Decentralized Work-In-Process Optimization in Cooperative Resource Allocation

Doraid Dalalah
Industrial Engineering Department, Faculty of Engineering, Jordan University of Science and Technology
PO Box 3030, Irbid, 22110, Jordan
E-mail: doraid@just.edu.jo

***Abstract*—Resource allocation entails deciding how to split a resource of restricted availability among various demands in a way that optimizes current objectives. In this paper, we focus on one type of distributed resource allocation problems in which a distributed system comprising networked heterogeneous agents and processors/servers where the agents strive to boost their efficiencies by issuing more work transactions for higher throughput. Each agent can issue work transactions which comprise a set of tasks that have to be completed by a networked set of servers. The agents get more utilities as their transactions delivery rate increases, however, network administration entails a set of constraints on the tolerable delay on the transactions. An optimization model is constructed to characterize the system in which the available servers are allocated to the present agents. The optimization model is solved in a decentralized way so that the agents can work separately to maximize the global benefit measure of the network. The presented model numerical solution is put together using ARENA simulation package and experimented for different network topologies. The simulation results show fair allocation of the resources whereas the anonymous agents work in parallel to achieve optimality.**

Keywords: Resource allocation; Parallel computing; Primal dual optimization; Cooperative control.

I. INTRODUCTION

Algorithms for resource allocation date back to the 1960s [1, 2]. Classical resource assignment problems usually deal with the optimal allocation of tasks to agents in such a way that each member is given one task to be completed in a busy environment, [3]. Such problem has been dealt with under the consideration of constraints on the resource consumption. Literature has dealt also with more generalized assignment problems where multiple tasks are assigned too. It was also seen that several location problems can be represented and solved as generalized assignment problem. In the multi resource generalized problems the agents consume multiple resource types in performing their task. Although a single resource type can be successful in modeling some of the allocation problems, however, recent advances in business and business administration with fully integrated and automated systems make the problem further intricate, [4]

Resource allocation problems can be found in different applications going back to the early research in transportation and truck routing [5], to the recent nowadays communication networks. In fact, the research in such field has achieved quite interesting and pioneering advances particularly in communication networks taking the advantage of fast system response as compared to allocating resources in other areas such as transportation, machines, human and social networks [4, 6, 7, 8, 9].

The recent advances in internet and intranet communication brought such motivation for the need of efficient and fair algorithms that can cope with the massive number of network users. Indeed, numerous approaches were used in internet traffic control and resource allocation such as the optimization techniques as in [10], the economic optimization approaches [11, 12, 13, 14, 15,16], control theoretic based solutions [17, 18] and per-flow queuing [19].

Scarce attention has been paid to the other types of networks such those which deal with people, machines, work and multiple task allocation as compared to communication networks. As a matter of fact, the flow of work inside a business organization or the flow of work among people, processes, and tasks directly impacts the productivity and quality of the organization outcomes. An effective workflow can dramatically improve projects throughput and the return on investment for any production facility. One of the fundamental tasks in business administration consists of the performance or management of business operations by organizing people and resources efficiently so as to direct activities toward common goals and objectives, [20].

Administrators, broadly speaking, engage in a common set of functions to meet the organization's goals. These "functions" include planning which maps the path from where the organization is to where it wants to be. Other functions include organizing, staffing, commanding, and controlling which is a function that evaluates quality in all areas and detects potential or actual deviations from the organization's plan. Accordingly, resources allocation seems to be an important factor for perfect administration of an organization, [21].

This paper introduces a novel framework in a particular business environment for the analysis and design of business network which simply consists of distributed agents and processors (servers) who/which work together to complete transactional type of work. Here, the distributed agents can issue transactions which have to be processed by a set of servers.

A distributed asynchronous policy is presented that dynamically adjusts the transactions flow across the network

COLLABORATECOM 2011, October 15-18, Orlando, United States

DOI 10.4108/collaboratecom.2011.26

of agents and servers. It is shown that even though agents independently adjust their transactions issuance rate, the set of all agents' tendencies converges to the unique equilibrium of a cooperation game. The presented framework will stabilize the size of work-in-process and provides a better control on the work flow between the servers.

The presented model entails that the agents are anonymous to each other and have to work in parallel to maximize the benefits of the whole network. Here, a global objective function is proposed subject to a certain level of work-in-process which has to be set by the network administration. The model is solved in a decentralized way, where the agents can stick to some transactions delivery rates by which the network can reach a steady state operation while allocating the necessary proportions of the servers' capacities to the different agents.

The presented model is tested for different network configurations using ARENA simulation package. The simulation results show that the model can drive the network to steady work-in-process levels. The servers were seen to have limited queues and the agents -once committed to the assigned delivery rates- will drive the network to optimality in a decentralized manner. Further, the network throughput is calculated for the given experiments, however, our model does not target the throughput, instead, it promises constant work-in-process and steady state operation.

The rest of the paper is organized as follows: next to this introduction, the optimization model is presented followed by the constraint relaxation in section III. The agents adaptation is illustrated in section IV followed by the experimental part in section V. The rates and queues proportionality are presented in section VI and finally the conclusions in section VII.

II. THE MODEL

Here, we assume a set of agents $\boldsymbol{A}=\{1, 2, \ldots, n\}$, the agents can initiate their own transactions/work orders that have to be processed by a set of servers $\boldsymbol{S}=\{1, 2, \ldots, m\}$. Essentially, in order to complete a transaction, it has to go through a predetermined sequence of tasks. Each server is held responsible for performing one single task, however, the same task may be required by different transactions where in such scenario the server will have to perform the similar task for transactions belonging to different agents. The processing capacities of the servers is given by c_s (transactions/time unit) $\forall s \in \boldsymbol{S}$.

The transactions issued by agent 'a', ($a \in \boldsymbol{A}$) will be processed by a subset of servers $\boldsymbol{S}(a) \subseteq \boldsymbol{S}$. Similarly, each server 's' will have a subset of agents $\boldsymbol{A}(s)$ whose transactions are processed by 's'. That is, $\boldsymbol{A}(s) = \{a \in \boldsymbol{A} \mid s \in \boldsymbol{S}(a)\}$. The agents are competing against each other so as to process as many transactions as possible, however, the limited servers' processing capacities as well as the tolerable delays due to transactions waiting at the buffer of the servers make such competition unfair, particularly, if not governed by a control mechanism that guarantee some fairness scheme among the agents. Fig. 1 depicts the proposed concept.

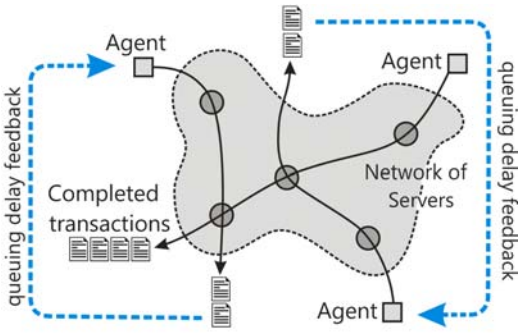

Fig. 1: Agent-Server network concept.

We present an optimization model composed of a general objective function which measures the social benefit of all the agents. It is assumed that an agent gets a utility according to the rate of transactions delivery where the more transactions issued by agent 'a' the more returns/benefits are expected. Let (x_a) denote the rate of transactions delivery of agent 'a' and $U(x_a)$ denotes the utility the agent attains at such transaction rate. The function $U(x_a)$ has to be strictly increasing and differentiable over some period [0, M_a]. In this paper, we suggest the following function which is strictly increasing over the given period above:

$$U(x_a) = \ln(x_a) - x_a / M_a \qquad (1)$$

From the network stand point, maximizing the aggregation of such objective function will maximize the overall agents' utility. However, the agents cannot just issue transactions and overwhelm the servers by piles of transactions. As a part of our model, the set of servers can take a certain limit of work-in-process; such work includes the transactions being serviced and those waiting in queues.

For every agent, denote the network allowable limit of transactions-in-queues by κ_a, meaning that, the number of transactions waiting in lines of agent 'a' should not exceed the limit κ_a. Accordingly, the total transactions processing time T_a spent in the whole system consists of two components, the total transaction service times and the total queuing delay (τ_a) at the servers of 'a', alternatively, this can be put across as $T_a = \sum_{s \in S(a)} 1/c_s + \tau_a$. Consequently, the average total number of transactions in service is given by $x_a . \sum_{s \in S(a)} 1/c_s$ while the number of transactions waiting in queues for the same agent is $x_a . \tau_a$. Under such notation, the optimal network transactions flow rate, will solve the following optimization problem:

$$\begin{aligned} &\max \sum_{a\in A} \ln(x_a) - x_a / M_a \\ &st. \\ &x_a \tau_a \le \kappa_a \quad \forall a \in A \\ &x_1,...,x_n \ge 0 \end{aligned} \tag{2}$$

Note that the above model imposes an upper bound on the number of transactions currently being in queues. The model is non-linear in its objective function as well as the constraints. In fact the LHS of the first constraint is a function of the transactions rates vector $\boldsymbol{x} = \{x_1,...,x_n\}$. However, to simplify our solution analysis, few sound assumptions will be made accordingly as will be illustrated shortly.

III. CONSTRAINT RELAXATION

In our model, the agents have to work in parallel to maximize the overall utility. But the dilemma is that agents are anonymous to each other, in fact they don't have the access to any global network parameters except the feedback about their transactions delays and minimum processing times. As matter of fact, each transaction has a predetermined sequence of tasks and hence, an agent is allowed to know its own server processing times as well as the encountered delay. The minimum processing time in the case of uncongested network is simply the sum of the processing times of all the tasks of each transaction.

The above model has to be solved in a decentralized manner where no coordination between the agents will take place. To ease the analysis, we assume that the servers receive heavy traffic arriving from the different agents to the extent that the queuing delay is not affected by only one single agent. This assumption has been recognized in queuing networks [16, 22], which can be represented as:

$$\frac{\partial \tau_a}{\partial x_a} \approx 0 \tag{3}$$

Now that such assumption has been made, we may relax the constraint using the dual of problem 2. The dual of the primal problem is given by:

$$g(x,\eta) = \sum_{a\in A} \ln(x_a) - x_a / M_a + \eta_a(\kappa_a - x_a \tau_a) \tag{4}$$

where η is a vector of Lagrange multipliers. The above relaxation shows that if the number of transactions in queues is equal to the limit κ_a, then the objective function is strictly increasing and there exists a rate vector that may further maximize the objective function, however, increasing the rates little more will result in queues that will build up at the servers. Hence, the above objective function will have a unique maximum at some optimal vector of Lagrange multipliers $\boldsymbol{\eta}^* = \{\eta_1^*,...,\eta_n^*\}$. The gradients of the above objective function is given by:

$$\frac{\partial g(x,\eta)}{\partial x_a} = \frac{1}{x_a} - 1/M_a - \eta_a \tau_a$$

$$\frac{\partial g(x,\eta)}{\partial \eta_a} = \kappa_a - x_a \tau_a$$

Accordingly, the optimal solution happens when the above two expressions equal to zero, meaning that the problem of maximizing a strictly increasing objective function cannot be bounded unless enclosed by an upper bound constraint, whereas, the optimal rates cannot be calculated only when the optimal vector $\boldsymbol{\eta}$ is known. Now, the gradient projection algorithm in [23] can help in such situation. Equating the expression $\partial g(x,\eta)/\partial x_a$ to zero results in:

$$x_a = \frac{1}{\eta_a \tau_a + 1/M_a} \tag{5}$$

while the Lagrange multiplier can be found as follows:

$$\eta_a^{new} = [\eta_a^{old} - \delta(\kappa_a - x_a \tau_a)]^+ \tag{6}$$

where δ is small scalar and $[.]^+$ refers to the maximum of 0 and the expression inside the brackets.

IV. AGENTS ADAPTATION

In order to maximize the objective function in a decentralized manner while not exceeding the upper bound on the queues belonging to each agent, the agents have to be able to observe the delay on their transactions through some feedback system which is an available service in recent automated queuing systems (the feedback in Fig. 1). When the quantity τ_a is available at the agent side, the agent can easily compute the transactions delivery rate which will drive the network to optimality or at least optimality neighborhood. To do so, the agents have to stick to following rules.

- Starting from a zero value, the agent multiplier η_a has to be calculated as follows:

$$\eta_a^{new} = [\eta_a^{old} - \delta(\kappa_a - x_a \tau_a)]^+$$

- Further, the agents should issue transactions at a rate calculated by the following expression:

$$x_a = \frac{1}{\eta_a \tau_a + 1/M_a}$$

Note that when M_a increases, the rate expression will reduce to $x_a = 1/\eta_a \tau_a$ while the objective function will approach the logarithmic shape. The higher the value of M_a, the more guarantees that the constraint of the primal problem will remain active. Accordingly, to avoid under estimation of the LHS of the first constraint, M_a of each agent has to be as large as possible. Once the agents update their rates according to the above mentioned rules, steady state rates will be achieved while keeping constant work in queues without having to overwhelm the network of servers. Note that the parameters δ can only affect the stability at real-time implementation as illustrated in the experimental examples in the next section.

V. SIMULATION EXPERIMENTS

In this example, three agents issue transactions according to the routing sequence shown in Fig. 2. Two servers are available for processing the assigned tasks. The processing capacity of the two servers is set to 50 transactions/hr and the transactions-in-queues upper bound is set to 10. The agents are to deliver their transactions according to the calculated rates in section IV.

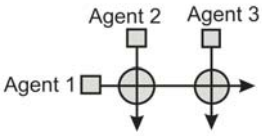

Fig. 2: A network consisting of 3 agents and two servers.

The real-time simulation gives the transactions delivery rate of the three agents as shown in Fig. 3. Here, at the steady state, agents 2 and 3 will deliver at a rate of 33.333 transactions/hr, while the first delivers at a rate of 16.667 transactions/hr. At the equilibrium, the total of the inward rates of a bottleneck server is equal to the server capacity. Further, 15 transactions were queued at the buffer of server 1 and the same at server 2. Each agent will have 10 transactions queued in system where the first agent will have 5 of its transactions queued at server 1 and the remaining at server 2 as shown in Table 1. The optimal Lagrange multipliers are also shown in Fig. 4. Note that both servers have 15 transactions queued at the buffer, hence for all agents we have $x_i \tau_i = 10$. Now when $x_1 = 16.667$, $x_2 = 33.333$ and $x_3 = 33.333$, the multipliers can be easily found using (5) to conclude that $\eta_1 = 0.07222$, $\eta_2 = 0.0444$ and $\eta_3 = 0.0444$ where $M_a = 60 \ \forall a \in \mathbf{A}$ in this example.

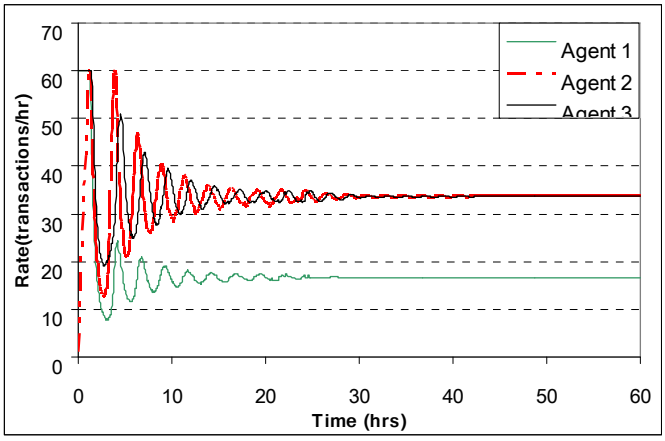

Fig. 3: Agents transactions delivery rates.

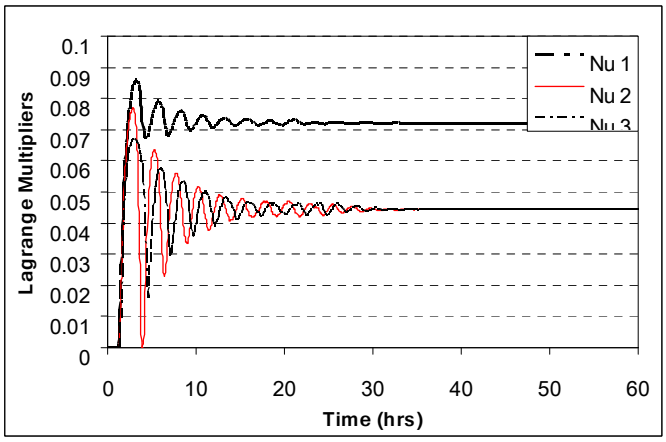

Fig. 4: Agents multipliers.

Agent (a)	Rate (trans/unit time)	Entities-in-queues q_1^a	Entities-in-queues q_2^a	Total entities-in-Queues per route $= \sum_{s \in S(a)} q_s^a$	κ_a
1	16.667	5	5	10	10
2	33.333	10	-	10	10
3	33.333	-	10	10	10
$\sum_{a \in A(s)} q_s^a = q_s =$		15	15	30	30

Table 1: Steady state simulated queues and rates, where q_s^a refers to the number of tranasactions of agent 'a' at server 's' and q_s is the queue size at 's'. Note that the "-" means the agent does not utilize the related server.

The network throughput (neglecting the unsteady start phase, i.e., warm up period) is calculated as follows: The rate of the first agent is 16.667, the second and the third agents are 33.333 which add up to 83.333 transactions/hr. The same example has been solved using LINGO where the same objective function was used while subjecting the rates to the processing capacities constraints. The delivery rates were found to be the same as those found in the numerical simulation. Noticeably, since agent 1 consumes more resources of the network, it has been assigned the lowest rate.

Let us consider another experiment in which a single server with a processing capacity of 50 transactions/hr is in charge of processing the delivered transactions of three agents, where the first agent starts delivering at time 0, the second agent starts after 10 hrs and the third after 20 hrs. Likewise the agents terminate their sessions one after another in the same sequence. The simulation results are shown in Fig. 5. Note that the agents will start delivering at a rate equal to M_a which is set to 60 in this example, later, the rates will converge to the optimal value according to the existing number of agents. Here, since one server is available, even shares will be allocated to the agents. Clearly, the algorithm can track the optimal solution in a distributed way regardless the number of admitted/ terminated sessions. The observed oscillations in the rates are due to the search for optimal allocation. At any network condition, once close to the equilibrium, each agent

will have 10 transactions queued at the server; hence the server will have a total of 30 transactions queues at its buffer.

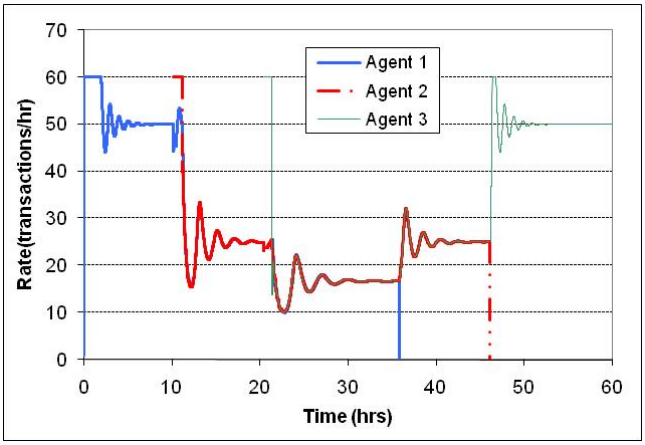

Fig. 5: Real-time simulated rates.

Next, we consider a network consisting of 15 different agents which have the access to a network of 12 servers. The transactions issued by the agents are to be processed by the servers according to the routing matrix in Table 2. For instance, to close the transactions issued by agent 1, two successive tasks have to be completed at the servers 2 and 6. The upper limit of transactions-in-queues for all the agents is set by the network administrator to 5 transactions per agent. This means no one agent will have more than 5 transactions of its own waiting in queues at the buffer of the servers. The processing capacities of the servers are given in Table 3.

	The servers set **S**										
Agents set **A**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1		1				1						
2					1						1	
3		1		1				1				
4									1			1
5		1							1			1
6				1			1				1	
7						1						
8	1					1						
9			1							1		
10	1				1							1
11											1	
12					1					1		
13	1								1			
14								1			1	
15			1			1				1		

Table 2: Agent-server routing matrix.

Servers	Processing capacity (trans/hr)
1	10
2	15
3	15
4	8
5	20
6	12
7	50
8	25
9	10
10	35
11	18
12	22

Table 3: Servers' processing capacities.

After conducting the numerical simulation, the steady state transaction delivery rates were found as shown in Table 4. In Table 5 we show the flows belonging to each agent at each server. Clearly, the servers 1, 3, 4, 5, 6, 9 and 11 are the steady state bottlenecks as their inward flows are equal to their processing capacities.

Agent	Transaction delivery rates
1	3.427
2	3.798
3	5.258
4	3.699
5	3.699
6	2.742
7	3.427
8	2.465
9	12.319
10	4.933
11	5.730
12	11.268
13	2.602
14	5.730
15	2.681

Table 4: Optimal rates.

The steady state number of transactions queued at each server as well as the queue proportions belonging to the different agents are shown in Table 6. The sum of queue components at the different tasks of agent '*a*' is given in the LHS column, i.e., the sum of rows, where for all agents, the total transactions-in-queue is equal to the upper bound κ. Here, we conclude again that the total number of different transactions waiting in queues inside the network is always equal to $\sum_{a\in A}\kappa_a$. At such delivery rates, the steady state network throughput of the given example is equivalent to 157.713 transactions/hr.

VI. AGENT DELIVERY RATES AND SEVER QUEUES PROPORTIONALITY

Suppose at the steady state that the proportion of a queue at server '*s*' that belongs to agent '*a*' is given by the ratio x_a / c_s, accordingly, the number of transactions of agent '*a*' that are waiting in the queue at the server '*s*' is given by $q_a^s = q_s x_a / c_s$, where q_s is the queue size at '*s*'. Consequently, the ratio of the delivery rates of any two agents, say '*o*' and '*p*' that share one bottleneck '*s*' is given by $x_o / x_p = q_s^o / q_s^p$. Such conclusion is apparent in our all simulation results. Now consider any two agents who are sharing one bottleneck server, say for example agent 8 and 10 at the server 1, here, we have $x_8 / x_{10} = q_1^8 / q_1^{10}$. Plugging the values from Table 4 and 6 results in 2.465/4.933=1.404/2.811. Similarly, take agent 8 and 15 who share the server 6, that is $x_8 / x_{15} = q_6^8 / q_6^{15}$. Plugging the values results in 2.465/2.681=3.596/3.912, (watch out, the rates and queues have been rounded to 3 decimal digits). Different experimental simulation examples have been test to verify the agent delivery rates and Sever queues proportionality. Interestingly, the results showed that such proportionality will hold for such objective function.

	The servers											
agent	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
1		3.427				3.427						
2					3.798						3.798	
3		5.258		5.258				5.258				
4									3.699			3.699
5		3.699							3.699			3.699
6				2.742			2.742				2.742	
7						3.427						
8	2.465					2.465						
9			12.319							12.319		
10	4.933				4.933							4.933
11											5.73	
12					11.268					11.268		
13	2.602								2.602			
14								5.73			5.73	
15			2.681			2.681				2.681		
Total arriving rate/server	10.000	12.384	15.000	8.000	20.000	12.000	2.742	10.988	10.000	26.268	18.000	12.331
Server processing capacity	10	15	15	8	20	12	50	25	10	35	18	22

Table 5: The agents' optimal rates across each server.

	The servers													
Agent	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	LHS	κ
1	-	0	-	-	-	5	-	-	-	-	-	-	5	5
2	-	-	-	-	1.685	-	-	-	-	-	3.315	-	5	5
3	-	0	-	5	-	-	-	0	-	-	-	-	5	5
4	-	-	-	-	-	-	-	-	5	-	-	0	5	5
5	-	0	-	-	-	-	-	-	5	-	-	0	5	5
6	-	-	-	2.607	-	-	0	-	-	-	2.393	-	5	5
7	-	-	-	-	-	5	-	-	-	-	-	-	5	5
8	1.404	-	-	-	-	3.596	-	-	-	-	-	-	5	5
9	-	-	5	-	-	-	-	-	-	0	-	-	5	5
10	2.811	-	-	-	2.189	-	-	-	-	-	-	0	5	5
11	-	-	-	-	-	-	-	-	-	-	5	-	5	5
12	-	-	-	-	5	-	-	-	-	0	-	-	5	5
13	1.483	-	-	-	-	-	-	-	3.517	-	-	-	5	5
14	-	-	-	-	-	-	-	0	-	-	5	-	5	5
15	-	-	1.088	-	-	3.912	-	-	-	0	-	-	5	5
Trans. queues	5.698	0	6.088	7.607	8.874	17.508	0	0	13.517	0	15.708	0	75	75

Table 6: Queue components at each bottleneck.

VII. CONCLUSIONS

We have described an optimization approach to resource allocation in cooperative network and derived a simple asynchronous distributed algorithm to solve for the optimal allocation. The model considers a set of agents which have the access to a set of servers. Transactions are issued by the agents and have to be completed by a predetermined set of servers in the network. The agents are anonymous to each other and have no access to the global parameters of the network. The presented numerical solution once implemented by the agent could track the optimum allocation when network conditions vary slowly. The allocation scheme has desirable fairness properties where agents consuming more resources are assigned less delivery rates. Further, steady state queues/work-in-process is maintained at the equilibrium operation. The algorithmic solution has been implemented in ARENA simulation package and has been test for different network topologies. The results conform to the stated assumptions and the scheme provides reasonable allocation in a distributed manner where the agents can work collaboratively to maximize the network global objective function.

REFERENCES

[1] Laue HJ (1968), "Efficient methods for the allocation of resources in project networks". Unternehmensforschung 12:133-143.

[2] Davis, E. W. (1966), "Resource Allocation in Project Network Models - A Survey," The Journal of Industrial Engineering, vol 17: 177-188.

[3] Hillier, F.S. and G.J. Lieberman. Introduction to Operations Research Holden Day, 1980.

[4] Yanfeng Wang, James R. Perkins (2002), "Optimal Resource Allocation in New Product. Development Projects: A Control- Theoretic. Approach.", IEEE transactions on automatic control, Vol. 47, pp 1267-1276.

[5] Murphy, R. A. (1986) "A Private Fleet Model with Multi-Stop Backhaul." Optimal Decision Systems.

[6] Alvaro E. Gil, Kevin M. Passino: Stability analysis of network-based cooperative resource allocation strategies. Automatica 42(2): 245-250 (2006)

[7] Gil, A.E., K.M. Passino, S. Ganapathy, and A. Sparks, "Cooperative scheduling of tasks for networked uninhabited autonomous vehicles", Proceedings of the IEEE Conference on Decision and Control (Maui, Hawaii), (December 2003), 522–527.

[8] S. Andradóttir, H. Ayhan and D.G. Down. Dynamic server allocation for queueing net- works with flexible servers. Operations Research, 51:952-968, 2003.

[9] Song, Y., Zhang, M. T., Yi, J., Zhang, L., Zheng, L., Bottleneck Station Scheduling in Semiconductor Assembly and Test Manufacturing using Ant Colony Optimization, IEEE Transactions on Automation Science and Engineering, 4(4), 569 – 578, (2007)

[10] Low S. H. (2003). A duality Model of TCP and Queue Management Algorithms, IEEE/ACM Transactions on Networking (TON) ,Vol. 11 , No. 4, pp. 525-536.

[11] Kelly F.P. (1997). Charging and rate control for elastic traffic, Eur. Trans. Telecommunication. Vol. 8, pp. 33-37. Available: http://www.statslab.cam.ac.uk/~frank/ elastic.html.

[12] Kelly F.P., A. Maulloo, and D. Tan, (1998). Rate control for communication networks: Shadow prices, proportional fairness and stability, J. Oper. Res. Soc., Vol. 49, No. 3, pp. 237-252.

[13] Low S. H. (2000). A duality model of TCP flow controls, in Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, September 18-20. http://netlab.caltech.edu.

[14] Low S. H. (2002)b. Understanding Vegas: A duality Model, Journal of the ACM, Vol. 49, No. 2, pp 207-235. Available: http://netlab.caltech.edu/ pub.html.

[15] Dalalah Doraid. 2010. “Real time optimization flow control” 2010. Computer Networks, Vol. 54, No. 5, pp. 797-810.

[16] Dalalah Doraid and Omar Al-Araidah, (2010). Dynamic decentralised balancing of CONWIP production systems. International Journal of Production Research, Volume 48, Issue 13 , pages 3925 - 3941

[17] Hollot C., Misra V., Towsley D., and Gong W. B. (2001). A control theoretic analysis of RED, in Proc IEEE Infocom, pp. 1510-1519, Anchorage, Alaska, April 22-26. Available at: http://www-net.cs.umass.edu/papers.html.

[18] Lachlan L. H. Andrew, Stephen V. Hanly and Rami G. Mukhtar, (2007). Active Queue Management For Fair Resource Allocation in Wireless Networks, IEEE Trans. Mobile Computing.

[19] Lachlan L. H., Tony Cui, Andrew, Moshe Zukerman and Liansheng Tan, (2006). Improving the fairness of FAST TCP to new flows, IEEE Comm. Letters, 10(5), pp414-416.

[20] Lee J. Krajewski, Larry P. Ritzman , Manoj K. Malhotra “Operations management”, 9th ed., pearson.

[21] Morgen Witzel (2003). Fifty key figures in management. Routledge, 2003. ISBN 0415369770, p.96.

[22] Kelly Tom, Sally Floyd, and Scott Shenker. (2003). Patterns of Congestion Collapse, International Computer Science Institute, and University of Cambridge.

[23] Bertsekas D., 1997. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, 1989; republished by Athena Scientific.