# The Passport Control Problem or How to Keep an Unstable Service System Load Balanced ?

A. ITAI
*Computer Science Department, Technion*

M. RODEH
*Computer Science Department, Technion*

H. SHACHNAI
*Computer Science Department, Technion*

**Abstract**

In many real life situations (such as the passport control system in Tel Aviv airport, department stores, and more) parallel queues are formed in front of control stations. Typically, some of the stations are manned while others are not. As the queues build up, management assigns additional officers to the unmanned stations. When this happens—some people move to the newly manned queues from nearby busy queues. In anticipation, people may prefer to line up in busy queues next to unmanned ones.

Mathematically we discuss the problem of dynamic arrangement of the queues in a service system where at any time each server can be in either active or inactive mode. We seek a partition of customers to queues, that minimizes the expected wait time of a customer in each of the active stations, thereby keeping the system *balanced* at all times. We study two balancing algorithms which we call *Split* and *Trim*. For the Split algorithm we discuss a special case (the stations are ordered on a line and a single unmanned station is at one end). We show how an optimal partition can be calculated recursively. We then give partitions that approximate the minimal expected wait time within a factor of $1 + \varepsilon$, for some small $0 < \varepsilon < 1$, under each of these algorithms.

**Keywords:** Queuing, load balancing, response time, service systems.

## 1 Introduction

Real life queuing systems appear in two major variants:

1. A single line which feeds multiple service stations in front of which no wait queue exists. This is the common practice in many banks.

2. Multiple queues, each of which is being served by a single service station. This is the system in use at the Tel Aviv international airport (as well as many others), in department stores, and elsewhere.

In this paper we concentrate on queuing systems of the second type. In such systems, as the queues build up, management assigns additional officers to the unmanned stations. When this happens—some people move to the newly manned queue from nearby busy queues. In anticipation, people may prefer to line up in busy queues next to unmanned ones. The problem of interest to us is the expected lengths of the different queues as a function of the configuration of the manned/unmanned queues.

More formally, the *passport control problem* is the problem of keeping a multi-queue service system load balanced; the system consists of a set of $N$ service stations positioned on the line. Each service station has a single server, and a wait_queue, from which customers are picked up in a first-come-first-served order; all stations provide the same type of service, with equal speed, thus an arriving customer can join any of the $N$ queues.

The system is *unstable* in the sense that at any point in time each of the servers can be in either an *active* or *inactive* mode. An active server $i$, $0 \leq i \leq N-1$ gives service to the first customer in queue $i$ (whenever the queue is non-empty). When server $i$ becomes inactive the customers lined up in its queue move to any of the other service stations that are currently active. When an inactive server $i$ becomes active, customers from other active queues move to station $i$ and form a queue there. Thus, at any time $t$, the number of queues in the system is $n$, where $0 \leq n \leq N$.

In this paper we study rules for distributing the workload in the system, so as to minimize the average response time for customers in each of the queues. Indeed, such rules should depend on:

(i) The number of active service stations and their location in the system at any given time.

(ii) The frequency of changes in the number of active servers.

(iii) The balancing strategy used when such a change occurs, namely, the rule which determines how the queues are rearranged when any of the servers becomes active/inactive.

For a given state of the system, we define a rule that distributes the workload among the active queues, taking into account future transitions of servers between the active and inactive modes. It is important to note that there is a large set of rearrangement rules, some of which dynamically modify the distribution of customers to the queues. In the present context we focus on a class of *static* rules, which maintain a fixed partition of customers to the queues when the number of

active servers is fixed. The efficiency of the proposed rules is measured against an (unknown) *optimal* rule.

Assuming that each of the inactive servers can become active at any time, our objective is to define a partition $\vec{l}$ of customers to the queues, that will be fixed as long as the number of active servers is fixed. In other words, while none of the servers changes mode, new customers join the queues so that the partition $\vec{l}$ is maintained. We seek a partition $\vec{l}$ that minimizes the expected wait time of customers in each of the active stations, thereby keeping the system *balanced* at any time.

## 1.1 Related Work

We briefly mention some related work:

In the context of queuing theory, a model close to the system model discussed above is the *server of the walking type* model, where each server in the system can become inactive for a certain amount of time, upon completion of a service period [2, 3, 6, 7]. Related work on this model studied the steady state distribution on queue lengths and on the wait times of customers, when some probabilistic assumptions are used, e.g., for determining the length of the "vacation" taken by a server. However, no balancing algorithms are used in this model, i.e., when the server is inactive the customers are assumed to line up in its service station, until the server becomes active again.

Another related problem is the dynamic server problem introduced in [1]. This problem refers to a task system in which the number of available servers can change at any time. The authors present competitive algorithms for this problem; the model is slightly different than our service system, as customers do not line up in queues for service; they are distributed in many sites in a computer network, thus rearrangements of the queues are inapplicable. A similar dynamic server model was used also in the study of load balancing schemes for multimedia systems (see in [4, 5]).

## 1.2 Main Results

We now summarize our results. For a system of $N$ queues, in which $K$ servers are inactive, for some $1 \leq K < N$, the main results presented in this paper are

- The Split balancing algorithm:
  - (i) For the case where there is a single inactive server in the system, we show how an optimal partition can be calculated recursively, based on the knowledge of the activation probability of that server at any time.
  - (ii) We give a partition which yields a $1 + O(1/N)$- approximation to the minimal expected wait time at any queue.

- The Trim balancing algorithm, which is shown to provide an $(1+\varepsilon_1)$-approximation to the minimal expected wait time, and $(1+\varepsilon_2)$-approximation to the minimal balance ratio of the system, where $0 < \varepsilon_1, \varepsilon_2 < 1$ are small.

### 1.3 Organization of the Abstract

In Section 2 we give some notation and define our performance measures. In Sections 3 and 4 we derive results for the case where $K = 1$. In Section 3 we present the Split algorithm, for which we show (in Section 3.1) how an optimal partition can be computed recursively. We then give (in Section 3.2) a partition that is a close approximation to the optimal.

In Section 4 we propose the Trim algorithm and give a partition that is a better approximation to the optimal, under this scheme. In Section 5 we show how our results can be extended to apply to the case, where the number of inactive servers in the system is any $1 \leq K < N$.

## 2 Preliminaries

For a given time interval $[0, T]$, during which we optimize the system's performance, we define a partition of $[0, T]$ to sub-intervals, that are handled as separate phases. We assume that each phase is long enough, so by the time it ends the system has stabilized, and that the number of active servers in a phase is fixed. Moreover, we assume that time is discrete, starting at $t = 0$.

Suppose that at some point in time $n$ out of the $N$ servers are active, and for some $L > N$, at any time there are $L$ customers in the system. Each server completes the service of a single customer within one time unit, whereupon the customer leaves the system. For any inactive server $j$, there is a fixed probability $p_j$, that the server becomes active between the present and the next time unit[1]. We seek a partition of the $L$ customers to the $n$ active queues, such that the expected wait time of customers in each of the queues is minimized.

At any time $t$ each customer in the system can be represented by a pair $(i, j)$, where $0 \leq i \leq N-1$ is the number of the queue in which the customer is waiting, and $1 \leq j \leq L$ is the position of the customer in that queue. A *Balancing Strategy* $\mathcal{S} : \{0, \ldots, N-1\} \times \{1, \ldots, L\} \rightarrow \{0, \ldots, N-1\} \times \{1, \ldots, L\}$ is a function that maps each customer $(i, j)$ to $(i', j')$ using some transition rule that balances the queues. A balancing strategy is used only when the number of active servers changes.

Let $\vec{l} = (l_0, \ldots, l_{N-1})$ denote a partition of the customers to the $N$ queues $Q_0, \ldots, Q_{N-1}$. The quality of such a partition with respect to a given balancing strategy $\mathcal{S}$ is measured as follows: Let $W_i(\vec{l}, \mathcal{S})$ denote the expected wait time of

[1] We simplify the calculations by assuming that arrival/departure of a customer and the activation of a server cannot occur simultaneously.

a customer joining queue $i$, when the partition is $\vec{l}$ and the balancing strategy is $\mathcal{S}$. Let

$$\text{max\_W}(\vec{l},\mathcal{S}) = \max_{0\le i\le N-1} W_i(\vec{l},\mathcal{S}) \ ,$$

$$\text{min\_W}(\vec{l},\mathcal{S}) = \min_{\{0\le i\le N-1; l_i>0\}} W_i(\vec{l},\mathcal{S})$$

be the maximal and the minimal expected wait time under strategy $\mathcal{S}$ of the partition $\vec{l}$, respectively. We denote by $W_{OPT}(L,\mathcal{S})$ the expected wait time of an optimal arrangement of $L$ customers in the $n$ queues, under the strategy $\mathcal{S}$.

**Definition 1** *The balance-ratio of a partition $\vec{l}$ under a balancing strategy $\mathcal{S}$ is given by*

$$\beta(\vec{l},\mathcal{S}) = \frac{\textit{max\_W}(\vec{l},\mathcal{S})}{\textit{min\_W}(\vec{l},\mathcal{S})} \ .$$

**Definition 2** *The partition $\vec{l}$ of $L$ yields a $(1+\delta)$-approximation to the minimal expected wait time, under a balancing strategy $\mathcal{S}$, if*

$$\textit{max\_W}(\vec{l},\mathcal{S})] \le (1+\delta)W_{OPT}(L,\mathcal{S}) \ .$$

# 3 The Split Algorithm

Assume first that the servers are linearly ordered, $Q_0$ is inactive, and $l_i$ customers wait at $Q_i$, $i = 1,\ldots,N-1$. The probability that a server appears at $Q_0$ in the next time slot is $p$. Let $\bar{L} = L/N$ be the average queue length when all the stations are active.

In this section we discuss a balancing strategy that we call *Split*. We first show how an optimal partition can be calculated recursively. We then show that if $p > 1/\bar{L}$ then a simple partition based on accumulating a large amount of customers near the inactive server can yield expected wait times that are close to the optimal within a factor of $1 + O(1/(N\bar{L}\,p))$.
The Split balancing strategy is as follows:

When a server arrives at $Q_0$, every other customer from $Q_1$ moves to $Q_0$. After that both $Q_0$ and $Q_1$ have $\bar{L}_1 = l_1/2$ customers.

At step $i$: queues $Q_0,\ldots,Q_{i-1}$ have each $\bar{L}_i = \frac{1}{i}\sum_{j=1}^{i-1} l_j$ customers. The customers of $Q_i$ who have to wait more than $\bar{L}_i$ time, evenly split to queues $Q_0,\ldots,Q_i$ in a cyclic manner, i.e., customer $\bar{L}_i+1$ moves to $Q_0$, ..., customer $\bar{L}_i+i$ moves to $Q_{i-1}$, customer $\bar{L}_i+i+1$ remains in $Q_i$, customer $\bar{L}_i+i+2$ moves to $Q_0$ etc.

A pseudo-code of the Split algorithm is given in Figure 1.

**Algorithm Split**
Input: A set of customer records $C[1], \ldots, C[L]$, indicating the queue index and position of each customer; the sequence $0, l_1, \ldots, l_{N-1}$.
Output: The updated set $C[1], \ldots, C[L]$.

```
for i := 1 to N − 1 do {
    let x_{L̄_i+1}, ..., x_{l_i} be the customers at positions L̄_i + 1, ..., l_i
    of Q_i;
    k := 0;
    L̄_{i+1} := L̄_i + 1;
    for j := L̄_i + 1 to l_i do {
        C[x_j].position := L̄_{i+1};
        C[x_j].queue := k;
        k := k + 1;
        if k > i then {
            k := 0;
            L̄_{i+1} := L̄_{i+1} + 1;
        }
    }
}
```

Figure 1: The Split Algorithm

## 3.1 Finding an Optimal Partition

Assume that the arrival probability of a server at $Q_0$ is a fixed parameter $p$. For any $t \geq 1$, the probability that the server arrives at $Q_0$ at time $t$ is

$$p_t = (1-p)^{t-1}p \ . \tag{1}$$

Let $W_i(\vec{l}, \textit{Split})$ be the expected wait time of a customer at $Q_i$, given that the initial lengths of the queues are $\vec{l} = (0, l_1, \ldots, l_{N-1})$.

Let $\Delta_i = l_i - \bar{L}_i = l_i - \frac{1}{i}\sum_{j<i} l_j$ denote the excess number of customers in $Q_i$ over the average number of customers in queues $Q_0, \ldots, Q_{i-1}$ after the customers of these queues moved in response to the arrival of the new server at $Q_0$, but just before the customers of $Q_i$ moved. Consider a customer $A$ that arrives at time $t = 0$. We simplify the calculations by allowing the queue lengths to be non-integral numbers. If the server arrives at time $t$, $1 \leq t \leq \Delta_i$, then $\frac{i}{i+1}(\Delta_i - (t-1))$ customers of $Q_i$ that are before $A$ leave $Q_i$ to join $Q_0, \ldots, Q_{i-1}$. Thus the wait time is reduced by that amount.

We will use the equality

$$p\sum_{t=1}^{k}(1-p)^{t-1}t = \frac{1}{p}\left[1-(1-p)^k(kp+1)\right] \ . \tag{2}$$

The expected wait time at $Q_i$ is

$$\begin{aligned} W_i(\vec{l}, \textit{Split}) &= \sum_{t=1}^{\Delta_i} p_t \left( l_i - (\Delta_i - t + 1)\frac{i}{i+1} \right) + \sum_{t>\Delta_i} p_t l_i \\ &= l_i - \frac{i}{i+1}\left[ \Delta_i \sum_{t=1}^{\Delta_i} p_t - \sum_{t=1}^{\Delta_i} t p_t + 1 - (1-p)^{\Delta_i} \right] \\ &= l_i - \frac{i}{i+1}\left[ \Delta_i - \frac{1}{p} + \frac{1}{p}(1-p)^{\Delta_i} + 1 - (1-p)^{\Delta_i} \right] \ . \end{aligned}$$

Given queues of length $l_1, \ldots, l_{N-1}$, a new customer would join the queue with minimum expected waiting time. Hence the system will become stable when

$$W_1(\vec{l}, \textit{Split}) = W_2(\vec{l}, \textit{Split}) = \cdots = W_{N-1}(\vec{l}, \textit{Split}) \ .$$

Let $l_1$ be given, we first calculate $W_1(\vec{l}, \textit{Split})$ and choose $l_2$ such that $W_1(\vec{l}, \textit{Split}) = W_2(\vec{l}, \textit{Split})$. We can easily perform this calculations, since $W_2(\vec{l}, \textit{Split})$ depends on $p$ and on $\Delta_2 = l_2 - l_1/2$. We continue in this fashion to compute the values of $l_3, \ldots, l_{N-1}$. The key observation is that $l_i$ depends only on $p$ and $\Delta_i = l_i - \frac{1}{i}\sum_{j<i} l_j$. Hence $l_i$ depends only on $p$ and $l_1, \ldots, l_{i-1}$, which we have already computed.

On the other hand, if $L$, the total number of customers, is known we can perform a binary search on $l_1$ to find values $l_1, \ldots, l_{N-1}$ that stabilize the system and satisfy $\sum_{i=1}^{N-1} l_i = L$.

For $p = 1$, we get

$$W_i(\vec{l}, \textit{Split})_{|p=1} = l_i - \frac{i}{i+1}\Delta_i = l_i - \frac{i}{i+1}(l_i - \frac{1}{i}\sum_{j<i} l_j) = \frac{1}{i+1}\sum_{j=1}^{i} l_j = \bar{L}_{i+1} \ .$$

Since $\bar{L}_2 = l_1/2$, in this case the system stabilizes when $l_1/2 = l_2 = \cdots = l_{N-1}$. In other words, the first queue (next to the inactive server) has twice as many customers in it—since with probability $p = 1$ at time $t = 1$ a server appears at $Q_0$, half of the waiting customers move to $Q_0$, and all queues are of equal length.

## 3.2 Approximating the Minimal Wait Time

Assume now that the inactive server is in position $h$, for some $0 \le h < N-1$. In the following we show how the minimal wait time can be closely approximated for this case.

We propose to use a static partition $(l_0, \ldots, l_h, \ldots, l_{N-1})$, in which all the queues that are of the same distance from the inactive station have the same length. In fact, we show that a close approximation to the minimal wait time can be obtained by choosing a partition, in which the queues next to the inactive server have the same length $l'$, and all the other queues in the system have the same length $l''$. Hence, we only need to compute the ratio $l'/l'' \geq 1$.

When the inactive server is positioned at the end of the line, the queue will be balanced (upon activation of this server) by the Split algorithm as given in Figure 1. When the server is in position $h$, for some $0 < h < N-1$, the queues will be balanced using the Split algorithm with a slight modification: The balancing process will handle in phase $i$ all queues that are of distance at most $i$ from the inactive server. Thus, after the first phase the queues $h-1, h, h+1$ will be of equal length. After the second phase the queues $h-2, h-1, h, h+1, h+2$ will be of equal length, and so on.

Let $s \in \{1, 2\}$ denote the number of immediate neighbors of the inactive server. For $p \geq 1/\bar{L}$ we define the partition $\vec{l}_{Split} = (l_0, \ldots, l_{N-1})$, where

$$l_i = \begin{cases} \frac{1}{s}((s+1)\bar{L} - \frac{1}{p} + \frac{\bar{L}}{p(N-1)} & i \text{ is a neighbor of inactive server} \\ 0 & i = h \\ \bar{L}(1 + 1/(p(N-1)) & \text{otherwise} \end{cases}$$

**Theorem 1** *The partition $\vec{l}_{Split}$ yields a $(1 + 1/(\bar{L}\, p(N-1)))$- approximation to the minimal expected wait time, when the balancing algorithm is Split, that is,*

$$W_i(\vec{l}_{Split}, \text{Split}) \leq \left(1 + \frac{1}{\bar{L}\, p(N-1)}\right) W_{OPT}(L, \text{Split}) \;\forall\; 0 \leq i \leq N-1 \;. \tag{3}$$

**Proof.** Let $p_t$ be as defined in (1). For the case $h = 0$,

$$W_1(\vec{l}_{Split}, \textit{Split}) = \sum_{t=1}^{l_1} p_t(t + \frac{l_1 - t}{2}) + \sum_{t > l_1} p_t l_1 \;.$$

Using equation (2) we have

$$\begin{aligned} W_1(\vec{l}_{Split}, \textit{Split}) &= \frac{l_1}{2} + \frac{1}{2}(1 - (l_1+1)(1-p)^{l_1} \\ &\quad + \frac{1 - p - (1-p)^{l_1+1}}{p}) + \frac{l_1}{2}(1-p)^{l_1} \\ &= \frac{1}{2}(l_1 + \frac{1}{p}) - \frac{(1-p)^{l_1}}{2p} \;. \end{aligned}$$

Taking $l_1 = 2\bar{L} - \frac{1}{p} + \frac{\bar{L}}{p(N-1)}$ we get inequality (3). A similar computation for $h \neq 0$ gives the statement of the theorem. □

# 4 The Trim Algorithm

In this section we propose a balancing strategy called Trim. We define a partition that is shown to provide a close approximation to the minimal expected wait time, while maximizing the balance in the system.

The following is an informal description of the algorithm: Consider first the case where the inactive server is at the end of the line (i.e., $h = 0$). Denote by $X$ the activation time of the server, then $X \sim G(p)$.

Suppose that initially there are $l_i$ customers in queue $i$, $1 \leq i \leq N-1$. When server $0$ becomes active, customers from queues $1$ through $N-1$ transfer to $Q_0$ as follows.

> If a customer is at position $j$ in $Q_i$, then all customers in queues $Q_1, \ldots, Q_i - 1$, whose positions are larger that $\bar{L}$, will line up in $Q_0$ before that customer. In addition, any customer $(i, r)$ with $\bar{L} \leq r < j$ will also precede $(i, j)$ in $Q_0$.

We call this balancing strategy Trim, since it trims all the long queues in the system, and transfers the excess customers to $Q_0$, until all queues are of the same length $\bar{L}$. A pseudo-code of the Trim algorithm is given in Figure 2.

**Algorithm Trim**
Input: A set of customer records $C[1], \ldots, C[L]$, indicating the queue index and position of each customer; the sequence $0, l_1, \ldots, l_{N-1}$.
Output: The updated set $C[1], \ldots, C[L]$ corresponding to the partition $(\bar{L}, \ldots, \bar{L})$.

```
for k = 1 to L
  with C[k] do
    if C[k].position > L̄ then {
      C[k].position := C[k].position − L̄ + Σ_{j=2, l_j > L̄}^{C[k].queue−1} (l_j − L̄);
      C[k].queue := 1;
    }
```

Figure 2: The Trim Algorithm

Assume now, that the inactive server is located in station $h$, for some $1 \leq h < N-2$. For this case we define the following extension of the Trim algorithm. If a customer $C[k]$ is in distance $i$ from the inactive server, then when the server becomes active

- All the customers in positions $j > \bar{L}$, that belong to queues in distances that are smaller than $i$ will line-up in $Q_h$ before $C[k]$.

- All the customers in positions $j > \bar{L}$, who belong to the queues $\{h-i, h+i\}$ then join in random order at the end of $Q_h$, including $C[k]$.

We call this strategy the *Random Trim*.

In the next result we define a static partition of the customers to queues, which is shown to be efficient with respect to our two measures, when the balancing strategy is (Random) Trim.

**Theorem 2** *Let $M$ be the median of the random variable $X$, and let*

$$r = \lg_2(\bar{L}\,/M)\ ,$$

*and*

$$r^* = \max\left\{ r' \geq r \mid r' \leq \frac{2^{r'}-1}{2^{r'-r}} \right\}\ . \tag{4}$$

*If $h = 0$ $h' = 1$, otherwise $h' = h-1$. Denote by $\vec{l}_{Trim}$ the partition $(l_0, \ldots, l_{h-1}, 0, l_{h+1}, \ldots, l_N)$, where*

$$l_i = \begin{cases} 0 & i = h \\ \bar{L}(2 - 1/2^{r^*}) & i = h' \\ \bar{L}(1 + 1/(2^{r^*}(N-2)) & \textit{otherwise.} \end{cases}$$

*Then for any $r \geq 1$ and a system of $N > 2$ stations,*

*(i) $\vec{l}_{Trim}$ yields a $(1 + \frac{1}{2^{r^*}(N-2)})$-approximation to the minimal expected wait time.*

*(ii)* $\beta(\vec{l}_{Trim}) \leq \frac{1+\frac{1}{2^{r^*}(N-2)}}{1-\frac{1}{2^{r^*}}}$ .

**Proof.** We give the proof for $h = 0$. The generalization to arbitrary $h$ is straightforward. We first note, that if $M \leq \bar{L}/2$ there exists $r^* \geq r$ satisfying (4) (since clearly, $r^* = r$ satisfies the inequality). Now, we show separately the approximation-bound and the balance-ratio of the partition $\vec{l}_{Trim}$.

(i) We first show, that

$$W_i(\vec{l}_{Trim}, Trim) \leq (1 + \frac{1}{2^{r^*}(N-2)}) W_{OPT}(L, Trim) \quad \forall\ 1 \leq i \leq N-1\ . \tag{5}$$

For $l'_1 = \bar{L} \cdot (1 - 1/2^{r^*})$, let $A$ be the event

$$\text{“The inactive server becomes active within less than } l'_1 \text{ time units”}\ , \tag{6}$$

then

$$\begin{aligned} W_1(\vec{l}_{Trim}, Trim) &= Prob(A)l_1' + (\bar{L} + l_1')(1 - Prob(A)) \\ &= l_1' + \bar{L}(1 - P(A)) \ . \end{aligned}$$

Since $M = \bar{L}/2^r$ and $r^*$ satisfies (4), the probability that the server remains inactive in the next $l_1'$ time units is

$$1 - P(A) = (1-p)^{\bar{L}/2^r \cdot \frac{2^{r^*}-1}{2^{r^*}-r}} \leq \frac{1}{2^{r^*}} \ . \tag{7}$$

Hence,

$$W_1(\vec{l}_{Trim}, Trim) = \bar{L}(1 - \frac{1}{2^{r^*}}) + \bar{L}\frac{1}{2^{r^*}} \leq \bar{L} \ ,$$

Since $W_{OPT}(L, Trim) \geq \bar{L}$, obviously inequality (5) holds for $i = 1$.
For queues $i = 2, \ldots, N-1$ we have

$$l_i = \bar{L}(1 + \frac{1}{2^{r*}(N-2)}) \ ,$$

and again inequality (5) holds.

(ii) We now bound the balance-ratio of the system under Trim. Since for any $1 \leq i \leq N-1$,

$$W_i(\vec{l}, Trim) \leq (1 + \frac{1}{2^{r^*}(N-2)})\bar{L} \ ,$$

it remains to show that

$$\text{min\_W}(\vec{l}) \geq \bar{L}(1 - \frac{1}{2^{r^*}}) \ .$$

We note, that the choice of $l_1$ implies

$$W_1(\vec{l}, Trim) \geq \bar{L}(1 - \frac{1}{2^{r^*}}) \ .$$

Let $B$ be the event

"The inactive server becomes active within the next $\bar{L}/(2^{r^*}(N-2))$ time units". (8)

Then for $i = 2, \ldots, N-1$

$$W_i(\vec{l}_{Trim}, Trim) \geq Prob(B)(l_1 - \bar{L}) + (1 - Prob(B))\bar{L} \geq l_1'$$

which yields the statement of the theorem.

□

**Corollary 1** *If $p \geq \frac{3}{\bar{L}}$ then the partition $\vec{l}_{Trim}$ defined in Theorem 2 gives*

*(i) a $(1 + \frac{1}{8(N-2)})$-approximation to the optimum*

*(ii) $\beta(\vec{l}, Trim) \leq \frac{9}{7}$.*

# 5 Extensions

In this section we discuss the case where the number of inactive servers is any $1 \leq K \leq N-2$. We show how our results for the case $K = 1$ can be used to obtain similar bounds for the Split and the Trim algorithms.

We handle separately two cases:

(i) If $K \leq N/2$, then we can divide the set of queues to $K$ *regions*, such that each region $j$ contains exactly one inactive station, and $A_j \geq 1$ active stations, where $\sum_{j=1}^{K} A_j = N - K$ (The stations in each region are not necessarily neighboring stations on the line). The total number of customers in the queues of region $j$ is $A_j \cdot \bar{L}$ . Using The Split or the Trim algorithm for each region with $K = 1$, we have the results in Theorems 1 and 2 respectively.

(ii) For the case where $K > N/2$ we choose one of the active queues in the system $0 \leq i^* \leq N-1$ to be the *master queue*. Let $\bar{L}\ (K) = L/(N-K)$ be the average number of customers when divided equally among the active queues. When $K \geq 1$, we take $\bar{L} = \bar{L}\ (K-1)$. We can then use the partitions $\vec{l}_{Split}$ and $\vec{l}_{Trim}$ as defined in Sections 3 and 4 for the case $K = 1$. Suppose that the balancing algorithm used in the system is Trim. Then upon activation of a server at $Q_h$, first the customers lined up in position larger than $\bar{L}$ in $Q_{i^*}$ move to $Q_h$. Then batches of customers (whose positions were $\bar{L}$ or larger) from all the other queues join $Q_h$ in arbitrary order, one batch at a time. Thus, in each of the active queues there are exactly $\bar{L}$ customers. Now, we proceed by taking $K = K - 1$, and by modifying $\bar{L} = \bar{L}\ (K)$ accordingly. Any new customers will join the queues so as to form the partition $\vec{l}_{Trim}$ with the updated value of $\bar{L}$ .

Note, that when the number of inactive servers in the system is $K$, the minimal expected wait time is at least $\bar{L}\ (K)$, thus, the approximation bound in Theorems 1 and 2 hold for any number of inactive servers.

# References

[1] CHARIKAR, M., HALPERIN, D., AND MOTWANI, R. A queue with server of walking type. In *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)* (San Francisco, January 1998).

[2] GELENBE, E., AND IASNOGORODSKI, R. A queue with server of walking type. *Annales de l'Institut Henry Poincaré, Série B (Probabilités et Statistiques) XVI, no. 1* (1980), 63–73.

[3] GELENBE, E., AND MITRANI, I. *Analysis and Synthesis of Computer Systems*. Academic Press, 1980. Ch. 2, Section 2.2.

[4] GOLUBCHIK, L., AND LUI, J. C. S. Bounding of performance measures for a threshold-based queuing system with hysteresis. In *ACM SIGMETRICS Conference* (1997), pp. 147–157.

[5] LIE, P. W. K., LUI, J. C. S., AND GOLUBCHIK, L. Threshold-based dynamic replication in large-scale video-on-demand systems. In *Eighth International Workshop on Research Issues in Database Engineering (RIDE)* (Orlando, February 1998).

[6] SHACHNAI, H., AND YU, P. S. On analytic modeling of multimedia batching schemes. In *Proceedings of the 3rd International Workshop on Multimedia Information Systems (MIS'97)* (Como, September 1997).

[7] SKINNER, C. E. A priority queuing model with server of walking type. *Operations Research 15* (1967), 278–285.