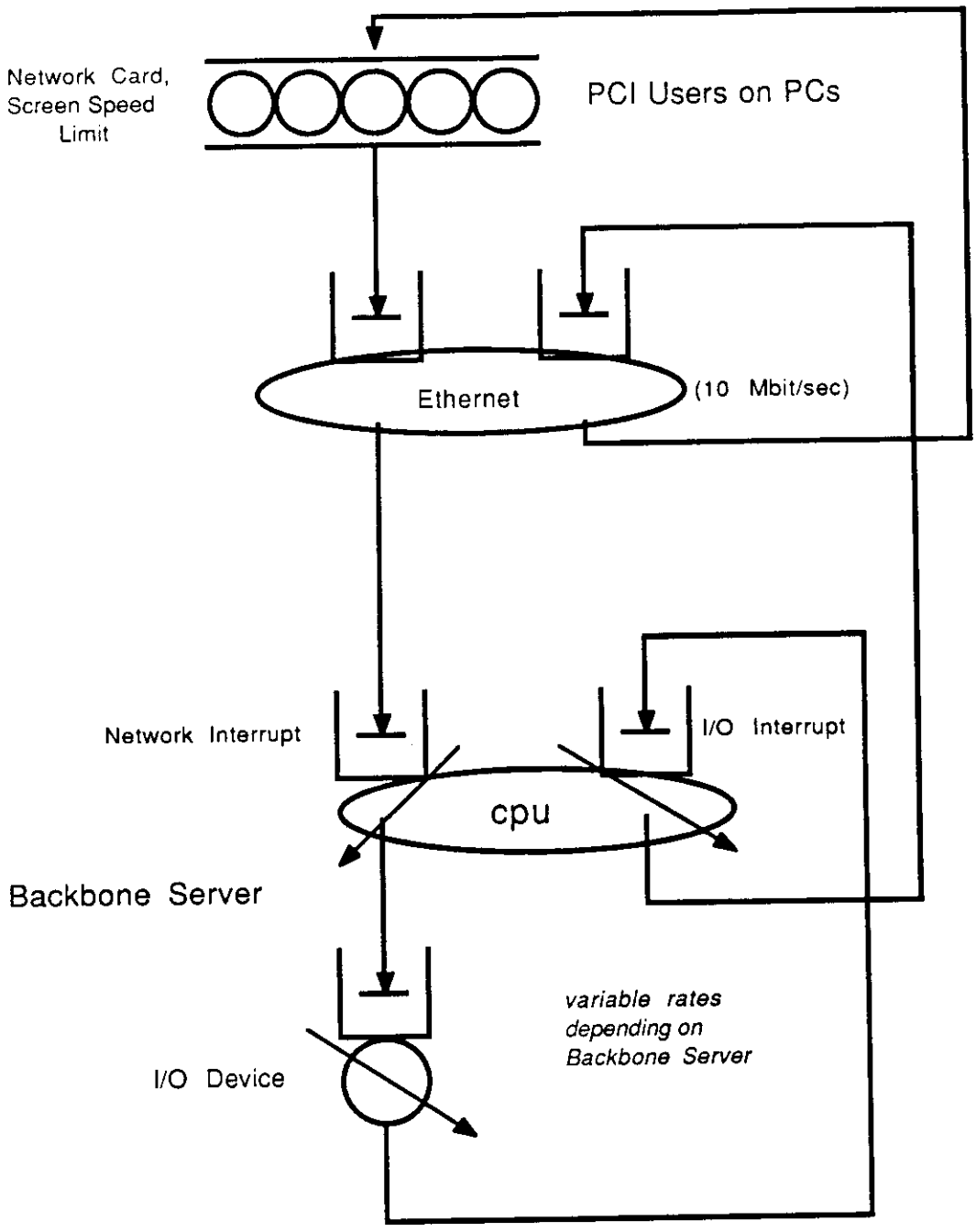
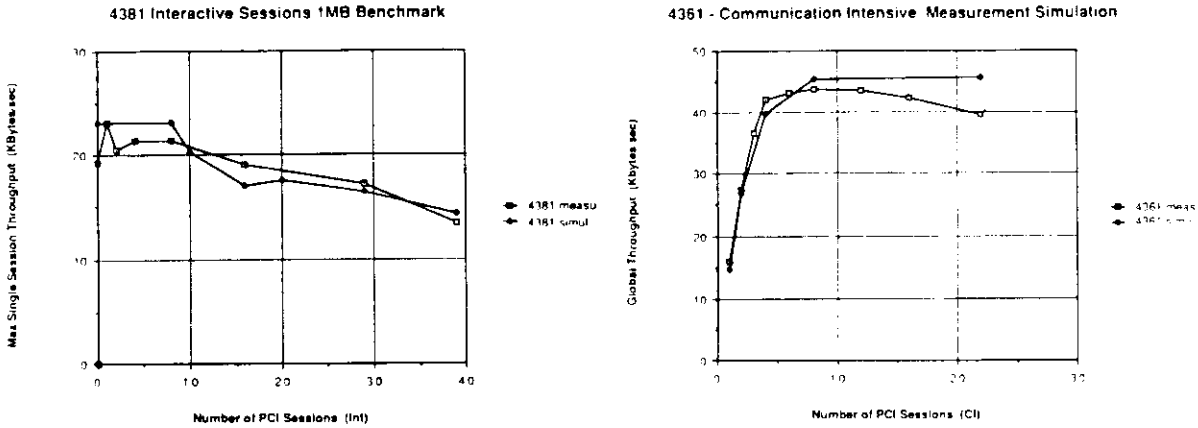


Figure 3: Queueueing Network Model for a Backbone and PCI Sessions



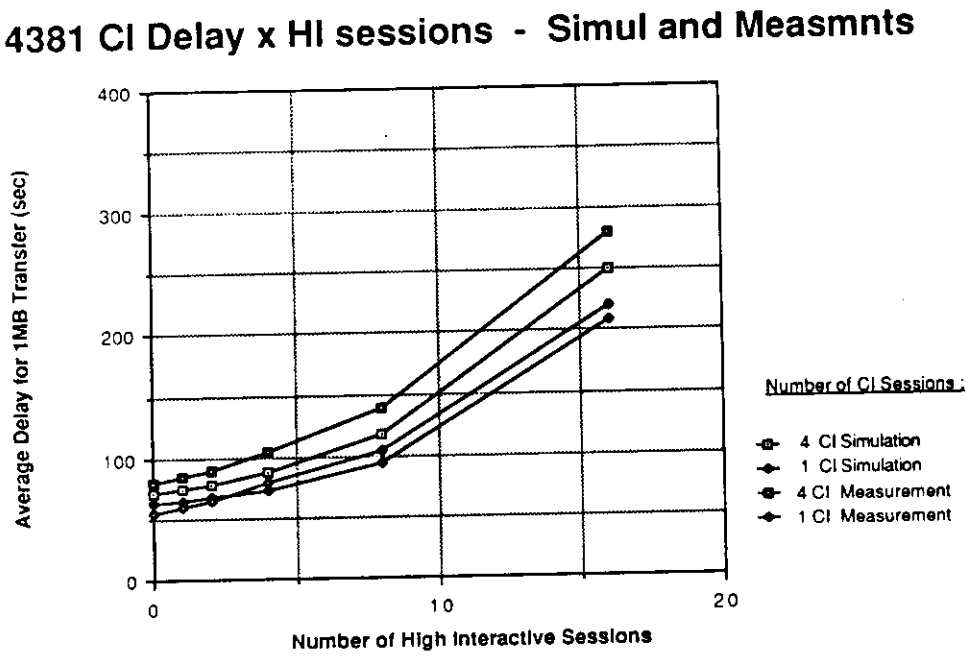


Figure 4: Comparison of Measurement and Simulation Results

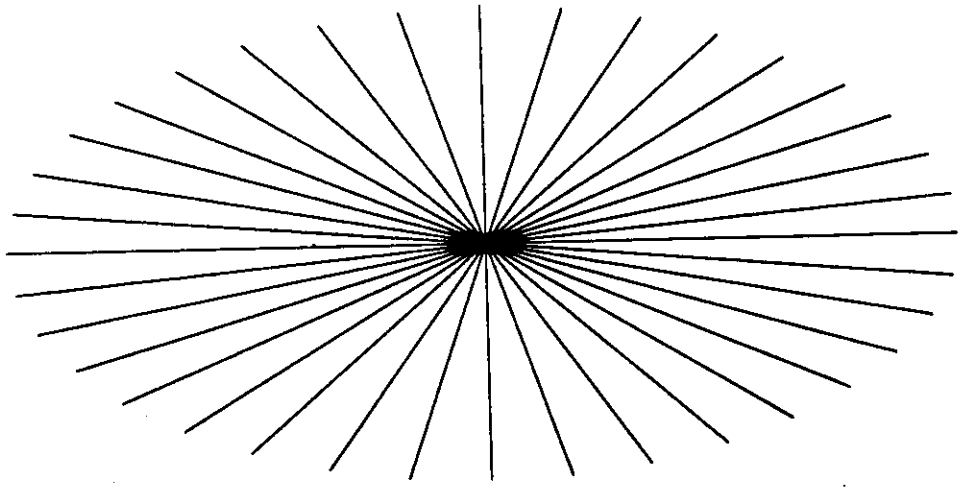


Figure 5: Corresponding Star Topology for Backbone-Server Cluster

1. Number and type of backbone servers to allocate (cost constraint bounded)

2. Specific session allocation onto the backbone servers

The problem can be transformed into a discrete Capacity and Flow Assignment problem (CFA) by mapping each backbone server onto a link, and mapping the number of sessions allocated to a backbone onto the flow of the corresponding link. This entails a star topology, as illustrated in Figure 5. This entails a star topology, as illustrated in Figure 5.

We will solve the original problem by solving the CFA between a source and a destination connected by a number of links equal to the maximum number of sites that any feasible solution may include and having as the total flow the entire user workload. When the CFA algorithm converges, the number of backbone servers in the system will be equal to the number of links with nonzero capacity, and the flow on each link will represent the session set assigned to the backbone server.

We must search through many local minima until we find a cost effective solution that satisfies the problem constraints. That will be done by choosing different starting feasible flows for each topology considered.

We will execute the capacity assignment algorithm to find the initial session allocation to each backbone server that satisfies the cost constraints. We will then iterate between the Flow Deviation (FD) and the Capacity Assignment (CA) until we find a local minimum. The workload that will result from this process will be suboptimal due to the fact that the objective function is concave.

In the following sections we will consider the algorithms in detail, and apply them to concrete examples to evaluate their effectiveness.

4 The Capacity and Flow Assignment Algorithm

4.1 Capacity Assignment Algorithm

Let's first introduce some notation:

$backbone(i)$	- backbone server (site) allocated to link i.
$cost(i)$	- cost of backbone(i).
$upgrade(i)$	- backbone server allocated to link i after upgrade.
T	- average system response time.
T_{max}	- constraint on average system response time.
$\Delta T(i)$	- difference in the delay of link i if one more session is transferred to it.
r	- Incremental delay per upgrade cost.

The Capacity Assignment algorithm is as follows:

1. Choose the minimum fit capacity assignment that matches the flows in each link. For every backbone server that has at least one session assigned to it, the minimum fit assignment will be the least expensive backbone server. If no session is assigned to a backbone server, that site will be deleted.

2. If the total cost assigned to the links is greater than the maximum cost, then STOP. The problem is not feasible.

3. Calculate the total average response time of the system by table look up.

4. If T $< T_{max}$ STOP. The problem is feasible, and the allocated capacity is suboptimal.

5. Compute the ratio $r = -\frac{\Delta T(i)}{cost(upgrade(i)) - cost(backbone(i))}$ for all links. r is the incremental delay per upgrade cost. If the backbone server assigned to a site cannot be improved (it is already the most expensive backbone server available), then set r=0.

6. Find the link that has the largest r (r is always positive). If $r = 0$, then STOP (all sites are already using the most expensive backbone server). Otherwise upgrade the backbone server in that link and GOTO step 2.

The algorithm will find a cost effective solution, since at every step it upgrades the capacity that gives the greatest response time reduction per Dollar. The method will typically generate suboptimal solutions. An optimal solution could be obtained (at higher cost) using a dynamic programming approach [Frank.etal69].

4.2 The Sub Optimal Session Assignment Algorithm

The sub-optimal session assignment algorithm is as follows:

1. Compute T_a the average response time at the initial flow assignment.

2. For each link compute the incremental delay as a function of a unit increment in the flow (transfer of δ_c sessions).

3. Find the link that has shortest incremental delay.

4. Find the link (having nonzero flow) that has the maximum incremental delay (the zero flow links are not taken into consideration).

5. Deviate a unit flow(δ_c) from the maximum incremental delay class of the maximum incremental delay link to the minimum incremental delay link.

6. Compute T_c the average response time at the current flow assignment.

7. If $T_a - T_c < \epsilon$ or $T_c > T_a$, then STOP. Otherwise do $T_a = T_c$, and GOTO step 2.

The algorithm computes the incremental delay for each link by computing the numerical partial derivative with respect to each class and deviates δ_c sessions of the maximum incremental delay class from the link with maximum incremental delay, to the one with minimum incremental delay. Each class has a constant δ_c that is calculated according to the load that a session from that class brings to the system. In our computational examples the classes High Interactive and Communication Intensive have $\delta_c = 1$ and the class Interactive has $\delta_c = 4$.

In the next section we apply the algorithms to three different topologies under various costs and compute the cost performance curves for uniform, skewed, and suboptimal workloads. The suboptimal workload is the workload that results upon convergence of the Capacity and Flow Assignment algorithm.

5 Configuration Synthesis Examples

In this section we present three configuration synthesis examples. The workload consists of three classes, namely Interactive (Int), High Interactive (HI), and Communication Intensive (CI). There is a total of 90 sessions divided among the classes (48Int, 24HI, 18CI). We consider costs in the range of 1.2 to 7.0 million Dollars and topologies with 4,6, and 10 backbone servers. The backbone servers can be a 4361 or a 4381 with typical costs of 300k and 700k Dollars[3]. The assignment of sessions to the backbone servers can be *skewed, uniform, or suboptimal.* A skewed assignment is one where some backbone servers are overloaded and others are underloaded. With this workload we represent the situation that occurs when users are permitted to connect to any backbone server and the system is not balancing the load. A uniform assignment results when each backbone server receives an equal number of sessions from each class. The subopti-

[3]These costs are illustrative. They can vary substantially depending on system peripherals, customer discount, etc.

	Uniform			*Skewed*		
Site	*Int*	*HI*	*CI*	*Int*	*HI*	*CI*
1	12	6	4	16	0	4
2	12	6	4	16	0	4
3	12	6	5	8	12	5
4	12	6	5	8	12	5

Table 1: Uniform and Skewed Workloads with 4 Backbone Servers

mal assignment is the result of the optimization described in the previous sections.

5.1 Synthesis with 4 Backbone Servers

In this example we divide the 90 sessions among the classes as follows: 48 interactive, 24 high interactive and 18 communication intensive. The uniform and skewed workloads for 4 sites are shown in Table 1.

We run the optimization for costs ranging from 1.2 million Dollars to 2.8 million Dollars. This represents the entire cost allocation space. In Table 2 we show the backbone servers allocated to each site, the corresponding cost, and the suboptimal workload computed.

In Figure 6 we plot the corresponding average delay for each configuration. We can see that the upgrade of the backbone servers is producing significant improvement in the performance. This occurs because the load applied to system is overwhelming the 4361s.

5.2 Cost Performance with 4,6, and 10 Backbone Servers

In Table 3 we report the optimization results for 6 backbone servers in the range from 2.2 M Dollars to 3.0 M Dollars. Suboptimal session allocations are indicated for each backbone configuration set.

Figures 7 and 8 show the condensed cost performance curves for the optimization with 4, 6, and 10 sites. Solutions are defined for a spectrum of sub-optimal cost and performance. Once the constraints are given, a solution becomes readily available.

Site	*Backbone*	*Sessions in each class*		
		Int	*HI*	*CI*
cost of 1.2 M Dollars				
1	4361	16	0	5
2	4361	16	0	4
3	4361	8	12	7
4	4361	8	12	8
cost of 1.6 M Dollars				
1	4361	12	6	4
2	4381	16	5	7
3	4361	12	7	3
4	4361	8	6	4
cost of 2.0 M Dollars				
1	4361	12	6	3
2	4381	12	5	7
3	4381	16	7	5
4	4361	8	6	3
cost of 2.4 M Dollars				
1	4381	12	6	6
2	4361	12	6	2
3	4381	12	5	6
4	4381	12	7	4
cost of 2.8 M Dollars				
1	4381	12	4	6
2	4381	12	5	6
3	4381	12	9	2
4	4381	12	6	4

Table 2: Suboptimal Workload with 4 Backbone Servers

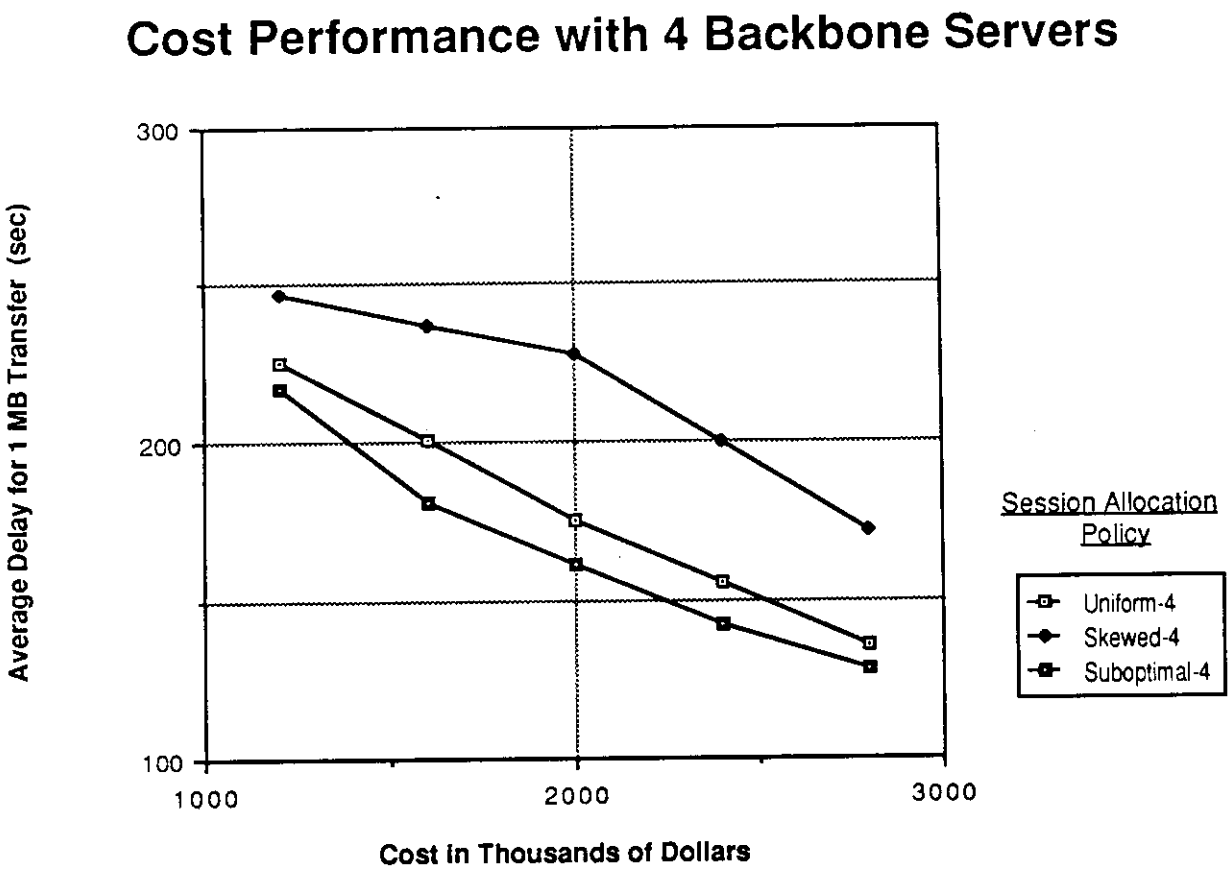


Figure 6: Cost Performance with 4 Backbone Servers

Site	*Backbone*	*Sessions in each class*		
		Int	*HI*	*CI*
cost of 2.2 M Dollars				
1	4381	16	5	3
2	4361	12	3	3
3	4361	12	2	3
4	4361	4	0	5
5	4361	4	7	2
6	4361	0	7	2
cost of 2.6 M Dollars				
1	4381	12	4	5
2	4381	12	4	5
3	4361	12	4	1
4	4361	4	0	4
5	4361	4	6	2
6	4361	4	6	1
cost of 3.0 M Dollars				
1	4381	12	3	6
2	4381	12	3	5
3	4361	12	4	1
4	4361	4	0	3
5	4381	4	8	2
6	4361	4	6	1

Table 3: Suboptimal Workload with 6 Backbone Servers

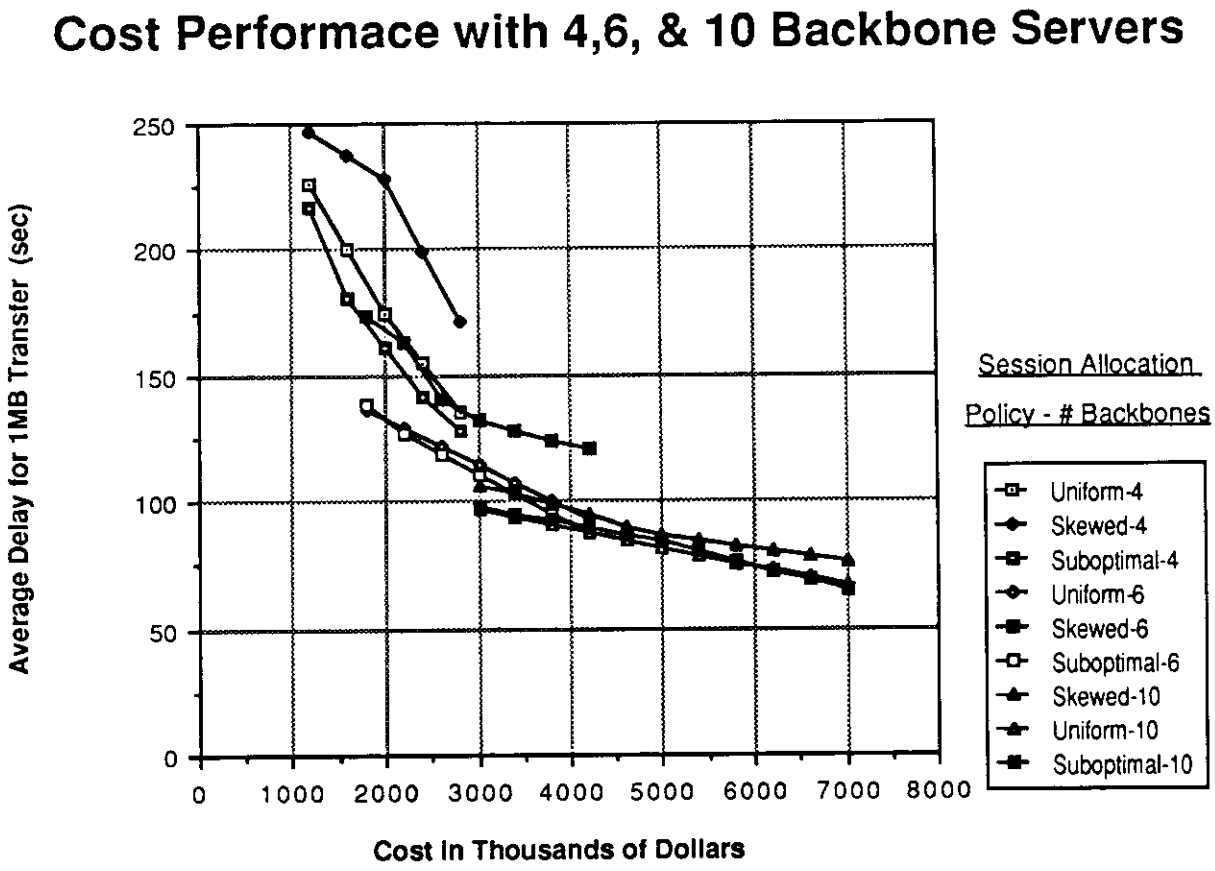


Figure 7: Cost Performance with 4,6 & 10 Backbone Servers

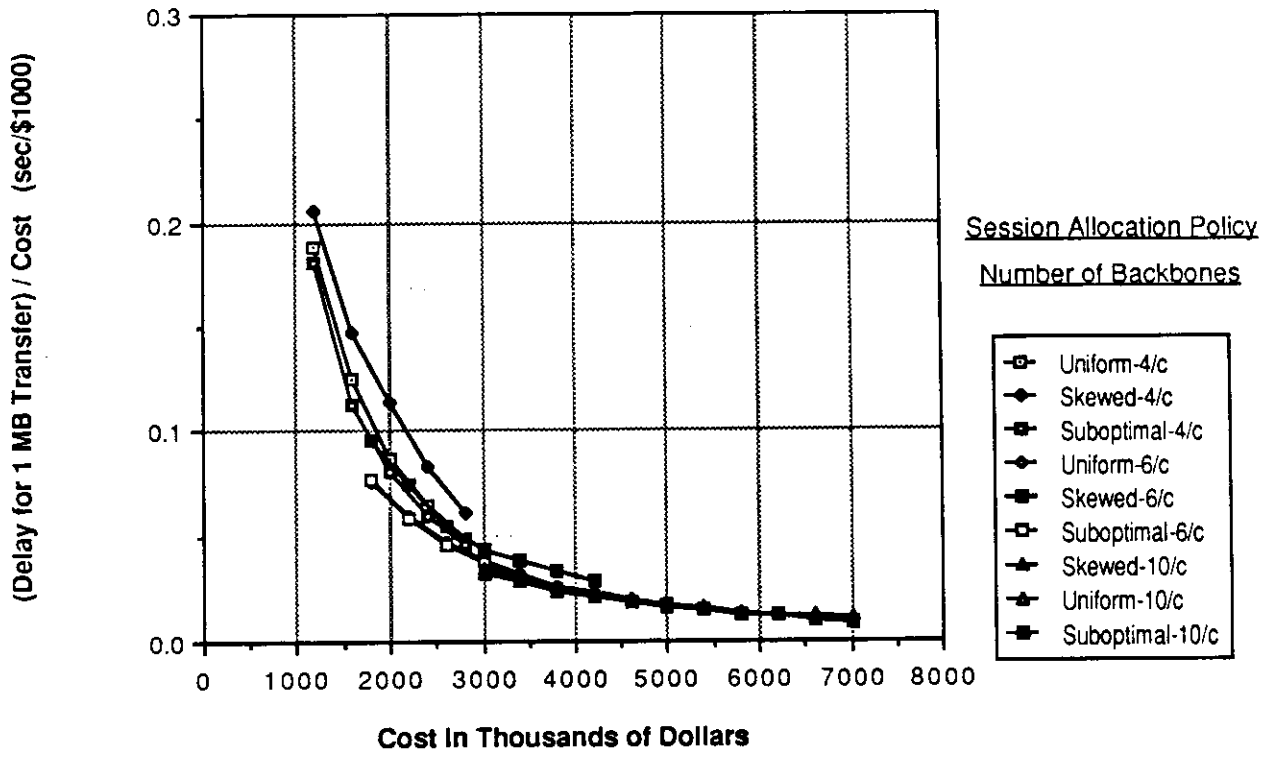


Figure 8: Cost vs (Performance/Cost) with 4,6 & 10 Backbone Servers

We can see that the best working range is located in the range of 2-3 M Dollars. This working range is defined by the knee of the curves. We note that the best workload configuration for 10 sites with 3.0 M Dollars is the uniform workload. This occurs because the 90 sessions workload distributed among 10 sites drives all backbone servers at very low utilizations. This makes performance less sensitive to load allocation in this capacity-overkill situation. Clearly, this is not a cost effective solution for the user load given in our example.

5.3 Computational Considerations

The synthesis of networks with discrete capacity is a time consuming task due to the concave shape of the objective function. In our case we have multiple user classes and backbone servers, as well as an unknown topology. An optimal solution using dynamic programming approach [Frank.etal69] would entail extremely expensive computations. Our approach iteratively finds a suboptimal solution that is as good as the computational budget available.

The designer must allocate his budget to the subtasks of user workload characterization, the generation of the system delay tables, and the configuration synthesis optimization. Each of the phases is time consuming and computationally costly due to the very large state space. The optimization is quite efficient but it requires a number of iterations with different initial feasible flows until a good workload distribution is found. The uniform distribution produces good initial results and should be the first to be compared with the given delay constraints.

6 Conclusions

In this work we considered a rather complex configuration synthesis problem. Both the offered load and the computation/communication resources carry a high degree of richness and complexity. This makes an intuitive or straightforward engineering solution very difficult.

Through an extension of the CFA algorithm, combined with application to backbone computing resources, we derive a direct way to construct a performance-cost effective backbone network. This is accomplished through

an optimization technique which deviates resource assignments in order to identify sub-optimal solutions to the problem.

We also note that heterogeneous resources are modeled by studying measured behavior, and by emphasizing the most dominant characteristics of the integrated system. The fact that we have measured and compared the physical system to predicted simulation figures gives us increased confidence in the presented results.

Unlike recent contributions to this area [TanTowWol88], we do not make restrictive assumptions with respect to service times for the user classes considered. We obtain stable numerical description of class behavior, and input these results into the optimization algorithm. This enhances the stability of the optimization algorithm, as possible instabilities in the simulation are handled in advance.

The convergence of the presented optimization algorithm is impressive, as solutions were obtained in less than 12 iterations for most cases, consuming about 1-2 minutes on 68020 based microprocessor. Stability of the algorithm still needs to be studied, as in some cases there is divergence from local minima.

The presented configuration synthesis algorithms are very general, as they apply to resource allocation in the most general form. It is important to create an appropriate performance mapping from raw resources and load characteristics to the system under consideration. Once that is done, the optimization can be readily applied, resulting in efficient resource utilization.

It should be recognized that there is possibility to extend this work to obtain an automatic search on the resulting plots such as in Figures 7 and 8. Hence cost/performance criteria could be defined to suggest the best working area on the curve, depending on resource designation and financial capacity committed to the planned resources.

7 Future Research

This work branches into several interesting directions. Using the fundamental synthesis approach and tools we constructed, more complex problems could be attempted.

1. *Direct Unix applications on an AIX/370 Cluster.* In our analysis and synthesis we addressed the predominant SEASnet paradigm of PCI service by the Locus TCF cluster. There are additional modes of operation that would constitute new classes of customers. One such class is the Unix application family, interacting directly with Locus or AIX/370 or any Unix family system [BeLaCaKa87]. The combination of PCI users, Unix users and other classes, such as work-stations with windowing interface will constitute a very interesting study.

2. *User behavior detail.* The classes of user load could be enriched to include more statistics about usage histograms, traffic increases, time clustering, and other aspects of behavior, pertaining to additional classes as mentioned above.

3. *Resource assessment in finer granularity.* In addition to CPU and I/O resources in the clusters, one could address the following finer grain issues :

   (a) *Windows.* Many new work-stations with graphics capabilities provide windowing systems such as X, SUN-TOOLS and others. The communication and I/O resource requirements of such architectures is of high interest and importance for future designs.

   (b) *Distributed file accesses.* The availability of data has two vastly varied flavors. Local data is accessible directly within a machine. Remote data requires all the communication/network-protocol overhead to become available. The trade-offs of replicating vs remote access or process migration constitute a rich array of possible extensions.

   (c) *Reliability.* Another vast number of design optimization relates to performance costs due to higher reliability. This price could be traded off with reliability design constraints to achieve the final resource allocation for the system.

4. *Load Balancing.* In this work we are configuring a system such that cost is optimized. Part of the algorithm deals with good utilization of the resources. It would be even more challenging to maintain a balanced load situation for a variety of user loads in the future.

(a) *Static allocation.* In this case we address the pre-allocation of jobs to a given configuration. A new job is assigned according to mean time statistics.

(b) *Dynamic real-time.* This is a fine grain assignment scheme based on real-time monitoring of all jobs in operation. This scheme would not monitor a job once assigned, as only new jobs are assigned [EagLazZah88].

A number of the above topics are currently under investigation at UCLA. It should be pointed out that many extensions beyond the context of SEASnet and UCLA are possible, due to the generality of the optimization algorithms presented.

8 Acknowledgement

We wish to thank Mario Gerla for some preliminary thoughts that ultimately led to this work.

References

[AvrGer88] A. Avritzer and M. Gerla "Load Balancing in a Distributed Transaction System", In preparation.

[BaChMuPa75] F. Baskett, K. Chandy, R. R. Muntz, and F. Palacios, " Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", JACM Col 22, pp 248-260, April 1975.

[Betser84] J. Betser, "Performance Modeling and Enhancement within the Locus Distributed System", M.S. Thesis, UCLA Computer Science Department, 1984.

[BetGerPop84] J. Betser, M. Gerla, and G. J. Popek, "A Dual priority MVA Model for a Large Distributed System : LOCUS", Performance '84, Proceedings of the 10th International Conference on Computer Performance (ed. E. Gelenbe), Paris France 19-21 December 1984, pp 51-66.

[Betser87] J. Betser, "Performance Evaluation and Prediction for Large Heterogeneous Distributed Systems", Ph.D. Dissertation Prospectus, UCLA Computer Science Department, Los Angeles, CA 90024, 1987.

[BeLaCaKa87] J. Betser, Nick Lai, Jack W. Carlyle, Walter J. Karplus, "LOCUS and SEASnet - Performance Analysis", Technical Report, UCLA Computer Science, 1987.

[Betser88] J. Betser, "Performance Evaluation and Prediction for Large Heterogeneous Distributed Systems", Ph.D. Dissertation, UCLA Computer Science Department, Los Angeles, CA 90024, 1988.

[CheCarKar86] Shun X. Cheung, Jack W. Carlyle, and Walter J. Karplus, "Asynchronous Distributed Simulation of a Communication Network", in Proceedings of the Summer Computer Simulation Conference, Society for Computer Simulation, July, 1986.

[DeSMunLav84] E. de Souza e Silva, R. R. Muntz, and S. S. Lavenberg, "A Clustering Approximation Technique for Queueing Network Models with a Large Number of Chains", IEEE Transactions on Computers, Vol C-35, No 5, May 1986.

[DeSoGerl84] E. de Souza e Silva and M. Gerla, "Load Balancing in Distributed Systems with Multiple Classes and Site Constraints", Proceedings Performance '84, E. Gelenbe (ed), Paris 1984, North Holland, pp 17-33.

[EagLazZah88] D. Eager, E. Lazowska, and J. Zahorjan, "The Limited Performance Benefits of Migrating Active Processes for Load Sharing", Sigmetrics 88, May 1988.

[Fox66] B. Fox, "Discrete Optimization Via Marginal Analysis", Management Science, Nov 1966.

[Frank.etal69] H. Frank et al, "Design of Economical Offshore Natural Gas Pipeline Networks", Office of Emergency Preparedness, Report R-1, Washington DC, Jan 1969.

[FraGerKle73] L. Fratta, M. Gerla and L. Kleinrock, "The Flow Deviation Method - An Approach to Store-and-Forward Communication Network Design", Networks 3, 1973.

[HeidLaks87] P. Heidelberger, and M. S. Lakshmi, "A Performance Comparison of Multi-Micro and Mainframe Database Architectures", to appear in IEEE Tran. on Software Engineering, Presented in 1987 ACM Sigmetrics, Banff, Alberta, Canada, May 11-14, 1987 (Also IBM tech report RC 12230, T. J. Watson Research Center, Yorktown Heights, New York, 10598, February 1987).

[Kleinrock76] L. Kleinrock, "Queueing Systems, Volume II: Computer Applications", Wiley-Interscience, New York, 1976.

[Lavenber83] S. S. Lavenberg (editor), "Computer Performance Modeling Handbook", Academic Press, 1983.

[LaZaGrSe84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik, "Quantitative System Performance - Computer System Analysis Using Queueing Networks Models", Prentice Hall, Englewood Cliffs, New Jersey 07632, 1984.

[RaghSilv86] C. Raghavendra and J. Silvester, "A Survey of Multi-Connected Loop Topologies for Local Computer Networks", Journal of Computer Networks - ISDN Sys., June 1986, 29-42.

[Stenstro87] Michael K. Stenstrom, "SEASnet - A Network for Educational Computing", Proceedings 1987 ASEE (American Society for Engineering Education) Conference, Reno, Nevada, pp. 1540-1543, 1987.

[TantTows85] A. N. Tantawi and D. Towsley, "Optimal Static Load Balancing in Distributed Computer Systems, JACM, Vol 32, No 2 April 1985, pp 445-465.

[TanTowWol88] A. N. Tantawi, D. Towsley and J. Wolf, "Optimal Allocation of Multiple Class Resources in Computer Systems", Sigmetrics, May 1988, pp 253-260.

[WoodTrip86] C. M Woodside and S. K. Tripati, "Optimal Allocation of file Servers in a Local Network Environment", IEEE Trans. on Software Engineering, Vol 12, No 8, August 1986, pp 844-848.