Performance Modeling Of Computer Networks

Group - IV

Rhittotam De , 11CS30011
Shaik Nadeem , 11CS30033
Krishnendu Saha , 11CS30047

Multi-Resource Fairness: Objectives, Algorithms and Performance

Supervisor: Prof. Sandip Chakraborty
November 8, 2015

1. **Context:**
   Designing efficient and fair algorithms for sharing multiple resources between heterogeneous demands is becoming increasingly important. Applications include compute clusters shared by multitask jobs and routers equipped with middleboxes shared by flows of different types.

2. **Objectives:**
   1. Define previous Multi-resource sharing algorithms like Dominant Resource Flow (DRF) and Proportional Fairness (PF).
   2. Introduction of a new algorithm - Bottleneck Max Fairness.
   3. Define a fluid model to compare performance metrics.
   4. Talk about their strategy proofness.
   5. Identification of practical packet-based algorithms that realize PF and BMF sharing between flows in a router or network.
   6. Justifying the claim that DRF is not in fact the best objective in a dynamic demand environment.

3. **Fluid Model:**
   Consider \( J \) infinitely divisible resources of normalized capacity 1 to be shared by \( n \) transactions indexed by \( i \). Each transaction \( i \) requires amounts of each resource in fixed proportions and the amount allocated determines the rate at which the transaction progresses. We denote the requirements of transaction \( i \) by a vector \( \mathbf{a}_i = (a_{i1}, \ldots, a_{ij}) \) where, for definiteness, and without loss of generality, we set \( \max_j (a_{ij}) = 1 \). A resource for which \( a_{ij} = 1 \) is called a dominant resource for transaction \( i \).
   An allocation is defined by a vector of real numbers \( \mathbf{\phi} = (\phi_1, \ldots, \phi_n) \) such that \( \phi_i a_{ij} \) is the fraction of resource \( j \) allocated to transaction \( i \). The allocation must satisfy capacity constraints:

4. **Fairness Objectives:**
   In the following sections we talk about three major types of fairness:

   A. **DRF (Dominant Resource Fairness):**

   DRF was the first generalization of max min fairness resource sharing system for multiple resources. Max min fairness ensure that any user gets at least the minimum requirement of resources as per his demand, but only for one resource. DRF tries to extend it to multiple
resources. DRF satisfies many properties like Pareto efficiency, Strategy-Proofness, sharing incentive and envy-freeness. DRF provides incentives to users to share resources as no user is better off if resources are shared statically and equally. DRF has an edge over other two algorithms because of strategy-proofness, i.e., no user can alter his resource requirement vector to get more of his max required resource.

To realize the ideal DRF allocation of one can employ a water-filling algorithm with the dominant resource requirement normalized to 1, the $\phi_i$ are increased at the same rate until some resource is fully used; transactions using that resource are frozen while rates of the others with non-zero requirements on non-saturated resources are increased together until a second resource is full; the process continues until all the $\phi_i$ are frozen.

DRFQ determines the order in which packets are served at each resource through an adaptation of start-time fair queuing (SFQ). The virtual start time $S_i^k$ of packet $k$ of flow $i$ is determined recursively,

$$S_i^k = \max \left( V(u_i^k), S_i^{k-1} + \max_j \{a_{ij}\} \right)$$

where $u_i^k$ is the packet arrival time and the virtual time function of real time $t$, $V(t)$, is set equal to the largest start time at $t$ of any packet to have begun service at any resource. Packets are served at each resource in increasing order of the $S_i^k$.

B. PF (Proportional Fairness):

PF is another one of the multi-resource sharing algorithms which is said to achieve a better efficiency-fairness tradeoff than DRF. Though it was criticized for not being strategy-proof, though it is not considered as a discriminating factor in case of dynamic demand which is a more realistic case. PF ensures that users achieve than minimum resource requirement while striking the balance between efficiency (getting max resources) and fairness. Proportional fairness allocates resources based on various factors like demand, weightage etc. PF also follows along DRF in terms of its properties like pareto-efficiency, envy-freeness, sharing-incentives etc.

To realize the required packet rates we can adapt the SFQ algorithm. Start times $S_{ij}^k$ are defined recursively and independently for each resource $j$,

$$S_{ij}^k = \max \left( V_j(u_{ij}^k), S_{ij}^{k-1} + a_{ij}/Q_{ij} \right)$$

where $u_{ij}^k$ is the packet arrival time at resource $j$ and virtual time $V_j(t)$ is the start time at $t$ of the last packet to have begun service at $j$.

C. BMF (Bottleneck Max Fairness):

BMF is based on the intuitive alternate of the objective proposed by DRF as “no justified complaint”. BMF adopts simpler version stating that a transaction has no justified complaint if it receives at least a fraction $1/n$ ($n$ : number of transactions) of the bottleneck resource. It is
restricted even further by ensuring every transaction receives an allocation which is maximum for that resource. BMF satisfies all the properties that PF does. BMF claims to have a much simpler implementation of its algorithm.

To realize BMF using SFQ, start times must be calculated as follows,

$$S_{ij}^k = \max \left( V_j(u_i^k), S_{ij}^{k-1} + a_{ij} \right)$$

Further we discuss all the properties pertaining to each of these algorithms. We will also describe the necessity of those properties in the context of multi-resource fairness distribution.

5. Definition of different properties of these algorithms:

Here we describe the properties that most of these algorithms satisfy at the same time with a few exceptions. All the three algorithms satisfy the following properties.

**Pareto-efficiency:** This basically states that the allocation of resources be such that it be impossible to make the share of any transaction better without making it worse for any other transaction. This is considered the minimal notion of efficiency and does not always result in an overall desirable (equal by weightage) distribution.

**Envy-freeness:** As the name goes, no transaction should envy other, i.e. no transaction should be better of with the allocation of any other transaction. Each should get the best it can ever get. This is required so that no transaction tries to manipulate its requirement vector to get a better result.

**Sharing-incentive:** Here we state that no user will be better of in a system with statically and equally distributed resources. This means that a user or transaction may be better off if it shares its existing resources as per its requirements. This will lead to better allocation status for all transactions.

**Single-resource fairness:** For a single resource, the solution should reduce to max-min fairness, i.e. all allocation of the single resource is equal to 1/n.

**Single-bottleneck fairness:** If there is a resource that is percent-wise the most required resource for every user, then the solution dials down to max min fairness for that resource, i.e all users have 1/n th of that resource.

**Scale-invariance:** The sharing remains the same even when the resource requirement vector of any transaction is multiplied by a scaling factor.

There is another property that is theoretically only displayed by DRF
**Strategy-proofness:** This basically means that a transaction can never be better off by lying about its requirements. This provides incentive compatibility as transaction cannot improve the allocation by lying.

**6. Performance:**

Markovian demand model: We suppose transactions belong to one of K classes and transactions of class k arrive as a Poisson process of rate $\lambda_k$. Class k transactions have requirement vector $(a_{k1}, \ldots, a_{kJ})$ and bring an exponentially distributed amount of work (number of tasks $\times$ mean task duration or size of a flow in bytes, say) of mean $1/\mu_k$. The state vector $(n_1, \ldots, n_K)$, giving the current number of transactions in progress, is then a Markov process with component-k birth rate $\lambda_k$ and death rate $n_k \phi_k \mu_k$. This process is stable as long as loads $\rho_k = \lambda_k / \mu_k$ satisfy the following inequalities,

$$\sum_{k=1}^{K} \rho_k a_{kj} < 1,$$

for $j = 1, \ldots, J$. In this case, the process has a stationary distribution $\pi(n)$ from which we can compute performance measures like expected completion times. In the following we compare algorithm performance via the mean service rate $\gamma_k$, defined as the ratio of the mean work $1/\mu_k$ to the mean completion time. Its reciprocal is thus a normalized completion time. Using Little’s law, we find $\gamma_k = \lambda_k / E(n_k)$ where $E(n_k)$ is the mean number of class k transactions in progress.

![Figure 1](image1.png)

**Figure 1:** Service rates $\gamma_k$ against resource load for BMF and DRF with balanced load: $a_1 = (1,1)$, $a_2 = (1,1)$, $a_3 = (1,1)$; $\rho_1 = \rho_2 = \rho_3$; for each allocation we have, $\gamma_1 = \gamma_2 > \gamma_3$. 
7. Lessons:

1. The popularity of DRF is not justified as alternative fairness objectives like PF and BMF show better efficiency-fairness tradeoff.

2. Though PF and BMF model aren’t strategy-proof, i.e., users can manipulate their allocation by falsely boosting their requirements, but the gain of such malicious users depends on the knowledge of competitors’ requirements, which is clearly inconceivable in the context of highly dynamic environments.