Task Allocation in Distributed Multimedia Systems based on the Host-Satellite Model

Gabriel Dermler, Ashraf Iqbal

Abstract. Multimedia applications require intermediate processing between media sources and sinks. In addition to end-user machines intermediate computers can be used for performing media processing. This possibility leads to the problem of allocating processing components on various computers. In this paper, we study this problem in the context of star-shaped application graphs which have to be allocated between given end-user machines (satellites) and a central computer (host). The problem is formulated in terms of best achievable bottleneck resource usage. Several approaches are considered including an approximate scheme and two fast-heuristics. Performance measurements show the efficiency of the considered approaches. A discussion of our approach shows important differences to solutions provided for related problems of graph partitioning and mapping.

1 Introduction

Advances in the computer and communication technology have stimulated the integration of digital audio and video with computing, leading to the development of distributed multimedia systems [StNa95]. This class of systems combines the advantages of distributed computing with the ability of processing several media in an integrated fashion. This capability enhances conventional application environments and opens the door for new innovative applications ranging from on-demand teleservices and teleconferencing systems to distributed game and virtual reality applications ([DVV94], [BLMS96]), [DGRO95], [SMK96], [HaSm96]).

The value of a distributed multimedia application to its end-users is measured in terms of the delivered Quality-of-Service ([VKBG95]), e.g. the video window size and/or rate. In order to guarantee a requested QoS, the availability of resources such as CPU, memory and communication bandwidth has to be ensured by resource reservation prior to using the application. Resource reservation protocols have been proposed for the coordination of resource allocation for the entire distributed application ([NaSm95], [Wo96], [DFR96], [RDF97]). These protocols attempt to find the best possible QoS under the constraint of limited resource availability.

Above protocols assume that the distributed processing tasks of the application, referred to here as application components, are allocated, i.e. there is no degree of freedom in selecting processing sites. This assumption is often unavoidable for source and sink components, since their location is determined by the availability of source data and the location of end-users generating respectively consuming source resp. sink data. For instance, in a teleconferencing application, source data will be generated by camera components capturing videos at the sites of the conference participants while (sink) display components will display a mixture of these videos at the same sites.

Intermediate processing components (i.e. such processing media streams between sources and sinks) have been proposed for various purposes, e.g. for conversion between different formats (e.g. different encoding schemes , large and small video, high- and low-rate video, etc.; e.g. [YMGH96], [BeSt97], [ShSe95]), media content processing (video/audio smoothing, feature detection and analysis; e.g. [SSJH96], [MAHT97]) and for mixing several media streams into a new stream (e.g. virtual reality, picture-in-picture video, audio mixing; e.g. [HaSm96], [KKKM95], [MXBX96]). Intermediate components are not subject to the location restrictions mentioned above. In fact, several projects aim at developing efficient computer platforms for supporting intermediate processing (e.g. [Sren96], [SSJH96], [AMZ95]), in order to relieve end-user computers from heavy processing load.

This paper considers the problem of finding an optimised allocation of intermediate processing components in order to maximize achievable QoS. Its basic model assumes a star shape for applications (Figure 1) and thus covers typical multimedia application scenarios such as multicasting, mixing and conferencing ([DFR96]). In this model source and sink components are attached to satellites (end-user computers) while the component at the star center is attached to a host (intermediate processing computer). Our goal in this paper is to describe algorithms deciding whether intermediate components are to be allocated on satellite or host computers, while taking into account the specific requirements arising for multimedia applications.

The paper is structured as follows. In Section 2 we first present our application model in detail and explain the relationship between QoS requests, application load requirements and resource constraints. Based on this, in Section 3 we formulate the task allocation problem for host/satellite scenarios. In Section 4 we describe both approximate and heuristic solution schemes and present measurement results. In Section 5 we discuss our results and relate our approch to earliear work in the area of task allocation for multiprocessing and distibuted systems. The paper concludes with a short summary and an outlook on future work.

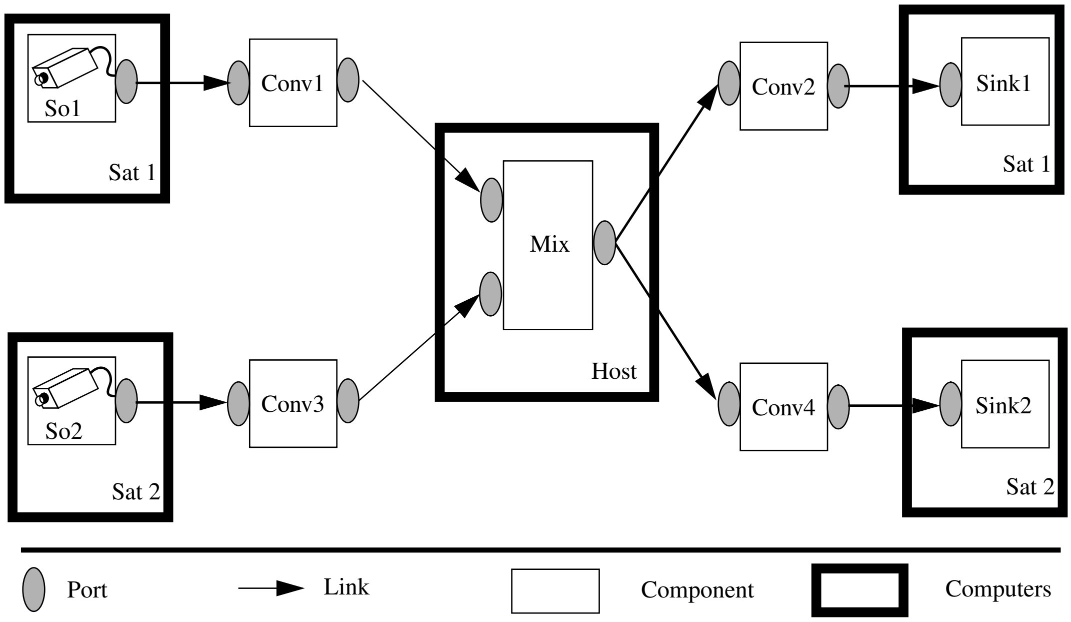

Figure 1 : A Conferencing Scenario with Mixer and Converters

2 Quality-of-Service and Resource Model

Application Abstractions

We briefly introduce our abstractions for modeling distributed multimedia applications. Similar concepts have been proposed by other groups including the one defining IMA MMS [IMA93]. Here, we describe the terminology used for our CINEMA platform [RDF97].

In CINEMA, distributed multimedia applications are represented by flow graphs, consisting of components interconnected via links. Components are processing elements encapsulating functions for capturing, storing, presenting and manipulating continuous data streams. They are associated with ports, which allow them to communicate media data. Flow graphs are configured by linking the ports of components, where a link is an abstraction of a unidirectional communication channel. Links can be of both unicast or multicast type.

As mentioned, in this paper we focus on star-shaped flowgraphs which only require certain component types for flowgraph construction. We distinguish source components, associated with one output port only, sink components, which only have one input port, and intermediate components, which receive data from input ports, perform some operation and send the result via output ports (see Figure 1). Intermediate components can be of two types. Filters (or converters) have one input port and one output port, while mixers may have several input and output ports (see [RDF97] for more complex settings).

In CINEMA, media streams are typed. For example, the media type of a stream may be "uncompressed_video", or "JPEG_encoded_video". Each media type defines a set of media parameters, which specifies the characteristics of a particular stream instance. For example, "uncompressed_video" may be associated with parameters, such as frame rate and frame size. For simplicity, in this paper we assume that only one media parameter (e.g. video frame size) is given at each port in the flowgraph.

Components may be restricted in their support for media (parameter) values by their functional design. For instance, a source component may only support one media value, though the corresponding media type allows for a set of values. In order to reflect such design limitations[1], each component port is associated with a format constraint indicating the possible media values at the port. In this paper, initially we assume that format constraints can be selected individually for sources and sinks, but are the same for all other components.[2]

Figure 2 shows the format constraints for a flowgraph. It also indicates (encircled) the best possible consistent media value selection at all component ports. For setting the media values under format constraints, two flowgraph properties have to be observed. Firstly, the media value at two interconnected ports has to be the same. The reason is simple: the media data provided at an output port has to be the same like the one consumed through the connected input port.

Secondly, media values can vary within a flowgraph. One reason is that processing involving several media streams (done in a mixer) requires different media values for incoming streams. For instance, the mixer in Figure 2 requires that the media value at port 1 is half the value of the value at port 2, since it is supposed to generate a picture-in-picture mixture of the incoming video streams. Another reason is that media conversions are required whenever the media values expected at sink components are different (see end-user QoS requests below) such that media conversions have to take place prior to them (see Figure 2). A similar argument applies to media values provided by source components.

Quality-of-Service Model

In multimedia systems, QoS implies considerations at various levels of abstraction. We refer to the architecture given in [RDF97] which distinguises an application and a resource level.

1. Note that format constraints do not depend on reource availability.
2. This is done for simplifying descriptions and will be discussed at the end of the paper.

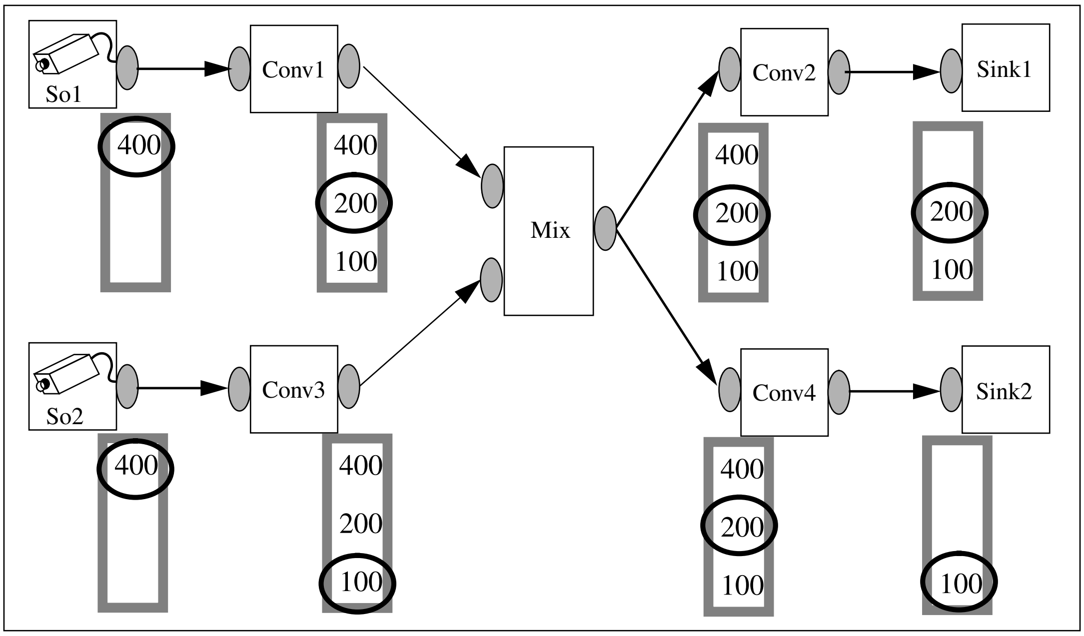

Figure 2 : Format Constraints and Media Value Selection

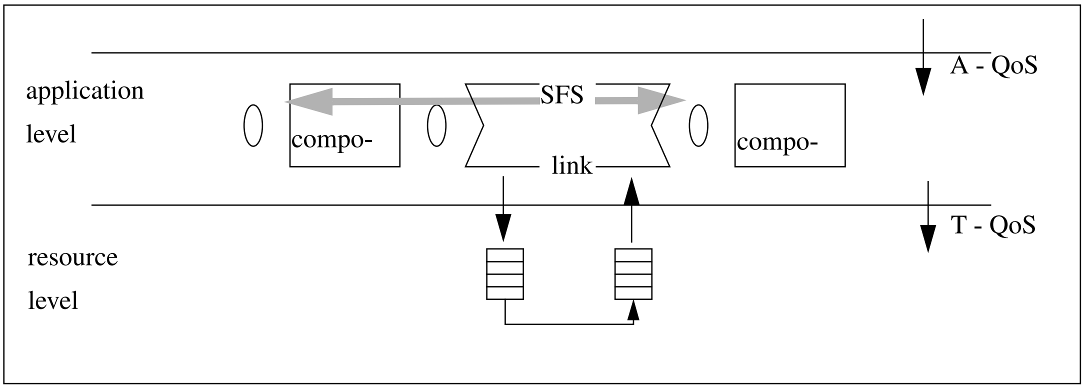

Fig. 3 : Quality-of-Service Architecture

At the application level, QoS requests (A-QoS) are issued by clients describing the QoS they expect to be delivered. A-QoS has to relate to sink components, since these are responsible for media presentation to end-users. Like in [DFR96], we require an A-QoS to be specified at the input ports of sink components, using the media parameter given there (e.g. video frame size). Also, we allow that for each sink a different request can be issued, thus allowing for “heterogeneous” receivers ([Zhan93]).

In this paper, an A-QoS request is assumed to fix the media value at a sink component’s port. Of course, a request has to select a valid value, i.e. one supported by the corresponding sink format constraint. As can be seen from Figure 2, fixing the media values at the sinks, automatically fixes the media values to be supported at all component ports in a flowgraph. Note that this fixing is done in such a way that media value reduction is done as closely as possible to the sources. Also note, that fixing the media values for converters may imply different conversion factors (e.g. coverter 1 may convert from 400 to 400, 200 or 100), depending on the requested A-QoS. This is a property of so-called variable converters introduced and motivated in [DFR96].

Resource Requirements

Components and links require resources for media data processing resp. communication. A component implies load on several resources, e.g. CPU, memory, DSP processor. Resources are shared among parts of an application and among several applications. In order to guarantee resource availability at run-time, a component has to reserve resource loads in advance. A component's load requirement is given by a tuple (LRT), e.g. (CPU-load, Mem-load, etc.). The LRT figures depend on the media values to be supported by the component at its ports.

For links, we distinguish between local and remote types. A local link employs media data passing by reference to a main memory location and hence implies no resource load. A remote link employs a transport system to communicate media data between remote locations. Its resource requirements concern both the two interconnected end-systems and the interconnecting network. We model a remote link as consiting of two objects: a send link and a receving link object. Each of them is treated like an ordinary component with respect to end-system resources, i.e. both of them imply an LRT for their corresponding end-system. An LRT for a link object contains figures for CPU and main memory as well as for bandwidth. The latter can be regarded as the required load on an I/O processor attached to the end-system.

The LRT for components depend on the selected location of a component. We do not assume any homogenity between the LRTs required by a component for a host or a satellite computer. This reflects the fact that components may be supported by totally heterogeneous hardware, and that even their functional design may be different. Similarly, we do not assume homogeineity concerning resource types and capacity available on satellites and host. Each satellite is assumed to have its set of resources with its individual capacities.

The load description we assume is based on a component related to a location (i.e. a (component, location) pair) indicating the LRT for the specific location. For a link object we require a similar description. Note that each link can become a remote link, since it may connect two components allocated on two remote compters.

Remote Communication Requirements

A remote link does not only imply data transfer between two locations. Multimedia streams are often compressed for communication, in order to reduce the high badwidth requirements (especially of video). Compression is reversed by decompression on the receiving side prior to further processing. Therefore, the send and receive link objects making up a link contain components not just for data transport but also for compression/decompression (Figure 4). The LRT description for a link object has to take both load requirements into account (see below).

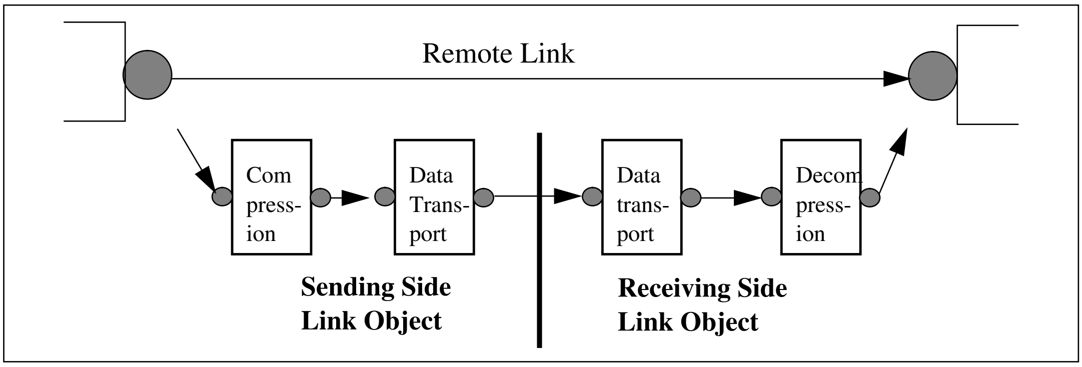

Fig. 4 : Internal Structure of a Remote Link

3 The Component Allocation Problem

Allocation Model

The problem definition is based on a star-shaped flowgraph and the mentioned host-satellite model. The flowgraph consists of component chains merging at a central component (mixer). The host-satellite model requires that sources and sinks (i.e. chain ends) are fixed on (not necessarily different) satellite computers, while the central component is fixed on the host computer. Intemediate components form component chains and each chain connects one satellite with the host.

The allocation process starts when A-QoS requests have been specified by end-users and media values at all component ports have been derived (as explained above). Given these values, the LRTs for (component, location) pairs are available as well as the LRTs for all send and receive link objects. Based on this data, for each component chain cut two cut LRTs can be computed (Figure 4), one for the load implied on the corresponding satellite (S-LRT), one for the load on the host (H-LRT). For the satellite all component LRTs are included up to the cut; in addition the load of the send link object (for the considered cut) is computed and added up. For the host, the procedure includes the LRTs beyond the cut and the LRT for the receive load object of the cut.[1]

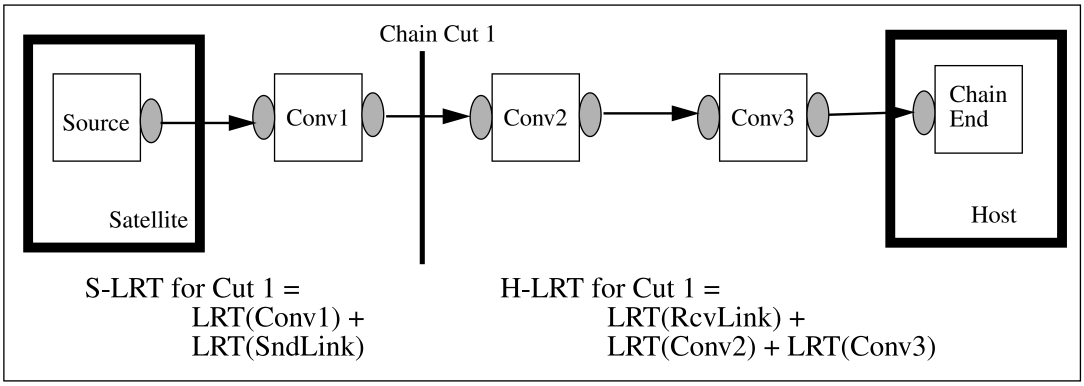

Fig. 5 : LRT Calculation for a Chain Cut

Allocation is constrained by resource availability on satellites, hosts and the interconnecting network. Each satellite and host resource (CPU, memory, DSP processor, I/O bandwidth, etc.) has a capacity limiting the sum of the LTRs implied by a specific allocation of components. The network is assumed to be able to indicate the available bandwidth capacity for the communication channel of any (satellite, host) pair.

The allocation problem is then to find the component allocation with the least bottleneck resource usage (BRU). The botlleneck resource, which may be any of the satellite or host resources, is the one which is used by an allocation in relative terms maximally (e.g. 120% of its capacity are required by the allocation and no other resource is used that much).

Minimising the BRU is linked to achieving maximal A-QoS in the following way. First, for given A-QoS requests, it finds a feasible allocation if there is one. If there is none (indicated by a BRU of more than 100%), the allocation indicates which of the satellites or the host is mostly overloaded. This gives opportunity to reduce A-QoS requests at the sinks accordingly. If the bottleneck resource is on the host, it makes sense to reduce A-QoS at some or all flow-graph sinks. If the bottleneck resource is on one of the satellites it makes sense to reduce A-QoS for that sink only. In short, the indication of a bottleneck resource indicates a feasible alo-

1. Note that the LRT of chain end-components are not included, since these are fixed in advance

cation if there is one and helps to develop A-QoS reduction policies appropriately (which, however, are out of the scope of this paper).

Abstract Problem Formulation

We are given a flowgraph consisting of a set CS of c chains. Each chain is assumed to contain (m+1) modules, thus allowing for m possible chain cuts. Each chain is characterized by a satellite identifier chainSat, indicating on which satellite it is anchored and a set of (S-LRT, H-LRT) pairs, one for each cut of the chain. For a chain cut, the S-LRT tuple indicates the LRT implied for chainSat, while H-LRT indicates the LRT implied for the host. Both are given as k-dimensional tuples, since (at most) k resources are assumed for each of the satellites and the host.

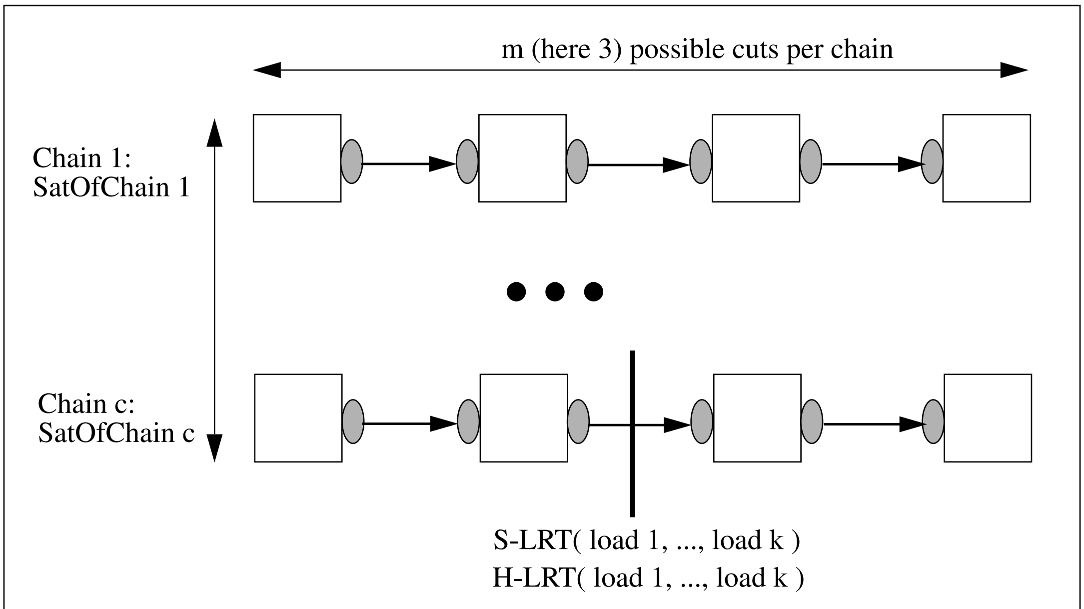

Fig. 6 : Layered Graph Problem Representation

A component allocation is given by a set of cuts (C) for the c chains. An allocation implies a requested load for each satellite and the host (S-Load resp. H-Load). For a satellite i, S-Load_i is given as the (k-vector) sum of all S-LRTs of cuts in C which belong to chains anchored on satellite i. For the host, H-Load is given as the sum of H-LTRs of all cuts in C.

Each satellite and the host is associated with a (k-dimensional) constraint vector (SC resp. HC) indicating the respective resource capacacity constraints. When an allocation is given, for each satellite i the maximum resource usage factor SMRU_i can be calculated, which is defined as the highest ratio between the requested load for a satellite resource and the corresponding resource constraint. Simlarly, for the host a HMRU can be obtained. The bottleneck resource usage BRU for an allocation is given as the maximum of all $SMRU_i$ and HMRU.

Our goal is to find an allocation with a minimal BRU.

4 Solution Approaches

The problem of partitioning chain-structured programs over a single host and multiple satellite system is discussed in [BOKH88]. A number of other researchers have also attacked this problem using faster exact algorithms or efficient approximation schemes [IQBL91], [IQBL95]. All these researchers work under the constraint that there is a single chain anchored on a satellite and the H-LRT tuple associated with the host is of a single dimension. With multiple metrics the problem is essentially to find an allocation (or a path in a graph) subject to multiple constraints. This is an NP-complete problem [GARY79], and can only be solved by pseudo-polynomial algorithms, approximation schemes [IQBL93], [IQBL96], or by using fast heuristics.

4.1 Approximate Solutions

In this section we discuss the concept of multiple weighted graphs and multiple sum paths in them. We would present an efficient algorithm for finding the multiple sum path which is useful in solving various task allocation problems in distributed multimedia systems. We first describe a pseudo-polynomial algorithm for finding a Two Sum (TS) path in a layered graph. We shall use the TS path technique to find the Multiple Sum (MS) path again in a layered graph. A layered allocation graph is shown in Fig. 7, it is the same graph as shown in Fig. 6, we have added a start and an end node, and a number of directed edges between two adjacent chains. The start node is connected to all m possible cuts in Chain 1, all m possible cuts in the last Chain c are connected to the end node. Similarly all m possible cuts in Chain i are connected to all m possible cuts in Chain i+1. It is important to note that no cut of Chain i is connected to a cut of any chain other than i+1 (where i is greater than or equal to 1 and less than or equal to c-1).

A path between the start and the end vertex passes through exactly one possible cut of each of c chains and thus corresponds to an allocation, there is in fact a one to one correspondence between an allocation and a path in the layered graph.

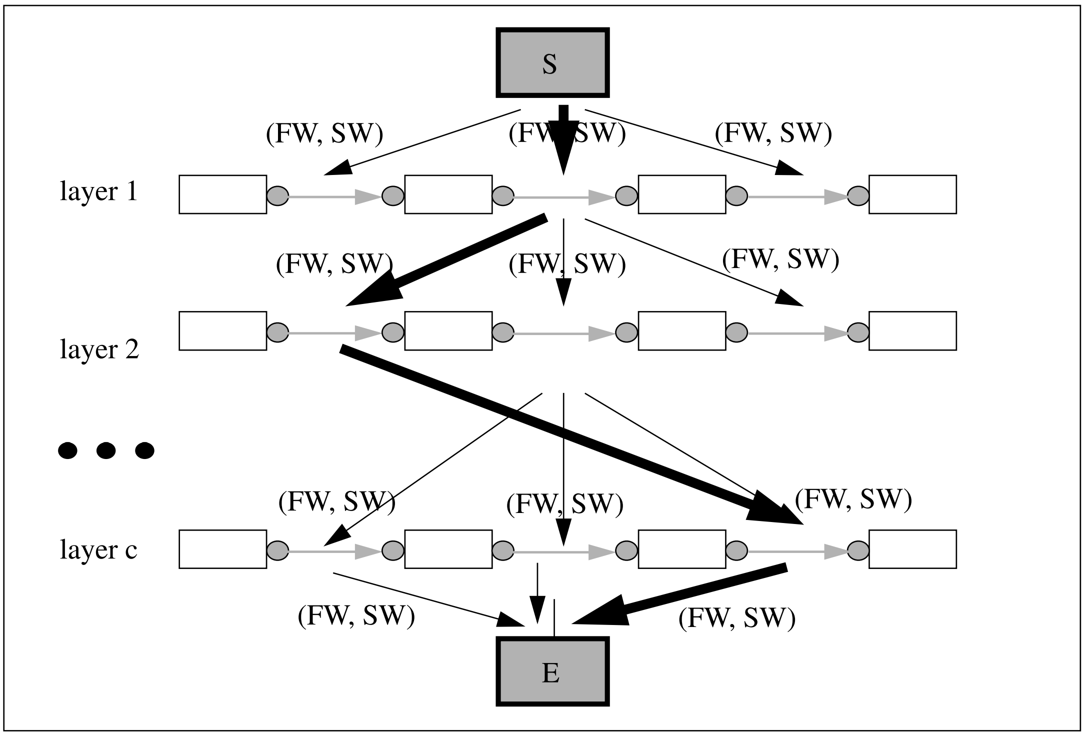

Fig. 7 : A Cut Path in the Layered Graph

4.1.1 A Multiple Sum Path in a Layered Graph

A Two Sum Path

In case of a TS path we consider a doubly (FW, SW) weighted graph in which there are only two weights associated with each edge. Thus instead of a single weight on each edge as in traditional graphs we have an ordered pair of weights on each edge. As usual, a path between the start and the end node would be composed of edges, the First Sum (FS) associated with the path would be the sum of the first elements of the ordered pairs of all edges in the path while the Second Sum (SS) would be the sum of the second element of the ordered pairs of all the edges in the path. We further assume that each weight associated with an ordered pair is an integer. The problem is to find a path between the start and the end node such that the FS and the SS associated with the path are constrained by an upper limit which is an integer, let us denote this by L. Note that the number of incoming paths at any cut belonging to layer i would be equal to m^{i-1}, that means the total number of paths between the start and the end node would be proportional to $O(m^n)$.

We try to restrict these paths at every layer by using the following procedure. Suppose there are two paths arriving at a cut belonging to any layer from the start node. We reject path P1 as compared to a path P2 provided FS(P1) as well as SS(P1) are larger than or equal to FS(P2) and SS(P2) respectively. Note that we can not reject a path P3 as compared to path P1 if FS(P3)>FS(P1) and SS(P3)<SS(P1). Thus we limit the number of paths leaving a cut to only L. That means that the number of incoming paths at any cut belonging to a layer i from all vertices belonging to layer i -1 would be at most equal to mL but the outgoing path from this very vertex would be only equal to L. A total number of O(mL) comparisons would have to be made at each cut in a layer in order to reduce the number of paths from mL to L. Thus the total number of comparisons per layer would be proportional to $O(m^2L)$. In order to find whether a path between the start and the end node exists such that FS as well as SS are both constrained by an integer limit L, would take time proportional to $O(cm^2L)$.

A Multiple Sum Path

The layered graph is now assumed to be a multiple weighted graph in which there are k weights associated with each edge, and each weight is an integer. The problem is to find a path between the start and the end node in the layered graph such that the First Sum (FS), Second Sum (SS), ..., and the Kth Sum (KS) associated with the path are constrained by an upper limit which is we also denote by L. We can use a similar procedure to limit the number of outgoing paths from a cut, thus we have to make a total number of $O(mL^{K-1})$ comparisons in order to reduce the number of mL-power K-1 incoming paths at each cut to L^{K-1} outgoing paths from that very cut. The total comparisons made per layer would thus be at the most equal to m^2L^{K-1}. The total complexity of this pseudo-plynomial algorithm would be limited by $O(cm^2L^K)$.

The Special Case of Uniform Edge Weights

We consider here the special case where all edges, incident on a cut, in between any two adjacent layers in the graph have the same K weights associated with them. We can use a similar procedure to limit the number of outgoing paths from a cut, and we have to make a total number of mL^{K-1} comparisons in order to reduce the number of mL^{K-1} incoming paths at each cut to L^{K-1} outgoing paths from that very cut. The total steps made per layer would, however, be limited to mL^{K-1}. The total complexity of the pseudo-polynomial algorithm would thus be $O(cm\ L^K)$ under the condition of uniform weights on edges in between two adjacent layers in the graph. Note that this case corresponds to our component allocation problem, since each edge leading to the same cut carries the same load tuples which are implied by the cut.

4.1.2 Task Allocation for One Chain per Satellite

We would use the Multiple Sum Path Algorithm to find a task allocation with minimal BRU. The layered allocation graph is shown in Fig. 8, note that there is only one chain anchored on a satellite. All m incoming edges at any cut ct in chain i from all possible cuts in chain i-1 are weighted with two k-dimensional tuples, i.e., the S-LRT, and the H-LRT pair associated with the cut ct in chain i. All incoming edges at the end node have zero weights associated with them.

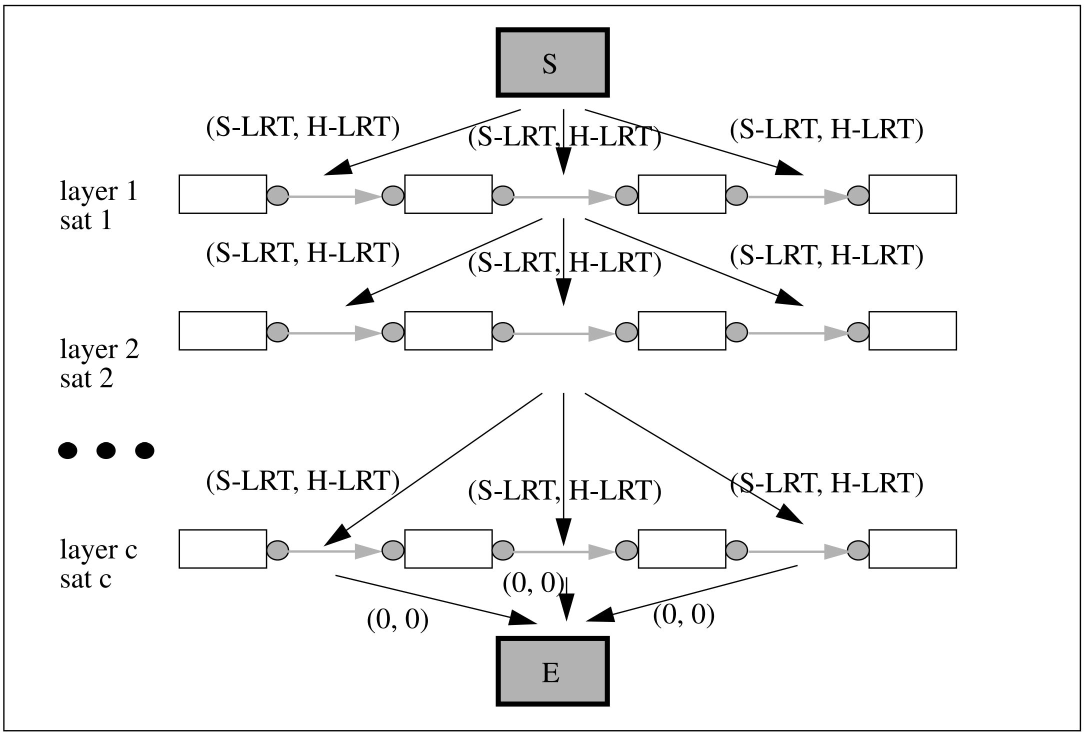

Fig. 8 : One Chain per Satellite

In order to find a task allocation with minimal BRU we use a function Probe(L) to find if an allocation exits with BRU bounded by L. If Probe(L) returns true meaning that such an allocation really exists then we lower the limit L otherwise we increase it. We thus make an efficient search for an allocation with minimal BRU. The working of the function Probe(L) is described below.

The Function Probe(L)

We remove each edge in the layered allocation graph for which the maximum resource usage factor $SMRU_i$ is higher than L for each satellite i. Note that in the special case of one chain per satellite each edge (except the one's terminating at the end node), in the allocation graph, represents an allocation for a chain i which is the only chain anchored on satellite i.

In the remaining layered allocation graph we ignore the k-dimensional S-LRT tuple and only consider the k-dimensional H-LRT tuple associated with each edge. Using the Multiple Sum Path algorithm we try to find if HMRU is less than or equal to L. If HMRU is bounded by L then the function Probe(L) returns true (as well as the possible allocation), otherwise it returns false. The complexity of the function Probe(L) is O(cm L^k). We can thus find an allocation with minimal BRU in steps proportional to O(cm [L^k]*log L).

4.1.3 Task Allocation for Multiple Chains per Satellite

We would use a similar Probe(L) function and the Multiple Sum Path algorithm to find a possible allocation where the BRU is bounded by L. We again use an efficient search technique to find the allocation with minimal BRU.

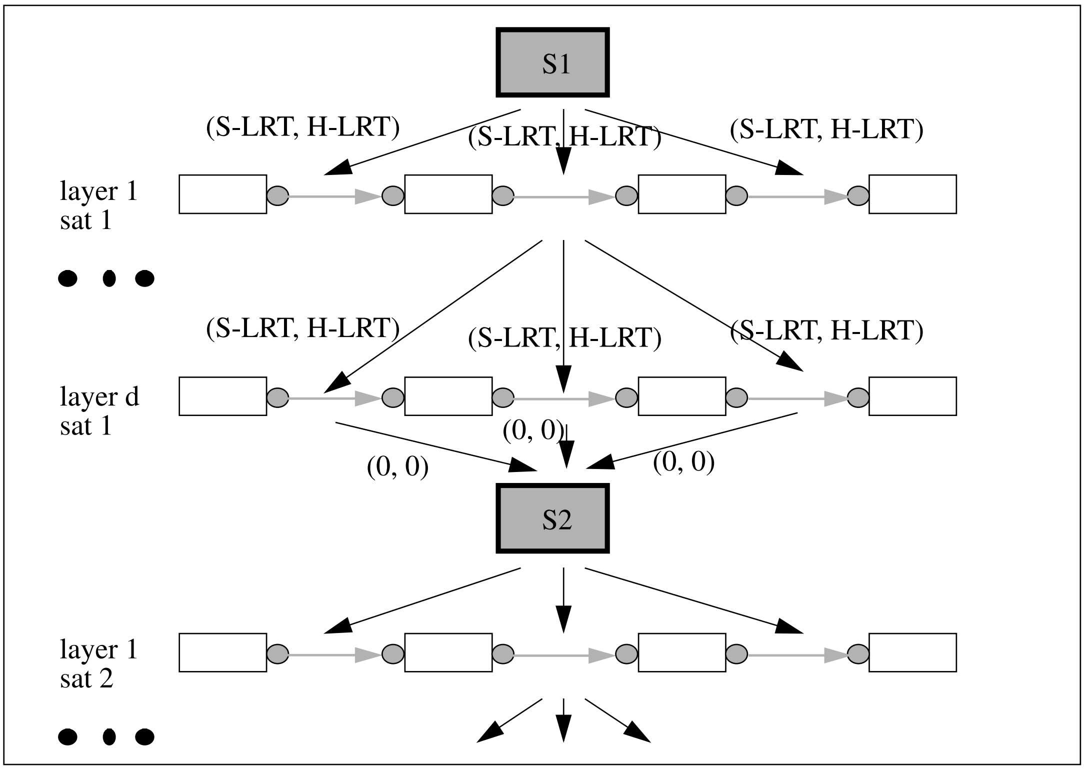

Fig. 9 : Chain Grouping per Satellite

We group all chains anchored on satellite 1, we shall first consider only these chains. The layered allocation graph is shown in Fig. 9, where all the d chains anchored on satellite 1 are shown. All m incoming edges at any cut ct in chain i are weighted with two k-dimensional tuples, one such tuple is S-LRT, and the other is H-LRT, both these tuples corresponds to cut ct in chain i (note that all d chains are anchored on satellite 1). Thus each incoming edge at any cut in the allocation graph is weighted with 2k weights. We can now design a suitable Probe(L) based on the Multiple Sum Path algorithm to find if a path between the start node and the end node exists in the layered graph for which both $SMRU_1$ and HMRU are bounded by L.

The total number of distinct paths leaving each cut in the last chain anchored on satellite 1 would be proportional to L^{2k}. Thus there would be at most mL^{2k} paths arriving at the end node. Corresponding to each path there would be an associated H-Load which is equal to the k-vector sum of all H-LTR's correspoding to all cuts of all chains anchored on Satellite 1. The k-dimensional H-Load can have at the most $O(L^k)$ distinct possibilities, thus mL-power 2k allocations (or paths) can be reduced into $O(L^k)$ distinct paths by rejecting unnecessary paths or paths that would lead to an overall degraded performance. The total number of comparisons made so far would be proportional to $O(dmL^{2k})$.

This process is repeated for each satellite and it will take $dmsL^{2k}$ steps to find if an allocation exists where the BRU is bounded by L. The total complexity of finding an allocation with minimal BRU would be $O(\ dms[L^{2k}]*\log L\ )$.

4.2 Fast Heuristics

The previously described algorithms imply an overhead which is increasing exponentially with increasing number of constraints. As measurements show (see below), time overhead is at most tolerable for the case of 1 constraint. In order to cope with situations where many constraints are considered, we have developed and analyzed several heuristics. The most performant one, referred to as the SQ(ueeze) algorithm, is presented here along with performance measurement results. SQ is compared with an algorithm proposed earlier, MMKP (m-dimensional multiple-choice knapsack algorithm []), which was designed for a different optimisation model. However, this model, with small modifications, is applicable to the host-satellite allocation model as well. We postpone a review of other heuristic approaches to Section xxx.

The SQ Algorithm

The basic idea of SQ is to calculate intial chain cuts using a greedy-type approach, and then to gradually improve the chain cuts by excluding “bad” alternatives, until, for each chain, only one alternative is left (see Figure below).

The intial cut set is computed considering the chains one after the other in a fixed, though arbitrary sequence. For each chain, the cut is computed which yields the lowest BRU. The procedure is continued until all chains have been cut. The second part of SQ is based on several iterations. During each iteration, the chains are considered in the given fixed order. For each chain, both the best and the worst cut (in terms of BRU) are derived. For this, each chain cut is considered as a possible replacement for the cut of the previous iteration (or the initialization part). The best cut is then used as replacement, The worst cut is disabled, i.e. is not considered during following iterations.

The computation complexity of SQ is determined by the iterative part. During each iteration, one cut is disabled for each chain. Since there are m cuts, there is a total of (m-1) iterations. Per iteration, at most c*m cuts are considered, where each cut consideration involves k additions and comparisons. Hence, the overall complexity is of order $o(k * c * m^2)$.

The MMKP Algorithm for the Host-Satellite Problem

The optimzation model of the MMKP algorithm assumes c groups (here chains), where from each group one of m elements (here cuts) has to be selected. All elements make use of the resources of one knapsack with r resources. Each element implies an (r-dimensional) load vector for the knapsack resources. Knapsack resources are limited by finite capacities. The goal of MMKP is to find the element selection (one from each group) which is within the given resource constraints and which minimizes overall cost (given as the sum of selected element costs). The reader is referred to [Mose96] for a more detailed description.

The host-satellite allocation model fits into the MMKP model, if all satellites and the host are considered as one super-system with a total of r = k*(noSatellites+1) constraints. Each chain forms an element group, where elements correspond to the cuts of a chain. However, there are two notable differences.

Firstly, the MMKP model seeks a solution withtin the resource constraints, not necessarily a solution with minimal BRU. However, the MMKP algorithm iteratively calculates and reduces the load for the bottleneck resource. It stops, when the (relative) BRU is below 100%. It is straight-forward to see from [Mose96], that if this stopping criterion is removed, the MMKP algorithm seeks a solution with minimal BRU. Secondly, the MMKP model seeks a solution optimizing for cost, in this respect it is more general than the SQ model. Including cost into

```
Input:  c chains      X_i,       1≤ i ≤ c each with
          m cuts      X_iC_j     1≤ j ≤ m each with a 2*k-dimensional load tuple
                      X_iC_jL^1
          1 satId     X_iS       1≤ i ≤ c
        s satellites  S_p        1≤ p ≤ s
        1 host        H

Output:     Set of Cuts { 1 ≤ i ≤ c | X_iC }
            Bottleneck resource usage BRU
            Bottleneck resource id BR

Step 0:     satellite and host load initialisation
        for all S_p, set used load(-vector) to 0
        for H, set used load(-vector) to 0

Step 1:     calculate inital cuts for chains
        for( i = 1; i <= c; i++) {
            for( j = 1; j <= m; j++) {
                vector-add load tuple of chain i at cut j to used load of X_iS and H
                get the currentBRU
                if currentBRU lower than bestBRU so far,
                    store j as best cut so far, update bestBRU
                remove load of chain i from X_iS and H
            }
            vector-add load of chain i at the best cut found to used load of X_iS and H
        }

Step 2:     improve iteratively the selected cuts
        for( iter = 1; iter <= m-1; iter++ ) {
            for( i = 1; i <= c; i++ ) {
                remove load of chain i from X_iS and H as implied by best cut so far
                find best cut and worst cuts for chain i as in Step 0
            }
            mark worst cut as disabled
            vector-add load of chain i at the best cut found to used load of X_iS and H
        }

Step 3:     compile result
        for( i = 1; i <= c; i++)
            X_iC = best cut after last iteration of Step 2
        BRU = bottleneck resource usage after last iteration of Step 2
```

1. Load elements are assumed to be normalized to values between 0 and 1 with respect to the corresponding capacity constraints.

Fig. 10 : The SQ algorithm

our model is easy, for instance each cut may be associated with the communication cost of the cut.

The main expected drawback of MMKP was its relatively high complexity o($k^* c^2 *NC^2$), when applied to the host-satellite allocation problem. However, measurement results (see below) show that the SQ algorithm is superior not only with respect to time complexity but also concerning the quality of achieved results.

4.3 Measurement Results

We implemented the three approaches introduced above and evaluated them for various problem sizes. In addition we considered a fourth approach (RCut) randomly selecting for each chain a cut.

Since the approximative scheme yields optimal solutions, we used it as an absolute comparison base. However, this scheme proved to imply too much overhead for problems including more than one constraint per computer. For these cases, the results were compared in relative terms.

We considered various problem sizes, varying both the number of chains and the number of cuts per chain, while keeping the number of chains per satellite fixed (to 2 for the measurements below). We selected for each cut (linearly distributed) random load figures between a minimum and a maximum value [minload = 100, maxload = 300]. Similarly, for all satellite capacities we assumed random values between [mincap = 800, maxcap = 2400].

We balanced the host capacities accordingly, i.e. its constraints were randomly set between [mincap*NoSatellites, maxcap*NoSatellites], given that a host has to support all chains coming from all satellites. We selected this setting since it allows to consider settings where the botlleneck resource varies its location equally between satellites and the host. For each problem setting, we performed a run of 200 measurements and computed for each approach the averaged relative BRU with respect to the optimum result (of the approximative scheme).

.....noSats noCuts	2	4	8	
2: Approx RCut SQ MMKP	100 134 101 102	100 146 100 102	100 153 100 102	
4: Approx RCut SQ MMKP	100 153 102 107	100 182 102 110	100 221 101 113	
8: Approx RCut SQ MMKP	100 164 104 114	100 189 103 120	100 236 101 128	

Table 1. Relative BRU (1 constraint per computer)

.....noSats noCuts	2	4	8	16
2: Approx RCut SQ MMKP	261 0 0 0	594 0 0 0	3019 0 1 3	4118 0 1 3
4: Approx RCut SQ MMKP	333 0 0 0	721 0 0 2	1141 0 1 5	5449 0 2 15
8: Approx RCut SQ MMKP	515 0 0 2	950 0 1 7	1700 0 3 20	7932 0 5 56

Table 2. Time Overhead in msec (1 constraint per computer)

Table 1 shows the relative BRU (in %) obtained for the RandomCut, SQ and the MMKP algorithms for various problem sizes, while Table 2 shows worst-case time overheads. It is obvious that the approximative scheme is too slow for a computation which is required to be done in real-time. For more than one computer constraint, the overhead is in the range of, at least, minutes. On the other hand, the tables show that SQ performs very well both in terms of achieved BRU and time overhead. The achieved BRU deviates only 2% from the optimum, while time overhead stays within the range of a few msec. SQ always outperforms the MMKP approach.

More measurements have been done, confirming qualitatively above results. For instance, in cases when satellites had abundant resources (i.e. the host was the bottleneck), MMKP results deteriorated, while SQ results did not. Also, the results were comparable for higher number of constraints. For instance, Table 3 shows the BRU achieved for the case of three constraints again indicating the superiority of SQ over the other approaches.[1]

.

.....noSats noCuts	2	4	8	16
2: Enum **RCut** **SQ** **MMKP**	**100** **145** **100** **121**	**100** **142** **100** **109**	**X** **146** **100** **105**	**X** **146** **100** **105**
4: Enum **RCut** **SQ** **MMKP**	**100** **191** **101** **111**	**X** **184** **100** **114**	**X** **183** **100** **120**	**X** **183** **100** **124**
8: Enum **RCut** **SQ** **MMKP**	**100** **202** **103** **117**	**X** **197** **100** **122**	**X** **200** **100** **133**	**X** **195** **100** **139**

Table 3. Relative BRU (3 constraints per computer)

4.4 Related Work

The multimedia component allocation problem, as presented in this paper, bears similarity to the problem of partitioning and mapping parallel programs onto multiprocessor architectures

1. For small problem sizes the table contains results obtained by enumeration. In these cases the absolute optimum could be considered as reference base.

or distributed systems (e.g. [NoTh93] and other references below). The general representation of this problem is that of a static program graph consisting of modules interconnected via links, where modules serve to perform various processing tasks and links indicate required intermodule communication. Both modules and links are labelled with figures indicating load or cost properties.

A multiprocessor or distributed system graph is used to describe possible computation sites for program modules. Nodes and links of this graph are labelled with resource constraints limiting node processing or communication capabilities. The goal is to find a program partitioning and mapping on the system graph without violating any of the given constraints and minimizing overall cost. This problem formulation seems similar to the one considered in this paper. However, important differences and mismatches exist.

First, most suggested approaches consider specific resource load types (and constraints) only. In fact, a large group does not consider load constraints at all, but aims solely at finding a graph mapping with minimum cost ([KLZ97], [LeSh97], [BiEl95], [Bill94], [LLK92], [Fern89], [Lo88], [Tows86], [Ston77]). A second group considers for each program module a (CPU) processing load figure and aims at a mapping implying a balanced load on each processor ([OlMa95], [IqBo95], [HaLi92], [NiHa91], [Bokh88]). Further load types (e.g. memory, communication loads) are not considered. Other approaches extend the model by considering cost in addition ([SBAM96]), [YaSa93], [BNA92], [KiPa90], [Efe82]).

A few approaches have been described taking into account many resource constraints given in a particular mapping problem context. In [YWPS95] the mapping problem is considered taking into account (among others) CPU load, memory load and communication bandwidth. However, the approach employs branch-and-bound and simulated annealing techniques, both of which have exponential worst-case complexity. Similarly, in [MLT82] an approach based on branch-and-bound techniques is described. The approach presented in ([WoMo93]) has polynomial time complexity and takes both a processing and a shared bus communication load into account. However, it does not consider more endsystem load types (e.g. memory, network I/O).

A second important aspect concerns the cut-oriented description of the multimedia allocation problem. Here, each cut is associated with load figures, not each module. This description allows to take into account for each cut both the processing load(-tuple) implied by allocated components and the load(-tuple) implied by the remote communication at a specific chain cut (see Section 3). This allows to take into account the endsystem load required for remote communication which can be very significant, given that compression/decompression may be provided. Using cut descriptions for solving the problem is accurate. Almost all existing approaches consider endsystem load as being not influenced by communication requirements (i.e. communication implies load within the network only). The approach in [WoMo93] does a first step in this direction. However, it simplifies the relationship by interpreting required communicaton load twofold: as network bandwidth requirement and as a processing load requirement which has to be added to a processor's workload. In general (and especially when compression/ decompression is done), this simplification does not hold.

Finally, existing approaches imply a higher computation complexity than our approach. One reason is that many of them were designed for other, in some respect more general, mapping problems (e.g. for instance for arbitrary program graph shapes). Another reason is the method used. The approach in [SBAM96] was specifically designed to achieve a low computational complexity, which is also lower than our protocol's. However, when applied to the host/satellite problem, the method is basically equivalent to the initialisation part of the SQ algorithm, which (as a stand-alone solution) delivers significantly worse solutions.[1] This also shows that solutions exhibiting good average behaviour in general settings (as indicated in [SBAM96]), can prove inferior to specialized solutions when applied to special cases.

In summary, none of the existing approaches can be considered as fully applicable to the mapping problem considered in this paper. In constrast, our approach directly targets the requirements of the multimedia host-satellite allocation problem including independence of the number of load types considered, cut-oriented load considerations and computation efficiency.

5 Summary and Future Work

Many multimedia applications require intermediate processing between media sources and sinks. In order to support such applications computers are being designed relieving end-user terminals from heavy processing load. In this paper, the problem of allocating intermediate processing components on such computers was studied. The problem was formulated for common structures occuring in multimedia application settings, namely star-shaped application graphs and host-satellite distributed computer settings.

1. Also, the approach does not consider all required load types and is also not cut-oriented in the sense decribed above.

The allocation problem was formulated in terms of chain cuts taking into account both processing and communication load requirements. Several approaches for determining optimized cuts have been considered including an approximation scheme and several fast heuristics. All of these are capable to take into account as many load constraints as required. Performance measurements showed the efficiency of the SQ algorithm delivering chain cut solutions very close to the optimum, while requiring less time than other approaches, both in comparison with an optimal scheme which was used for benchmarking and an approach designed for the related multiple-choice multiple-knapsack problem.

Though designed for the described host-satellite problem, the SQ algorithm is applicable to more general cases as well. It is easy to see that SQ is applicable without modifications to the case of several star-shaped graphs using the same host. In addition, with minor modifications, SQ can be used for sets of star-shaped and linear graphs traversing the same host. Currently, we investige settings where sets of graphs are allocated using several of a set of potential hosts.

Along another line, we look at an integrated way to couple QoS calculations and determining component distribution. Finally, we design a protocol for gathering required QoS and allocation information and performing resource reservation in accordance to calculated QoS and mapping decisions.

References

[AMZ95] E. Amir, S. McCanne, H. Zhang. An Application Video Level Gateway. 3rd ACM Int. Multimedia Conference, San Francisco, Nov. 1995.

[BeSt97] R. Bertram, R. Steinmetz. Scalability of Audio Quality for Networked Multimedia Environments. IEEE Int. Conference on Multimedia Computing and Systems, Ottawa, June 1997.

[BiEl95] A. Billionnet, S. Elloumi. An Algorithm for Finding the k-best Allocations of a Tree Structured Program. Journal of Parallel and Distributed Computing, 26, pp/ 225-232, 1995.

[Bill94] A. Billionnet. Allocating Tree Structured Programs in a Distributed System with Uniform Communication Costs. IEEE Trans. on Parallel and Distributed Systems, Vol. 5, No. 4, Apr 1994.

[BLMS96] H. Bryhni, H. Lovett, E. Maartmann-Moe, D. Solvoll, T. Sorenson. On-Demand Regional Television over the Internet. 4th ACM Int. Multimedia Conference, Boston, Nov. 1996.

[Bokh88] Partitioning Problems in Parallel, Pipelined, and Distributed Computing. IEEE Transactions on Computers, Vol. 37, No. 1, pp 48-57, Jan 1988.

[BNG92] N.S. Bowen, C.N. Nikolau, A. Ghafoor. On the Assignment Problem of Arbitrary

Process Systems to Heterogeneous Distributed Computer Systems. IEEE Trans. on Computers, Vol. 41, No. 3, pp. 257-273, 1992.

[DFR96] G. Dermler, W. Fiederer, I. Barth, K. Rothermel. A Negotiation and Resource Reservation Protocol (NRP) for Distributed Multimedia Applications. IEEE Int. Conference on Multimedia Computing and Systems, Hroshima, Japan, June 1996.

[DFR97] G. Dermler, W. Fiederer, K. Rothermel. QoS Negotiation and Resource Reservation for Distributed Multimedia Applications. IEEE Int. Conference on Multimedia Computing and Systems, Ottawa, June 1997.

[DGOR94] G. Dermler, T. Gutekunst, F. Ruge, E. Ostrowski. Sharing Audio/Video Applications among Heterogeneous Platforms. 5th IEEE COMSOC Int. Workshop on Multimedia Communications, Kyoto, Japan, 1994.

[DVV94] D. Deloddere, W. Verbiest, H. Verhille. Interactive Video on Demand. IEEE Communications, Vol. 32, No. 5, pp. 82-88, May 1994.

[Efe82] K. Efe. Heuristic Models of Task Assignment Scheduling in Distributed Systems. IEEE Computer, pp. 50-56, June 1982.

[Fern89] D. Fernandez-Baca. Allocating Modules to Processors in a Distributed System. IEEE Trans. on Software Engineering, Vol. 15, No. 11, Nov 1989.

[GaJo79] M. R. Garey, D.S. Johnson. Computers and Intractability- A Guide to the Theory of NP-Completeness, Freeman, California, USA, 1979.

[IMA93] HP Company, IBM Corp., SunSoft Inc. Multimedia System Services, Version 1.0, available via ftp from ibminet.awdpa.ibm.com.

[HaLi92] P. Hansen, K.W. Lih. Improved Algorithms for Partitioning Problems in Parallel, Pipelined and Distributed Computing. IEEE Trans. on Computers, Vol. 41, No. 6, pp. 769-771, June 1992.

[HaSm96] J. Han, B. Smith. CU-SeeME VR Immersive Desktop Teleconferencing. 4th ACM Int. Multimedia Conference, Boston, Nov. 1996.

[Iqbl91] M.A. Iqbal. Approximate Algorithms for Partitioning Problems, International Journal of Parallel Programming, October 1991.

[IqSh93] M.A. Iqbal, M.E. Shaaban. Heterogeneous Partitioning of Chain Structured Image Processing Tasks, IEEE Workshop on Computer Architectures for Machine Perception (CAMP'93), December 1993.

[IqBo95] M.A. Iqbal, S.H. Bokhari. Efficient Algorithms for a Class of Partitioning Problems, IEEE Trans. Parallel & Distributed Systems, vol.6, no. 2, February 1995.

[KiPa90] C.H. Lee, M. Kim, C.I. Park. An Efficient K-Way Graph Partitioning Algorithm for Task Allocation in Parallel Computing Systems. IEEE Computer, pp. 748-751, 1990.

[KKKM95]P.H. Kelly, A. Katkere, D.Y. Kuramura, S. Moezzi, S. Chtterjee, R. Jain. An architecture for multiple perspective interactive video. 3rd ACM Int. Multimedia Conference, Boston, San Francisco, Nov 1995.

[LeSh97] C.H. Lee, K.G. Shin. Optimal Task Assignment in Homogeneous Networks. IEEE Trans. on Parallel and Distributed Systems, Vol. 8, No. 2, Feb 1997.

[Lo88] V.M. Lo. Heuristic Algorithms for Task Assignment in Distributed Systems. IEEE Trans. on Computers, Vol. 37, No. 11, pp. 1384-1397, Nov 1988.

[KLZ97] Y. Kopidakis, M. Lamari, V. Zissimopoulos. On the Task Assignment Problem: Two New Efficient Heuristic Algorithms. Journal of Parallel and Distributed Computing, Vol. 42, pp. 21-29, 1997.

[LLK92] C.H. Lee, D. Lee, M. Kim. Optimal Task Assignment in Linear Array Networks. IEEE Trans. on Computers, Vol. 41, No. 7, pp. 877-880, July 1992.

[MAHT97] K. Minami, A. Akutsu, H. Hamada, Y. Tonomura. Enhanced Video Handling based on Audio Analysis. IEEE Int. Conference on Multimedia Computing and Systems, Ottawa, June 1997.

[MLT82] P.R. Ma, E.Y.S. Lee, M. Tsuchiya. A Task Allocation Model for Distributed Computing Systems. IEEE Trans. on Computers, Vol. 31, No. 1, Jan 1982.

[Mose96] M. Moser. Declarative Scheduling for Optimally Graceful QoS Degradation. IEEE Int. Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996.

[MXBX96] Z. Maojun, H. Xiaofeng, Y. Bing, K. Xishu. Fats Algorithms for Compositing Multi-Way Compressed Video. IEEE Int. Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996.

[NaSm95] Klara Nahrstedt, Jonathan Smith. The QoS Broker. IEEE Multimedia, Vol. 2, No. 1, pp. 53-67, Spring 1995.

[NiHa91] D.M. Nicol, D.R. O'Hallaron. Improved Algorithms for Mapping Pipelined and Parallel Computations. IEEE Trans. on Computers, Vol. 40, No. 3, pp. 295-305, Mar 1991.

[NoTh93] M.G. Norman, P. Thanisch. Models of Machines and Computation for Mapping in Multicomputers. ACM Computing Surveys, Vol. 25, No. 3, Sep. 1993.

[OlMa95] B. Olstad, F. Manne. Efficient Partitioning of Sequences. IEEE Trans. on Computers, Vol. 44, No. 11, pp. 1322-1326, Nov 1995.

[RBH94] K. Rothermel, I. Barth, T. Helbig. CINEMA - An Architecture for Distributed Multimedia Applications. Architecture and Protocols for High-Speed Networks, pp. 253-271, Kluwer Academic Publishers, 1994.

[SBAM96] A.D. Stoyenko, J. Bosch, M. Aksit, T.J. Marlowe. Load Balanced Mapping of Distributed Objects to Minimize Network Communication, Journal of Parallel and Distributed Computing 34, pp. 117-136, 1996.

[ShSe95] B. Shen, I.K. Sethi. Inner-Block Operations on Compressed Images. 3rd ACM Int. Multimedia Conference, San Francisco, Nov 1995.

[SMK96] L.C. de Silva, T. Miyasato, F. Kishino. Emotion Enhanced Multimedia Meetings Using the Concept of Virtual Space Teleconferencing. IEEE Int. Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996.

[Sren96] C.J. Sreenan. Resource Management System for a Broadband Multipoint Bridge. IEEE Int. Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996.

[SSJH96] F. Samaria, H. Syfrig, A. Jones, A. Hopper. Enhancing Network Services through Multimedia Data Analysers. 4th ACM Int. Multimedia Conference, Boston, Nov. 1996.

[StNa95] R. Steinmetz, K. Nahrstedt. Multimedia: Computing, Communications and Applications. Prentice Hall, Innovative Technology Series, 1995.

[Ston77] H.S. Stone. Multiprocessor Scheduling with the Aid of Network Flow Algorithms. IEEE Trans. on Software Engineering, Vol. 3, No. 1, Jan 1997.

[Tows86] D. Towsley. Allocating Programs Containing Branches and Loops Within a Multiple Processor System. IEEE Trans. on Software Engineering, Vol. 12, No. 10, pp. 1018-1024, Oct 1986.

[VKBG95] Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann and Jan Gecsei. Distributed Multimedia and QOS: A Survey. IEEE Multimedia, Summer 1995, pp. 10-18.

[Wo96] Lars Wolf. Resource Management for Distributed Multimedia Systems. Kluwer Academic Publisher, Boston/Dordrecht/London 1996.

[WoMo93] C.M. Woodside, G.G. Monforton. Fast Allocation of Processes in Distributed and Parallel Systems. IEEE Trans. on Parallel and Distributed Systems, Vol. 4, No. 2, Feb 1993.

[YMGH96] N. Yeadon, A. Mauthe, F. Garcia, D. Hutchison. QoS Filters: Addressing the Heterogeneity Gap. European Workshop on Interactive Distributed Multimedia Systems and Services, Berlin, Germany, March 1996.

[Zhan93] L. Zhang et al. RSVP: A New Resource Reservation Protocol. IEEE Netowrks Magazine, pp. 8-18, Sep 1993.

[YaSa93] S.S. Yau, V.R. Satish. A Task Allocation Algorithm for Distributed Computing Systems. IEEE Computer, pp. 336-342, 1993.

[YWPS95] S. Yalamanchili, L.T. Winkel, D. Perschbacher, B. Shenoy. Partitioning and Mapping in Embedded Multiprocessor Architectures in the Presence of Constraints. Concurrency: Practice and Experience, Vol. 7(3), pp. 167-189, May 1995.