Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 169026, 10 pages
doi:10.1155/2012/169026

Research Article

Resource Management in Satellite Communication Systems: Heuristic Schemes and Algorithms

Shahaf I. Wayer and Arie Reichman

School of Engineering, Ruppin Academic Center, Emek Hefer, Israel

Correspondence should be addressed to Arie Reichman, arier@ruppin.ac.il

Received 26 February 2012; Revised 30 April 2012; Accepted 1 May 2012

Academic Editor: Yi Su

The high cost of frequency bandwidth in satellite communication emphasizes the need for good algorithms to cope with the resource allocation problem. In systems using DVB-S2 links, the optimization of resource allocation may be related to the classical multi-knapsack problem. Resource management should be carried out according to the requests of subscribers, their priority levels, and assured bandwidths. A satisfaction measure is defined to estimate the allocation processes. Heuristic algorithms together with some innovative scaling schemes are presented and compared using Monte Carlo simulation based on a traffic model introduced here.

1. Introduction

1.1. DVB-S2 in Satellite Communication. The most popular standard today in satellite communication is DVB-S2 [1], designed for transmission of digital television broadcasting. This relatively new standard is gradually replacing the classic DVB-S [2] in new systems as well as in existing systems. It is also used in new VSAT systems in forward link as well as in reverse link, in cases when the required data rates are higher than those supplied by DVB-RCS [3]. DVB-S2 is also used in satellite modems for point-to-point applications. In this paper we focus on a system that consists of DVB-S2 links with dynamic changes in the required parameters of the links due to changes in the traffic demands.

1.2. Resource Management. In order to save the operational cost of the satellite communication system, transmissions that share the total available frequency band have to be fairly divided according to the requirements of the subscribers [4]. For a star-type system with a central HUB that communicates with subscriber terminals, the control is conducted by the control center in the HUB according to subscriber needs. For other type of systems, like point-to-point, the control is conducted by a dedicated communication and control system that operates in addition to point-to-point links.

Regardless of the way the control is performed, a resource allocation algorithm has to be designed for optimal performances and best satisfaction.

In commercial systems run by a service provider, the Service Level Agreement (SLA), which is a set of parameters that defines the capabilities and the paid rights of every terminal in the system, connects the service provider and the subscribers. The Committed Information Rate (CIR) is the rate that is guaranteed by the system. Typically it is low enough so that the system can guarantee the user of this specific rate. The Peak Information Rate (PIR), on the other hand, is the maximum rate that may be allocated. When the user asks for a rate higher than CIR and lower or equal to PIR, the system will allocate this rate if available. When a user asks for a rate higher than PIR, the system will consider it as if the request was equal to PIR. The service provider has difficulty in providing the Committed Rate with certainty. In this research, we assumed that the bandwidth of a terminal is proportional to its rate. We use the term *assured bandwidth* as a quantity that the system can provide *most of the time* but not always.

The terminals have priority levels that determine their significance according to their payment to the service provider. In a situation of low resources, requests from high priority terminals may cause the system to decrease the rate of low priority terminals to their minimum.

In the course of the resource allocation cycle, subscribers issue requests for bandwidth. The bandwidths they own are removed from the spectrum list. The resource allocation process follows, and the bandwidths are reconfigured over the spectrum. This whole process of removing spectrum bands and reconfiguring the spectrum with the newly allocated bands creates gaps (*holes*) in the spectrum, resulting in a fragmented spectrum. The task of the resource allocation algorithms is to fit the requests into those *holes* in the spectrum.

In [5] it was shown that this task is equivalent to the multi-knapsack problem (MKP). Two heuristic algorithms were presented together with a single scaling scheme for the bandwidth requests. Scaling down the requests is necessary when their sum exceeds the available bandwidth budget, and utilizing the scaling scheme improves the algorithm results.

In this paper, we review the MKP problem and suggest an extension to it. We then present our idea of combining *heuristic algorithms* with scaling down schemes. Some innovative schemes are suggested to maximize a defined *satisfaction measure* and at the same time support the allocation of the *assured bandwidth* for each subscriber.

In Section 2, the system's quality of service parameters is presented and performance criteria are defined. The resource management process is presented in Section 3, including the mathematical background for the MKP problem with extension, together with a problem reformulation that may suggest a *scaling down* strategy as a natural way to solve the extended MKP problem.

Section 4 focuses on the algorithms used for the resource allocation, while Section 5 shows the significance of the scal-ing schemes. Several schemes are presented, based on the quality of service parameters. Then a simplified example is presented to illustrate the whole process. The techniques used to model the system dynamics are presented in Section 6, while Section 7 compares the performances obtained in simulation. Finally, we draw our conclusions in Section 8.

2. System Parameters and Performance Criteria

2.1. The Quality of Service Parameters. Parameters of quality of service determine the capability of providing better service to network traffic. The system we study assumes some quality of service characteristics for each subscriber. The algorithms presented here take into consideration two such parameters.

(a) Priority Level. A priority factor w_k is assigned to each subscriber type, such that higher level subscribers get priority in allocation of bandwidth following user request.

(b) Assured Level. Subscribers are guaranteed (with a high probability) to receive the bandwidth they request up to some assured amount Ω_k. Any bandwidth above this is optional, up to a maximal value determined by PIR.

These two parameters are used in the definition of the *satisfaction measure* function and play an important role in algorithm design and optimization of bandwidth utilization.

2.2. Disconnections. The fact that the DVB-S2 links were designed for fixed center frequency and symbol rate would cause disconnections while changing them. Therefore, allocating a new bandwidth to a user causes a disconnection to the subscriber. To take into account this fact, we use *effective allocated bandwidth* A related to the *allocated bandwidth* A_0 according to:

$$A = \begin{cases} \beta \cdot A_0, & \text{(if disconnected)}, \\ A_0, & \text{(otherwise)}, \end{cases} \tag{1}$$

where $\beta < 1$ is a constant expressing the effect of disconnection period, and the assumed dissatisfaction it causes. In this paper, we narrow ourselves to the case where users are not moved in the frequency range if they have no request for a change.

2.3. The Satisfaction Criterion. Resource allocation is conducted according to the requests of subscribers, their priority levels, and their assured bandwidths. For a subscriber with assured bandwidth Ω, a requested bandwidth R, and an effective allocated bandwidth A, and assuming that $R \geq A$, we suggest the satisfaction measure S that may be calculated for a factor $\alpha < 1$:

$$S(R, A, \Omega) = \begin{cases} \dfrac{\Omega + \alpha \cdot (A - \Omega)}{\Omega + \alpha \cdot (R - \Omega)}, & (R > A \geq \Omega), \\ \dfrac{A}{\Omega + \alpha \cdot (R - \Omega)}, & (R > \Omega \geq A), \\ \dfrac{A}{R}, & (\Omega > R \geq A). \end{cases} \tag{2}$$

The satisfaction function ranges from 0 to 1. The maximum satisfaction value $S = 1$ is reached when the allocated bandwidth is equal to the requested bandwidth. Also, one can see that when $A > \Omega$, the contribution of any increment ΔA to the function is smaller by a factor of $\alpha < 1$ than the contribution of the same increment ΔA when $A < \Omega$.

In order to reduce the number of variables involved in this formula, we express the satisfaction function in terms of the *normalized bandwidth variables*:

$$r = \frac{R}{\Omega}, \qquad a = \frac{A}{\Omega}, \tag{3}$$

such that the satisfaction measure may be defined as:

$$S(r, a) = \begin{cases} \dfrac{1 + \alpha \cdot (a - 1)}{1 + \alpha \cdot (r - 1)}, & (r > a \geq 1), \\ \dfrac{1}{1 + \alpha \cdot (r - 1)}, & (r > 1 \geq a), \\ \dfrac{a}{r}, & (1 > r \geq a). \end{cases} \tag{4}$$

This is a concave function increasing monotonically both in the r direction and in the a direction.

For a given set of users with normalized requests and allocated bandwidths r_i, a_i and priority factors w_i, the total (weighted) satisfaction $\overline{S}$ is defined as:

$$\overline{S} = \frac{\sum_i w_i \cdot S(r_i, a_i)}{\sum_i w_i}. \tag{5}$$

The total satisfaction function gets the maximum of 1 when all requests are fulfilled. This quantitative measure may be used to compare the performance of the various algorithms and schemes.

2.4. The Requests Scale Down Criterion. While resource allocation algorithms fit the bandwidth requests into the *holes* in the spectrum, some *holes* may become *under populated* and some *over-populated*.

The first situation (*under population*) happens when the sum of requests inserted into a *hole* does not fully cover it.

The other one (*over population*) happens when the sum of requests assigned into a *hole* is greater than the capacity of the *hole*. In this situation the requests assigned to that *hole* should be scaled down to fit into the *hole* capacity.

Let $\{R_j^{(k)}\}$ be the set of requests allocated into a *hole* $H^{(k)}$, Then the total amount δ of the requests bandwidth that is scaled down is:

$$\delta = \sum_k \max\left(\left[\sum_j R_j^{(k)}\right] - H^{(k)}, 0\right). \tag{6}$$

The amount δ is the *requests scale down* factor. It measures the effectiveness of the algorithms in finding a good arrangement of the requests into the gaps (*holes*) in spectrum. A lower value indicates better algorithm performance.

3. The Resource Allocation Process

The disconnections caused while changing frequency and symbol rates pose a serious constraint in the design of optimal allocation algorithms. Therefore, links of terminals that did not require any change should stay in the same location in the spectrum, while new requests should be allocated only in the free portions (*holes*) of the spectrum.

In this part, we will show that this problem is equivalent to the multi-knapsack problem (MKP) [6] known in computational complexity theory and then draw a scheme for a computational process to cope with it.

3.1. Resource Allocation as a Complexity Theory Problem. The connection between the MKP problem and satellite communication resource allocation was already reported by Birmani [7]. However, his work deals with *power allocation* and burst scheduling problems, which are *inherently quite different* in their nature than the *bandwidth allocation* problems in our concern.

The association between resource allocation in communication and the MKP problem was already observed by Rajkumar et al. [8]. Their research was on a QoS-based Resource Allocation Model (QRAM) for satisfying multiple quality-of-service dimensions in a resource-constrained environment. Using this model, available system resources can be apportioned across multiple applications, thus maximizing the net utility that accrues to the end users of those applications. They showed that the Q-RAM problem of finding the optimal resource allocation to satisfy multiple QoS dimensions is NP hard. There is a similarity between the resources, QoS, and utility factor in their research and the bandwidth, priority level, and satisfaction function in this research. However, in our work the satisfaction function is different than their utility factor and the methods we suggest are simpler to implement.

The task of the resource allocation algorithms is to find an optimal match between a set of N requests $\{R_i \mid 0 \le i < N\}$ and a set of M holes $\{H^{(j)} \mid 0 \le j < M\}$ in a spectrum such that *all* the N_j requests are inserted into those holes:

$$\sum_{i=0}^{N_j} R_{K_j(i)} \quad \le \quad H^{(j)}, \tag{7}$$

where $\{K_j\}$ are M disjoint subsets of indexes: the size of set K_j is N_j, such that their union covers the set $\{1, \ldots, N\}$:

$$\bigcap K_j = \phi, \qquad \bigcup K_j = \{i \mid i \le N\}, \tag{8}$$

where $K_j(i)$ is the ith element in the set K_j.

The algorithm theory classifies such a problem as a variation of the *multi-knapsack problem.* Given M containers (knapsacks) with capacities $H^{(j)}$ and N items with values R_i, the task is to split the set $\{R_i\}$ into M disjoint subsets, such that the sum of item values in each subset will not exceed the given capacities $H^{(j)}$ as in (7).

Although the set of inequalities in the problem definition resembles a typical linear programming formulation, it is not a linear programming problem. Instead, it belongs to a family of *subset sum* problems known as knapsack problems.

The knapsack problem is part of the family of combinatorial NP-complete problems [9], meaning that it is computationally difficult to solve in general (except for a $O(M^N)$ brute force solution). It is difficult enough to be chosen as a trapdoor cryptographic function (i.e., the Merkle-Hellman cryptosystem).

Yet the resource allocation problem does not fit the exact definition of a knapsack problem. There is one essential difference: when (7) cannot be fully satisfied, then the target of optimization should be replaced by:

$$\sum_{i=0}^{N_j} C_i^{(j)} \cdot R_{K_j\,(i)} \le H^{(j)}, \quad \left(0 < C_i^{(j)} \le 1\right), \tag{9}$$

such that the factors $C_i^{(j)}$ are maximal.

This is a "softer" version of the original problem (7). It extends the problem adding some degree of complexity (finding the factors $C_i^{(j)}$). On the other hand, it eases the task somewhat, adding several degrees of freedom so that a solution can be found, even in situations where the original problem is practically insolvable.

So, because this is an NP-Hard problem, we shall not try to solve it fully. Instead we may design some practical heuristic algorithms and reach suboptimal solutions [10].

3.2. The Resource Management Process. The resource allocation problem is too complex to be solved at once. Instead of solving (9) directly by a single algorithm, we suggest splitting it into three subproblems:

(1) find an initial approximation to the $C_i^{(j)}$ factors;

(2) solve the main resource allocation problem;

(3) find the final values of the $C_i^{(j)}$ factors.

Following this model, the entire resource management process consists of three stages, as illustrated in Figure 1.

(1) The prescaling Step. In order for the resource allocation algorithms to operate properly, subscriber requests for bandwidth *cannot* exceed the available bandwidth budget. However, this situation does occur frequently. To cope with this problem, we suggest the *pre-scaling* process, to scale down subscriber requests according to the priority factors w_i and the assured bandwidths Ω_j.

(2) The resource allocation algorithm finds the best fit between the requests and the *holes* in the spectrum.

(3) The postscaling step is performed in one of the following situations:

(a) if the sum of the allocated requests assigned to a *hole* is bigger than the size of the *hole*, then the requests should be *downscaled*;

(b) if the sum of the allocated requests assigned to a hole is smaller than the size of the *hole*, and the requests were downscaled (by means of pre-scaling stage), then they may be *upscaled*. The allocated bandwidths are scaled up such that their new value will not exceed the original requests, and in addition their sum will not exceed the size of the hole.

In either case, the scaling is conducted according to the priority factors w_i and the assured bandwidths Ω_i in a manner very similar to the prescale stage.

The feedback in Figure 1 marks the transition to the next phase of the algorithm, when new requests are generated.

4. The Resource Allocation Algorithms

Two heuristic algorithms are presented to solve the main resource allocation problem:

(i) the Fast MKP solver;

(ii) the IBF (Insert to Best Fit) MKP solver.

Both algorithms are classified as *greedy algorithms*. At each stage they take the locally optimal choice rather than aiming for the global optimum. Such algorithms are widely used to solve heuristically *dynamic programming* problems.

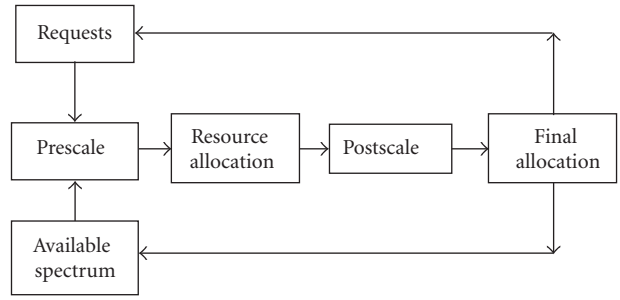

Figure 1: The resource allocation process flow chart.

4.1. The Fast MKP Algorithm. When allocating a request to a hole, we define the residue remainder after the allocation, that is, the size of the hole less the size of the request. The *Fast MKP* algorithm adopts a strategy of *largest residues first*, that is, it fits the largest requests to *holes* with the largest residue in decreasing order.

In order to perform this efficiently, we maintain a binary tree of *hole* items (sorted by the residue size) to enable a *search and insert* operation in $O(\log(M))$ time steps [11].

The algorithm consists of the following steps.

(1) Make a sorted list of requests in *decreasing order* (largest to smallest) and a binary tree of residues associated with the *hole* items.

(2) Fit the *largest request* to the biggest residue *hole* item.

(3) Update the residue value corresponding to the *hole* item and insert it back into the tree according to the residue value.

(4) Discard the request from the list of requests.

(5) Repeat steps (2)–(4) until the request list is empty.

This algorithm is efficient with respect to run time and memory consumption.

4.2. The IBF MKP Algorithm. This algorithm adopts a strategy of *best fit residue*, that is, it fits requests in decreasing order into *holes* with the residue that fits best (if no residue fits, it inserts to the largest one—). A *best fit* for any R_j over a *decreasing* series of residues $\{Z_i\}$, is any Z_i, such that $Z_{i+1} < R_j \leq Z_i$.

(1) Start with a sorted list of requests in *decreasing order* (largest to smallest) and a binary tree of residues associated with the *hole* items.

(2) Insert the largest request into the *hole* item, with the smallest residue that is larger than the request.

(3) If there is no such residue, insert the request into the *hole* with the largest residue.

(4) Update the residue value corresponding to the *hole* item and insert it back into the tree according to the residue new value.

(5) Discard the request from the list of requests.

(6) Repeat steps (2)–(5) until the request list is empty.

The purpose of stage (3) in the algorithm is to ensure that *all* the requests are inserted into the *holes*, even when *holes* become *overpopulated*. This may involve temporarily assigning negative values to some residues. This is quite legitimate, because the residue values have instrumental meaning only. The *postscaling* stage described in next section would resolve this problem.

5. The Scaling Schemes

The algorithms presented above perform poorly under a shortage of bandwidth resources. We suggest activating some scaling schemes when the sum of the requests is greater than the sum of the holes. In those cases, such schemes can share *fairly* the available bandwidth resources, taking into consideration the *quality of service* parameters in order to resolve the users' diversity issues.

For a set of requests $\{R_i\}$ and a set of *holes* $\{H^{(j)}\}$ such that $R = \sum R_i$ and $H = \sum H_i$ we introduce *scaling schemes* to take place when $R > H$.

Introducing scaling stages into the allocation process is crucial. It is analogous to *adding filters* at the entrance and outlet of an electronic system in order to adjust the signal to the dynamic range of a system. The design of scaling schemes may dramatically influence the overall performance of the allocating process just like filters that affect system's behavior.

We present first the schemes for the *downscaling* performed in the prescaling step. In correspondence with the quality of service levels [12], we consider several levels of scaling schemes in accordance with the number of parameters taken into consideration:

(i) the basic scheme, which takes no parameters;

(ii) *the priority-oriented scheme*, which takes into consideration the priority factors w_i only;

(iii) advanced schemes, taking into consideration the assured bandwidth Ω_i as well as the priority factors w_i.

5.1. The Basic Scaling Scheme. In absence of any priority criteria for sharing the bandwidth, the requests are simply scaled down proportionally to their value:

$$R_k^{(\text{new})} = \frac{H}{R} \cdot R_k. \tag{10}$$

Such a scheme corresponds to a network system that supports *best-effort* subscribers only, with no quality of service parameters.

5.2. The Priority-Oriented Scaling Scheme. This scheme takes into consideration only the *priority factors* w_i. Such a scheme is applicable for network systems that support *at least* the *differentiated service* level.

The amount of bandwidth to be reduced is the difference between the sum of requests and the sum of holes:

$$D = R - H. \tag{11}$$

This amount should be shared proportionally to the requests R_j (larger requests are reduced more than small ones) and inversely proportional to the priority factors w_j (higher priority requests are reduced less than the lower ones). So, the request R_j should be reduced by amount d_j given by: $d_j = C \cdot (R_j/w_j)$ such that $D = \sum_j d_j$ is the total reduction.

Hence: $D = C \cdot (\sum_j R_j/w_j)$ where C is some proportion factor that can be written as:

$$C = \frac{D}{\sum_k R_k/w_k} = \frac{R - H}{\sum_k R_k/w_k}. \tag{12}$$

And finally, the requests are downscaled according to:

$$R_j^{(\text{new})} = \left(1 - \frac{C}{w_j}\right) \cdot R_j. \tag{13}$$

This scheme works unconditionally (but it does not take into consideration the assured bandwidth Ω).

5.3. Advanced Schemes. High-quality network systems should take into consideration the subscribers' *assured bandwidths* Ω_i, as well as their *priority factors* w_i. The constraint imposed by the assured bandwidth condition can be expressed as:

$$R_i^{(\text{new})} \geq \min(R_i, \Omega_i). \tag{14}$$

This condition could be met only when:

$$\sum H_i \geq \sum \min(R_i, \Omega_i). \tag{15}$$

When this condition is satisfied, we can allocate bandwidth taking into account both the priority factors and the assured bandwidths. Otherwise, the *priority-oriented scaling scheme* should be used.

Our strategy is to grant all the requests below the assured bandwidth and scale down *only* the others.

The two schemes we present here have the same "prolog", and differ only in the final stage.

(0) Start with a set of indexes $I = \{1, \ldots, N\}$, where N is the number of requests.

(1) Split I into 2 subsets:

$$I^{(-)} = \{i \mid R_i \leq \Omega_i\} \text{ and}$$
$$I^{(+)} = \{i \mid R_i \leq \Omega_i\} \text{ (the complementary set).}$$

(2) Then define

$$H^{(+)} = H - \sum_{i \in I^{(+)}} R_i, \qquad \Omega^{(+)} = \sum_{i \in I^{(+)}} \Omega_i. \tag{16}$$

(3) $R_i^{(\text{new})} = R_i,\ i \in I^{(-)}$ ("grant requests less than user's assured bandwidth").

(4) Scale the requests in $I^{(+)}$ using one of the schemes described next.

(i) Difference-Oriented Scaling. Using the *difference* $(R_i - \Omega_i)$ weighted by the priority factors w_i, we get:

$$R_i^{(\text{new})} = \Omega_i + C \cdot w_i \cdot (R_i - \Omega_i), \quad i \in I^{(+)}. \tag{17}$$

In similar arguments used in developing the *priority-oriented scaling scheme*, we find:

$$C = \frac{H^{(+)} - \Omega^{(+)}}{\sum_{i \in I^{(+)}} (R_i - \Omega_i) \cdot w_i}. \tag{18}$$

(Note that the denominator $(R_i - \Omega_i) \cdot w_i$ is always positive because $R_i > \Omega_i$ for all $i \in I^{(+)}$).

(ii) Ratio-Oriented Scaling. Using the *ratio* (R_i/Ω_i) weighted by the priority factors w_i, we get:

$$R_i^{(\text{new})} = \Omega_i + C \cdot w_i \cdot \left(\frac{R_i}{\Omega_i}\right), \quad i \in I^{(+)}, \tag{19}$$

where again by similar arguments we get:

$$C = \frac{H^{(+)} - \Omega^{(+)}}{\sum_{i \in I^{(+)}} (R_i/\Omega_i) \cdot w_i}. \tag{20}$$

After the prescaling procedure, the algorithms in Section 4 are applied. This prescaling is a necessary but not sufficient condition to be able to complete the procedure. However, the postscaling can be used to complete it.

5.4. The Postscaling Schemes. The postscaling stage makes the final correction to the allocated bandwidths. It may involve more downscaling to some of the requests. This happens when the allocating algorithm inserts to a hole a set of requests whose sum exceeds the hole size and assigns a negative value to the residue associated with the hole. In this case, the downscaling is performed in essentially same way as in the prescale step, except that the calculations are made separately for each *hole* j. Hence H_j is used instead of H and $\{R_i^{(j)}\}$, and the set of requests allocated to H_j would replace R_i.

There are situations when the allocating algorithm assigns to a hole requests whose sum is less than the *hole* size. In this case, if some of the requests were over-decreased in the prescaling stage, they should be now increased back (up to the original amount) to fit the hole size.

5.5. Concluding Remarks. We have presented here some scaling schemes to support *fair* sharing of available bandwidth resources in shortage conditions.

Naturally, the *advanced schemes* are expected to perform better than the *priority oriented scheme* because they take into consideration the assured bandwidths as well as the priority factors.

However, as we have pointed out, the other scaling schemes should not be abandoned. In some exceptional situations (heavy traffic periods for instance), when condition (15) is not satisfied, the *priority oriented* scheme (or the *Basic* one) should be used instead.

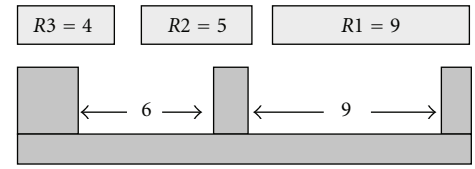

Figure 2: The initial state of *holes* and requests.

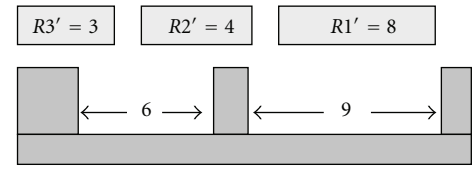

Figure 3: The state of *holes* and requests *after prescaling*.

5.6. An Illustrative Example for the Allocation Cycle. To illustrate the whole resource allocation cycle, consider the following simplified example. Consider three requests issued to the system:

$$R_1 = 9, \qquad R_2 = 5, \qquad R_3 = 4. \tag{21}$$

When there are two available "holes" in the spectrum:

$$H_1 = 9, \qquad H_2 = 6. \tag{22}$$

As shown in Figure 2.

The sum of the requests $\sum R_j = 18$ is greater than the available space in the spectrum $\sum H_i = 15$, meaning there is deviation of 3, and a scaling phase should take place to address it.

(i) Prescaling. For the sake of simplicity and in order to make it intuitive to the reader, tedious calculations involving QoS parameters considerations are dropped. Instead, we will focus on illustrating the concepts of the scaling stages.

To solve the deviation problem, the prescaling phase can reduce each request by 1. For instance,

$$R = \{9, 5, 4\} \longrightarrow \{8, 4, 3\}. \tag{23}$$

As shown in Figure 3.

This way $\sum R_j = 15$, which is exactly the amount of available space.

(ii) Allocation. The *residue* of H_i was denoted as Z_i. Initially set: $\{D\} = \{H\}$. Then:

(1) insert $R_1 = 8$ into H_1 (with $Z_1 = 9$) such that the residue is updated to $D_1 = 1$;

(2) insert $R_2 = 4$ into H_2 (with $Z_2 = 6$) which makes this residue become $Z_2 = 2$;

(3) finally, insert $R_3 = 3$ into H_2 (with $Z_2 = 2$). This makes the *residue* Z_2 receive a *negative value*: $Z_2 = -1$!

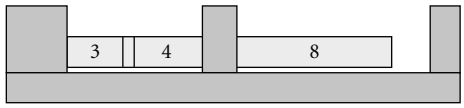

Figure 4: The requests inserted to the *holes by the allocation algorithm*. Notice the *overlapping* between R'_3 and R'_4.

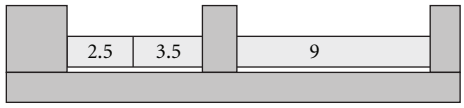

Figure 5: The final state *after the postscaling*.

This state of affairs is presented in Figure 4. The overpopulation situation in H_2 caused by the allocation algorithm is expressed by the *overlapping* R_2 and R_3 in the figure.

As stated in Section 4.2, assigning a negative value to some residues is legitimate as an intermediate stage in calculation, and the task of the *postscaling* phase is to fix this situation.

(iii) Postscaling. This phase fine tunes the bandwidths allocated. Recall that originally the request R_1 was 9, and then it was reduced to 8 and allocated to H_1, which has a capacity of 9. Obviously the original request can be granted. Thus the final allocation for R_1 should be $A_1 = 9$.

R_2 and R_3 were allocated to H_2 which has a capacity of 6, but $R_2 + R_3 = 7$, so they should be reduced again to: $A_2 = 3.5$ and $A_3 = 2.5$ such that $A_2 + A_3 = 6$ to fit the $H_2 = 6$ capacity.

So the final resource allocation yields:

$A_1 = 9$ inserted into $H_1 = 9$ and $\{A_2 = 3.5,\ A_3 = 2.5\}$ inserted into $H_2 = 6$. (See Figure 5).

6. Modeling the Resource Allocation Process

In order to simulate the system dynamics under the resource allocation process described above, we derived a model to produce the network traffic and the inputs to the process. We distinguish between the *demands* subscribers have and the *requests* they issue to the system. New demands for bandwidth emerge for each subscriber in random time steps. We assume that the demands ν_k of subscriber k are *gamma* distributed.

$$\nu_k \sim \Gamma(m, \lambda_k). \tag{24}$$

The gamma distribution [13] is a two-parameter family of continuous probability distributions. It has a shape parameter m, a scale parameter λ_k, and the average value:

$$E[\nu_k] = m \cdot \lambda_k. \tag{25}$$

The traffic models based on random variables with Gamma distribution are used in applications such as video and speech [14, 15]. Such a random variable has a probability distribution with a shape similar to a Gaussian random variable but has only positive values.

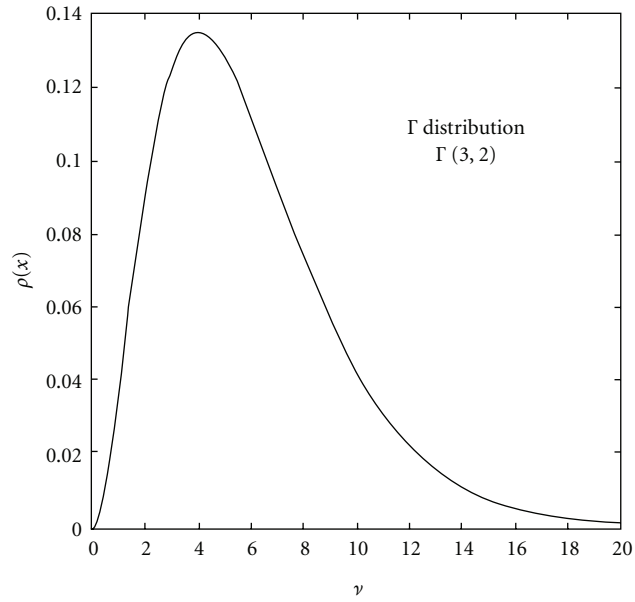

Figure 6: Gamma distribution: Γ(3,2).

The parameters are chosen such that the mean demand of subscriber k, ν_k, is proportional to Ω_k, the assured bandwidth of subscriber k, by a constant factor μ:

$$E[\nu_k] = \mu \cdot \Omega_k. \tag{26}$$

So comparing (25) and (26):

$$\lambda_k = \frac{\mu \cdot \Omega_k}{m}. \tag{27}$$

In the simulation, the values used are: $\mu = 2.0$ and $m = 3$, as shown in Figure 6.

We may assign a probability p for new demand to emerge such that in time step n we have demand $\nu_k^{(n)}$:

$$\nu_k^{(n)} = \begin{cases} \nu \sim \Gamma(m, \lambda_k), & \text{with probability } p, \\ 0, & \text{with probability } (1 - p). \end{cases} \tag{28}$$

We assume that *very often user requests are not fully granted*. In this situation, a subscriber should request the remainder $R_k - A_k$ in the next time step, added to a new demand (if one emerges). This can be modeled by assigning a variable Q_k to each subscriber k to accumulate requests R_k, subtract the allocated bandwidth A_k, and add demands ν_k:

$$Q_k^{(n+1)} = Q_k^{(n)} + R_k^{(n)} - A_k^{(n)} + \nu_k^{(n+1)}. \tag{29}$$

Stability considerations suggest limiting the request amount a subscriber k may issue. In Section 1.1, we called this limit the Peak Information Rate (PIR). We set this limit to $\kappa \cdot \Omega_k$, where $\kappa > 1$ is a constant ($\kappa = 2$ in our simulation):

$$R_k^{(n+1)} = \min\left(Q_k^{(n+1)}, \kappa \cdot \Omega_k\right). \tag{30}$$

Table 1: The simulation configuration.

Parameter		Equation	Value
Satisfaction coefficient	α	(2)	0.75
Disconnection penalty factor	β	(1)	0.9
Mean demand to assured bandwidth ratio	μ	(18)	2.0
Demand update rate	p	(21)	0.25
Threshold for new request	γ	(24)	0.40
Request bound to assured bandwidth ratio	κ	(23)	2.0

Table 2: Comparison of the "requests scale-down" between the algorithms.

	Fast-MKP	IBF-MKP
$\delta = 0$	34.0%	43.2%
$0 < \delta \leq 250$	6.0%	9.8%
$250 < \delta \leq 500$	5.4%	8.8%
$500 < \delta \leq 1000$	11.0%	14.6%
$1000 < \delta \leq 2000$	19.8%	22.6%
$2000 < \delta \leq 4000$	23.0%	35.2%
$\delta > 4000$	6.0%	10.0%

Table 3: Comparison of the *requests scale-down* for different scaling schemes.

	Priority oriented	Difference oriented	Ratio oriented
$\delta = 0$	32.0%	44.6%	48.2%
$0 < \delta \leq 25$	5.8%	8.4%	7.2%
$25 < \delta \leq 50$	8.2%	11.4%	6.4%
$50 < \delta \leq 100$	10.4%	20.6%	8.0%
$100 < \delta \leq 200$	14.4%	24.2%	12.2%
$200 < \delta \leq 400$	17.8%	20.4%	13.8%
$\delta > 400$	11.4%	15.0%	7.8%

Finally, in order to avoid the penalty of unwanted disconnections associated with new bandwidth allocations, a restriction is superimposed on issuing new requests. Only changes exceeding some threshold can justify issuing new requests. To assure this, we set up a rule: *a new request shall be issued only when:*

$$\left| R_k^{(n+1)} - A_k^{(n)} \right| \geq \gamma \cdot A_k^{(n)}. \tag{31}$$

In the simulation, $\gamma = 0.4$ is used.

7. Comparison between Algorithms

Algorithms can be ranked either by their efficiency or by the quality of the results they produce. We shall examine and compare our algorithms in the following aspects:

(i) algorithm complexity (memory and time consumption);

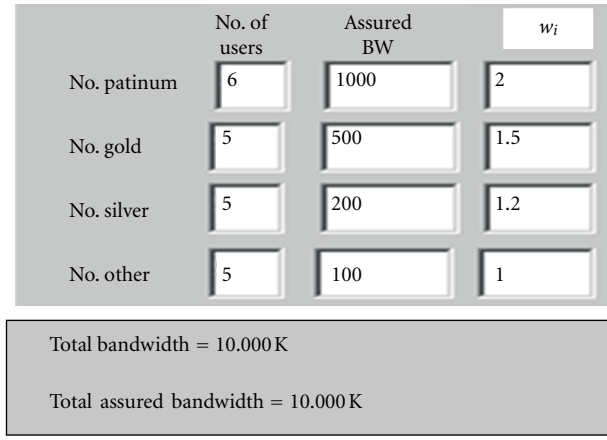

Figure 7: The system configuration in a simulation snapshot.

(ii) distribution of the satisfaction measure S;

(iii) distribution of the requests scale down δ.

We considered a typical system with 4 types of subscribers. Each type has its priority factor w_k and assured bandwidth Ω_k. The system setup is shown in a simulation snapshot presented in Figure 7.

The other simulation parameters are presented in Table 1.

7.1. The Complexity of the Algorithms. Let us consider the complexity of the algorithms (without the scaling part) for N new requests and M *holes* in the spectrum.

The time complexity of both Fast-MKP and IBF-MKP algorithm is bounded by $O(N \cdot \log(M))$ *integer operations*, as the *search and insert* operation in a binary tree is $O(\log(M))$.

The memory complexity of both algorithms is $O(N+M)$. However, we observed (experimentally) that the IBF-MKP algorithm runs slower than Fast-MKP by no more than 20% on the average. So, both algorithms are very efficient in runtime and memory consumption.

7.2. The Distributions of the Satisfaction Measure. The distributions for the satisfaction measure for the IBF-MKP and the Fast-MKP algorithms are presented by the histograms in Figures 8 and 9, obtained by simulation without the effect of the scaling schemes. For the criterion of satisfaction, the IBF-MKP algorithm achieves better results.

The effect of the scaling schemes on the satisfaction measure is shown in Figures 10, 11, and 12. The results are presented for the IBF-MKP algorithm only. The scaling schemes do improve results over those obtained without the scaling. The *ratio-oriented* scaling scheme yields better results than *Difference Oriented*, while the *Priority Oriented* scaling scheme is in third place.

7.3. The Distributions of the "Requests Scale-Down". The simulation records the amount of bandwidth scaled down in the resource allocation. Table 2 compares the distribution

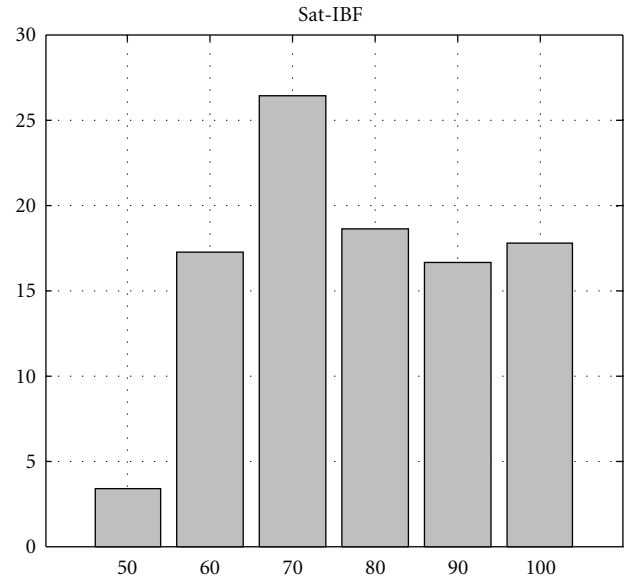

Figure 8: Satisfaction histogram IBF-MKP algorithm with *priority-oriented* scaling scheme.

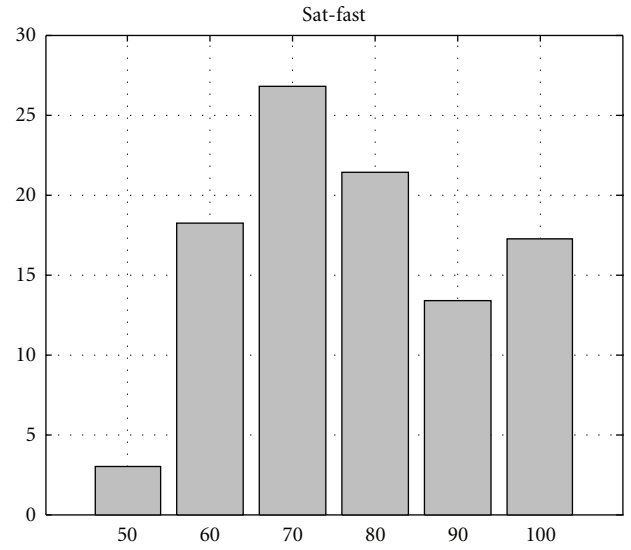

Figure 9: Satisfaction histogram for Fast-MKP Algorithm with priority-oriented scaling scheme.

of this variable for the two algorithms (without utilizing the scaling schemes).

Clearly, the IBF-MKP algorithm achieves better results for the average amount of requests scale-down in the allocation process.

The effect of scaling schemes on the *requests scale-down* factor δ is compared in Table 3, which presents results where again the algorithm is the IBF-MKP.

As expected, scaling schemes improve the *scale-down* results. The *ratio-oriented* gives the best results, while the *difference-oriented* is somewhat less effective, and the *priority-oriented* offers the least effectiveness.

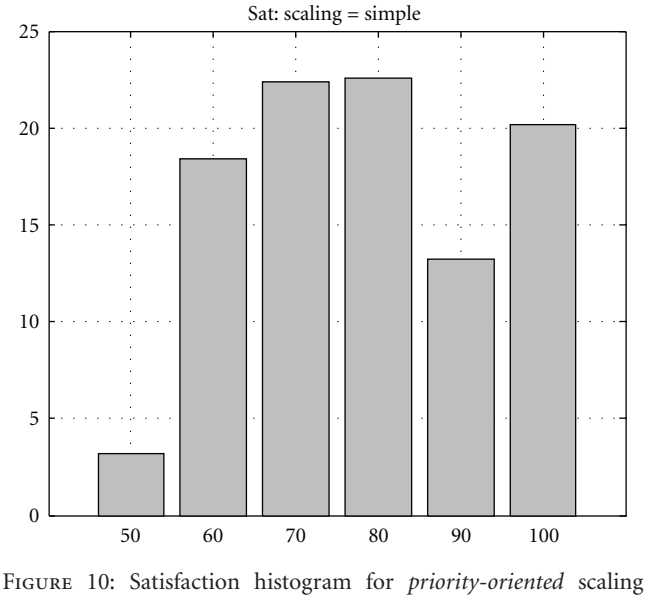

Figure 10: Satisfaction histogram for *priority-oriented* scaling scheme.

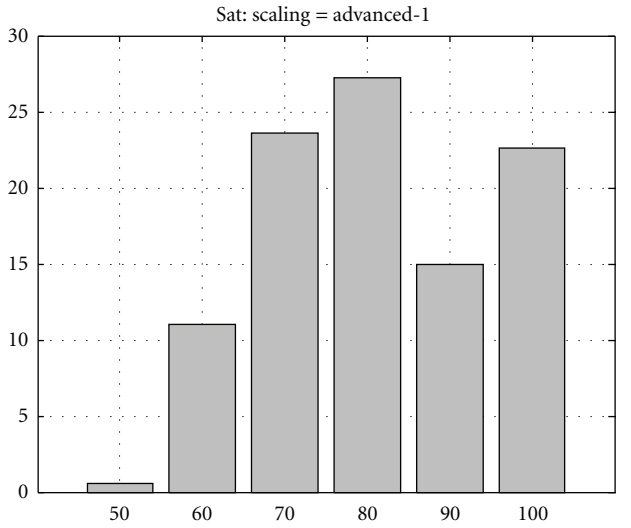

Figure 11: Satisfaction histogram for *difference-oriented* scaling scheme.

The difference is probably due to the ability to grant all requests below the assured bandwidth. This approach makes these advanced scaling schemes superior to other schemes.

8. Conclusion

In a rapidly evolving communications market with ever-growing demand for more bandwidth, a good resource management mechanism is essential for efficient digital communication of any kind. The algorithms suggested here together with the scaling schemes successfully perform this task with reasonable computational complexity, producing a high-quality resource allocation with remarkable performance.

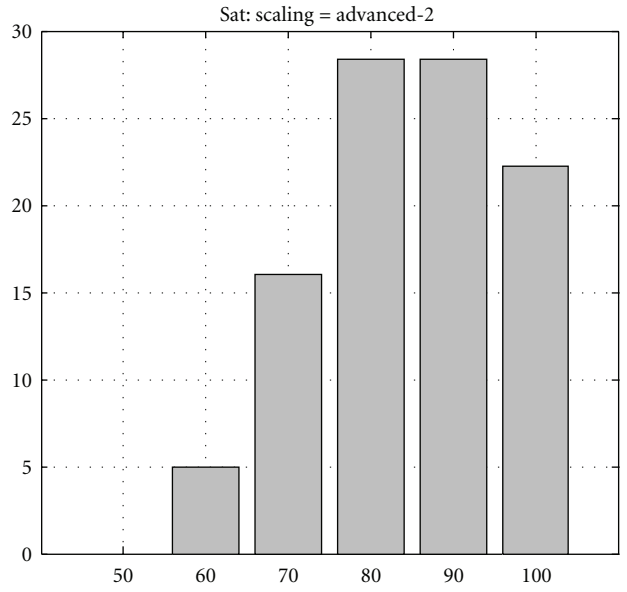

Figure 12: Satisfaction histogram for *ratio-oriented* scaling scheme.

Acknowledgments

The authors wish to acknowledge the cooperation of Ayecka Communication Systems Ltd. and the support of the MAGNETON program of the Israel Ministry of Industry, Trade and Labor.

References

[1] ETSI: EN 302 307, Digital Video Broadcasting (DVB), "Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)," *European Standard* (Telecommunications series), 2005.

[2] ETSI: EN 300 421, Digital Video Broadcasting (DVB), "Framing structure, channel coding and modulation for 11/12 GHz satellite services (DVB-S)," *European Standard* (Telecommunications series),1994.

[3] ETSI: EN 301 790, Digital Video Broadcasting (DVB), "Interaction channel for satellite distribution systems (DVB-RCS)," *European Standard* (Telecommunications series), 2005.

[4] G. Giambene, *Resource Management in Satellite Networks Optimization and Cross-Layer Design*, Springer, 2007.

[5] S. I. Wayer and A. Reichman, "Resource management in satellite communication systems—Heuristic algorithms," in *Proceedings of the IEEE 26th Convention of Electrical and Electronics Engineers in Israel (IEEEI '10)*, pp. 342–346, November 2010.

[6] D. Pisinger, "A Minimal Algorithm for the Bounded Knapsack Problem," *INFORMS Journal on Computing*, vol. 12, no. 1, pp. 75–82, 2000.

[7] V. Birmani, *Resource allocation for Ka-band broadband satellite systems [M.S. thesis]*, University of Maryland, 1999.

[8] R. Rajkumar, C. Lee, J. P. Lehoczky, and D. P. Siewiorek, "Practical solutions for QoS-based resource allocation problems," in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pp. 296–306, December 1998.

[9] S. Goddard, "P, NP, and NP-Complete," CSCE 310J lecture notes on "Data Structures & Algorithms", University of Nebraska-Lincoln, http://www.cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture10-NPcomplete.pdf.

[10] M. Hifi, M. Michrafy, and A. Sbihi, "Heuristic algorithms for the multiple-choice multidimensional knapsack problem," *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1323–1332, 2004.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 2nd edition, 2001.

[12] Quality of Service Networking, http://docwiki.cisco.com/wiki/Quality_of_Service_Networking.

[13] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, NY, USA, 2nd edition, 1984.

[14] M. Menth, A. Binzenhöfer, and S. Mühleck, "Source models for speech traffic revisited," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1042–1051, 2009.

[15] O. Rose, "Simple and efficient models for variable bit rate MPEG video traffic," *Performance Evaluation*, vol. 30, no. 1-2, pp. 69–85, 1997.