

# Integrated Planning and Execution for Satellite Tele-Communications

Christian Plaunt\* Kanna Rajan\* Barney Pell<sup>†</sup> Nicola Muscettola<sup>‡</sup>

{plaunt,kanna,pell,mus}@ptolemy.arc.nasa.gov

NASA Ames Research Center, Mail Stop 269-2

Moffett Field, CA 94035-1000

## Abstract

The next generation of communications satellites may be designed as a fast packet-switched constellation of spacecraft able to withstand substantial bandwidth capacity fluctuation ranging from unstable weather phenomena to intentional jamming of communication. We have designed and partially implemented an architecture for managing satellite telecommunications network resources. Our approach supports advance reservations and dynamic requests, negotiation and fulfillment of prioritized Quality of Service (QoS) contracts, graceful degradation in the presence of dynamic tasks and environmental changes, and optimization of geometrically constrained resources. Our integration of planning and execution to address this task uses planning to avoid resource contentions among requested activities and to configure an independently competent execution system. Our system can be used in routine operations or as a simulation-based design tool.

## 1 Introduction

The current revolution in information technology continually produces new advances in communications capability. In its vision for the future, the US Department of Defense (DoD) perceives information as critical to tactical and strategic decisions and satellite communication as an essential operational component (Department of Defense, Space Command 1997). One of the critical technologies being closely scrutinized is the application of Asynchronous Transfer Mode (ATM) technology to satellite communications systems. Satellites are limited and expensive communications resources, and ATM technology, through quality-of-service (QoS) contracts and statistical multiplexing, offers greater flexibility and capacity than existing circuit-switched systems currently used for military satellites.

However, extending ATM to support military communication satellites requires innovations beyond standard ATM networks. Unlike most quality of service work, the military domain requires advance guarantees and hierarchical resource allocation. One of our major goals is to support these domain requirements while increasing the efficiency of resource utilization and supporting unplanned resource allocations. In addition, we must also support geometric constraints and optimizations resulting from satellite beam management.

The DoD is in the process of evaluating the design parameters needed for such a system using simulation based design. One of the tools needed as part of this design analysis is a prediction and execution component. For this, we are proposing the use of the Planner/Scheduler (PS) and Smart Executive (Exec) subsystems of the Remote Agent (RA) (Bernard *et al.* 1998; Pell *et al.* 1998a; Muscettola *et al.* 1998b). The RA will be the first artificial intelligence-based autonomy architecture to reside in the flight processor of a spacecraft (NASA's Deep Space One (DS1)). We have built from these components a new system, the Remote Agent for Satellite Tele-Communications (**RAST**).

Similar to other high-level control architectures (Bonasso *et al.* 1997; Wilkins *et al.* 1995; Drabble *et al.* 1996; Simmons 1990; Musliner *et al.* 1993), **RAST** clearly distinguishes between a *deliberative* and a *reactive* layer. In the current context PS develops a schedule based on requested bandwidth allocations known a priori. Planning/Scheduling is used to smooth out resource consumption resulting from future requests, to establish configurations to enable future requests (like needing beam coverage to make a subsequent call request), and to set execution priorities to support efficient responses to dynamic requests, taking into account environmental projections. Planning and execution must support quality of service guarantees in highly dynamic environment. PS negotiates among requests in advance, and Exec negotiates contracts and adjusts and sheds tasks based on variable priorities at

---

\*Caelum Research Corporation

<sup>†</sup>Research Institute for Advanced Computer Science

<sup>‡</sup>RECOM Technologies

run time in the dynamic environment. As a result, the execution system handles planned and unplanned dynamic resource requests and supports load balancing, quality of service, fast responses, and graceful degradation.

### 1.1 Integrated Planning and Execution

In addressing these issues, we have explored a novel and interesting integration of planning and execution. There are number of ways to integrate planning and execution that have been explored:

- plans as coordination routines for multiple agents (including humans).
- planners generate tasks networks, which are then executed by doing the right task at the right time.
- plans as programs, which are run by the executive (e.g. planning in CIRCA (Musliner *et al.* 1993)) generates a program comprised of test-action pairs).
- plans as advice, which the executive uses in running is own goals (e.g. planner produces a navigation map, which exec uses when it is heading to targets).

Our integration has aspects of several of these. The advance planning of resource allocations tells users when they should place their calls, thus preventing resource conflicts before they happen and guaranteeing resource availability. The plans themselves have task networks with explicit configuration actions (beam configuration activities) and also advice in terms of execution priority updates and projections used for monitoring plan execution.

Most planning and execution work addresses specific resource requirements, whereas this work addresses multiple types of quality-of-service contracts, with resource sharing (statistical multiplexing), preemption, and even reconfiguration (in the case of beam migration and repositioning).

Finally, our executive has independent competence, and can run with or without a plan, but performance can be enhanced with a plan.

### 1.2 Organization

In Section 2 we describe the overall problem in greater detail. Section 3 describes the **RAST** architecture. In Section 4 the details of the approaches taken in the Planning/Scheduling component. Section 5 covers the run-time execution system. We explore work related to this project in Section 6, and in Section 7 we consider the open issues and future work to which this project points. Finally, in Section 8 draw our conclusions.

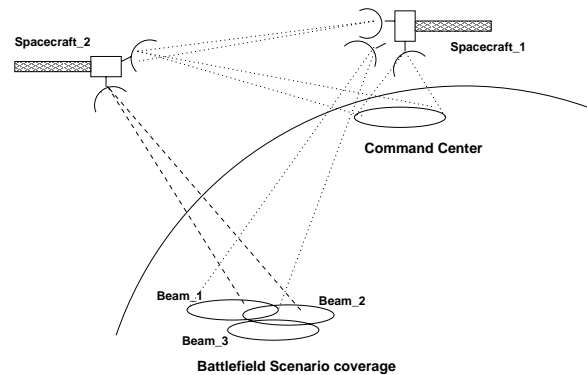


Figure 1: A simplified satellite telecommunications scenario supported by an ATM networked constellation

## 2 The Domain

### 2.1 Motivation

We are motivated by the requirements of complex, mission critical satellite tele-communications systems. In this domain, there are several conflicting goals which influence many levels of design choices (for instance, guaranteed connections versus maximal network throughput, fluctuating bandwidth, conflicting demand patterns, quality of service). These considerations make this a particularly interesting domain for our exploration. A communication network in this domain must be highly configurable and controllable in order to handle the strategic needs of the user, and also be highly autonomous in order to function efficiently in the potential absence of such control.

The objective of this overall effort is to build an operational system which can also be used as an analysis and design tool, capable of both controlling or simulating and analyzing multiple configurations, topologies, and environments in the unstable environment of mission critical communications with the purpose of controlling or designing a future generation satellite based telecommunications system. When used as a design tool, the agent generates output for designers to evaluate operations policy and provides flexibility in the operational constraints modeled. Rapid iteration of the system design is possible by comparison of throughput performance results for candidate designs. Moreover, a network planning and execution agent can optimize the policy for users and potential customers can be advised in their planning for network usage. At present, satellite communications network planning is a computation and labor intensive element of operations. The model-based planning and execution agent could improve efficiency and reduce cost and effort.

The work described in this paper is further moti-

vated by our interest in several research aspects of this domain. Issues include using planning and scheduling to smooth out resource consumption resulting from future requests, establishing configurations to enable future requests (e.g. requiring beam coverage to enable a subsequent call request), and setting execution priorities to support efficient responses to dynamic requests, taking into account environmental projections.

## 2.2 A Brief Background on ATM

The domain consists of a constellation of spacecraft which act as ATM switches directing and controlling traffic flow from a number of sources to a number of destinations (Figure 1). Traffic is based on an ATM model with different *contract types* and *priorities*. Contracts ensure a Quality of Service (QoS) so that guarantees can be made a priori about specific call connections. The user must inform the network upon connection setup of both the expected nature of the traffic and the type of QoS contract required. The idea is to ensure that critical calls, that need to get through under all circumstances, are guaranteed bandwidth capacity while those of lower priority — regardless of contract type — or of a non-critical nature are allocated bandwidth on an as available basis. Following are some terms from the ATM literature (see (Varma 1997) for a concise tutorial) we will use in this paper:

- CBR (Constant Bit Rate): Bit rate remains constant over duration of connection; requires delay, jitter and bandwidth guarantees<sup>1</sup>.
- VBR (Variable Bit Rate): Intended for supporting bursty data and defined by peak and average rate.
- ABR (Available Bit Rate): Intended for data transmission which do not preserve any timing relationship between source and destination.

In addition, in this domain we also deal with call priorities. For instance, critical calls that need to get through under all circumstances will have the highest priority. Such calls may be of any contract type, depending on the nature of the call (voice, video, data, etc.). Less critical calls might request an “expensive” contract (e.g. CBR), but also be willing to accept a less expensive contract (e.g. ABR) if that is the best contract available.

Currently, such communication is managed by restricting the identity, time, and bandwidth allocations of people and equipment that can use system to communicate. Multiple high priority channels are reserved

<sup>1</sup>We distinguish here between different peak and average bandwidth requirements among QoS contracts. E.g., CBR 2 requires roughly twice as much bandwidth as CBR 1.

just in case an important message needs to be sent. In this approach not only is the complete bandwidth allocation preallocated as a “pipe” (i.e once allocated the resources are completely tied to the user), but dynamic request allocations can only be accepted if the request is of a high enough priority, to preempt an ongoing call when enough capacity is not available. Needless to say, this is a highly suboptimal approach, especially in the forward tactical areas where frequently a large amount of bandwidth is needed on demand and where no accurate predictions can be made a priori.

## 3 System Architecture

The system architecture consists of several modules, as shown in Figure 2. The architecture is based on the components of the Remote Agent architecture (Pell *et al.* 1998a), plus several domain specific components (or simulators) which are used either at plan-time or run-time. In this section we discuss the various components of the system architecture with each module annotated as in Figure 2.

### 3.1 Plan-Time Components

As an operational system (see Figure 2), the Planner/Scheduler (3) takes input in the form of authorization requests from a Request Generator (1) and estimates of the effects of environmental conditions on bandwidth capacity fluctuations at run-time from a plan-time Environmental Expert (2). From this input, the PS generates a plan which includes the reservation schedule, beam movements, required configuration, policies, priority schedules, and so on, that will be required to carry out the authorized calls while maximizing dynamic potential. The schedule produced from these inputs is supplied both to the users of the system (in order to regulate usage by informing users whether their reservation has been accepted or not and when to place their call) and to the run-time execution system. Thus, the plan time components configure usage patterns as well as system resources and priorities.

### 3.2 Run-Time Execution Components

The run-time execution components monitor and execute the execution schedule while responding to dynamic requests and environmental changes. The major tasks at run-time are (i) to determine whether a call request can and should be admitted to the system, and (ii) to administer those call requests which have already been admitted to the system.

The execution schedule is executed by the Plan Runner (4), as follows. The primary form of configuration change is to move a beam to a new location, by sending the corresponding command to the Beam Manager

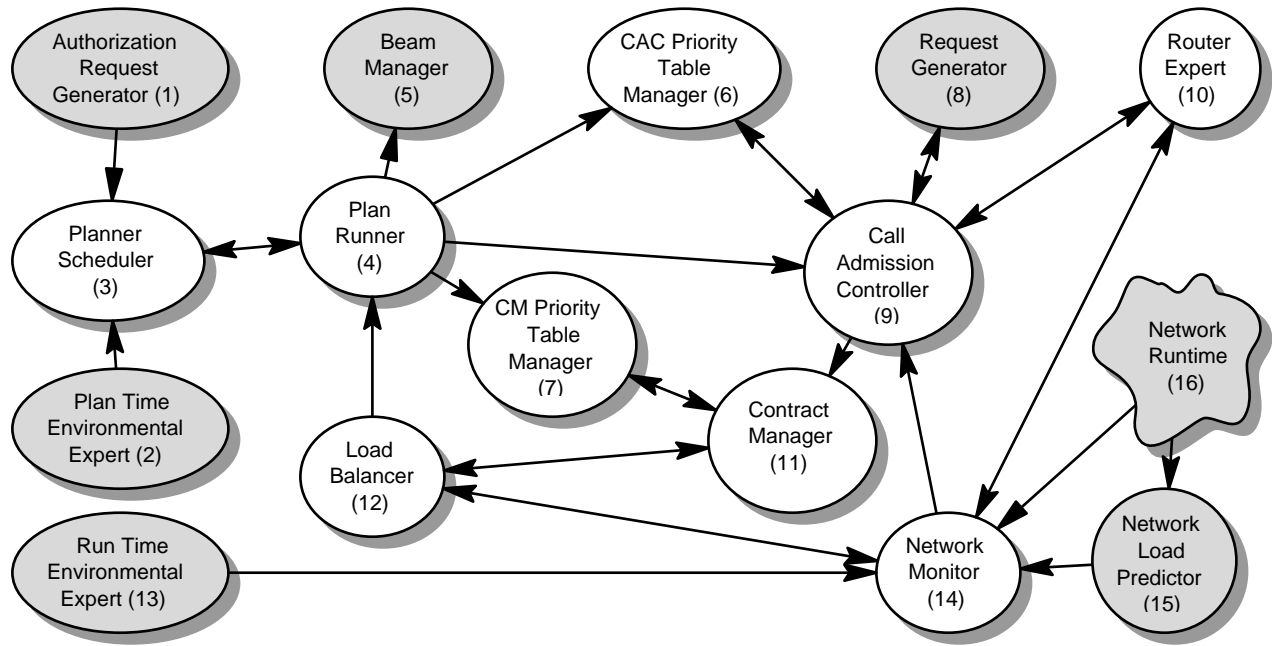


Figure 2: The **RAST** application architecture for a Modular High Level ATM Network Controller. This architecture can also be used as a simulation based design tool by simulating the shaded components in this figure.

(5). Policy and priority changes are issued by sending the commands to the corresponding priority table manager, one for the Call Admission Control (CAC) Priority Manager (6). and one for the Contract Manager (CM) Priority Table Manager (7). These priority tables, which are consulted dynamically at run time, control the behavior of the major run-time execution components of the system, the Call Admission Controller (9) and the Contract Manager (11). The Plan Runner uses information from the Load Balancer (12) to determine if the plan execution is proceeding within the bounds of the planner. If not, it continues with the current policy while requesting a new plan.

At run time, the Call Admission Controller (9) receives initiation requests from distributed users, represented here as a dynamic Request Generator (8). These call initiation requests are typically a variable mixture of scheduled and unscheduled requests. Each such request specifies information about the call contract requested, which includes quality of service contract types and parameters. When the system is running as simulation, the mixture of these requests is designed to simulate various “real world” probability distributions.

The CAC decides how to handle the requests based on (i) the policy specified by the CAC Priority Table Manager (6), (ii) the state of the network (i.e. current coverage, capacity, and usage) as reported by the

Network Monitor (14), (iii) the availability of communication resources as reported by the Router Expert (10), and (iv) the allowable types of contracts in the call request received. The initiation requests can be (i) serviced as requested, (ii) serviced but with some alternative contract type (as allocated by the Router Expert (10) and accepted by the call requester), or (iii) denied.

The Router Expert (10) allocates (or denies) connection contracts in response to requests from the Call Admission Controller (9). It decides whether or not to allocate such contracts based on several factors, including the state of the network reported by the Network Monitor (14) and the Network Runtime (16), availability of point to point virtual circuits, and so on. When a call request is accepted, the call and its allocated contract are passed to the Contract Manager (11) which tracks the calls thereafter.

The other major functionality provided by the run-time system is contract management, embodied here in the Contract Manager (11) and Load Balancer (12).

The Contract Manager is the run-time module which keeps track of all the contracted calls which have been received from the Call Admission Controller (10). The Contract Manager, based on the priority policy in the CM Priority Table Manager (7), and the current state of the network as reported by the Load Balancer (12), controls all of the “in progress” call traffic.

If at any time insufficient resources exist to support the current calls with sufficient robustness, The Contract Manager interacts with the Load Balancer (12) to free up resources. The Load Balancer keeps network usage within capacity by migrating call among different possible routes, reducing the bandwidth of calls with contracts which allow this, shedding low priority calls, and potentially repositioning beams to optimize ground coverage. In conjunction with the Contract Manager, lower priority calls can be moved or killed to make way for higher priority calls. This ability to migrate or shed calls becomes particularly important when the network is operating in an unstable, dynamic environment where network capacity can fluctuate enormously.

### 3.3 Network Components

The Network Monitor (14) is the interface between the run-time execution system and the network itself. Based on input from the run time Environmental Expert (13), the Network Runtime (16), and the Network Predictor (15), it reports the current total bandwidth capacity and current actual usage to the Load Balancer (12), the Router Expert (10) and the Call Admission Controller (9) at run time. The run time Environmental Expert (13) simulates changes of the environment which affect bandwidth capacity on the beams (e.g. weather changes, hardware problems, jamming, etc.). The Network Predictor (15) is a traffic expert which can be used by the plan-time and run-time Environmental Experts (2, 13) for better network usage predictions.

Finally, the Network Runtime (16) is the real (or simulated) network. Primarily, it feeds the Network Monitor (14) with run time fluctuations in network load, capacity, congestion, outages, and so on.

### 3.4 Simulation-Based Design Tool

The system can also be used as a simulation-based design tool. This is accomplished by simulating user and environmental factors (see the shaded components of Figure 2). Our modular design enables the external interface points to be unaware of whether the input is coming from a simulator or from operational use. For example, rather than running of real authorization requests, input to the planner can be provided by a statistical Authorization Request Generator. Similarly, dynamic calls can be generated in accord with alternative test cases for usage patterns, rather than coming from real users, and environmental conditions such as failures, weather changes, or jamming can be simulated. Together, this supports use of the system for 'what-if' analysis, in which we run different networks,

policies, and assumptions through simulated operational contexts and collect statistics such as throughput and call completion rates.

Clearly, this architecture is divided between plan-time and run-time. The focus of the plan-time components is to smooth the fluctuations in the actual run-time call requests as much as possible. The focus of the run-time components is to respond to just such fluctuations.

## 4 The Planning/Scheduling Component

The objective of the Planner/Scheduler (PS) is to schedule system resources and requested traffic allocations as optimally as possible. The Exec then takes this generated schedule and changes system configuration to support the scheduled calls and to meet the demands of dynamic real-time traffic to the extent possible. Using PS in the loop helps to optimize run time configuration and allocation and also permits dynamic call initiation by reconfiguring the network (antennas) to cover critical regions.

The PS is a timeline based non-linear temporal constraint posting planner which uses chronological backtrack search. Temporal information in the plan is represented within the general framework of Simple Temporal Constraint networks, as introduced by Dechter, Meiri, and Pearl (Dechter *et al.* 1991) in a Temporal Database (TDB). Details of the HSTS planner/scheduler and TDB can be found in (Muscettola 1994).

### 4.1 The Scheduling Process

The PS component generates a schedule of calls based on a domain model. The model describes the set of constraints that all the calls have to satisfy. The schedules consist of several parallel *timelines*, each of which consist of a sequence of *tokens*. A timeline in this domain describes the condition of each channel over time. Each call is a token on a timeline. In our domain there are primarily three token types; a call request token which specifies all the request parameters necessary for scheduling, a beam capacity token type which gives instantaneous capacity at any time and a beam location token type which specifies to the planner where the beam coverage is. Beam slewing (when the spacecraft's beam is to be transitioned from one area of coverage to another) is assumed to be instantaneous so no token is required.

**Beam Scheduling** The PS receives as input a traffic request allocation which specifies for each call request, the contract type, priorities, requested capacity, duration of the call and the source and destination

target areas. The PS then tentatively builds a partial plan based on the requested start times and duration. A constraint is posted on the beam timeline specifying a region of beam coverage which will satisfy the call constraints. Given a set of such requests, the planner searches through the space of possible configurations of the limited set of beams in order to optimize coverage. A simple example is shown in Figure 3. Calls (represented as tokens) assigned to some channel (represented as timelines) request bandwidth (not shown) and beam coverage. As the partial schedule is built both the location and the duration of the beam at those requested locations get refined. When no more beam requests are to be satisfied the PS can then determine slew boundaries when the spacecraft can move the beam from one area of coverage to another. Scheduling beam coverage as a result, is a matter of ensuring that most (if not all) requested calls are covered by some beam. Those calls that are not covered will be rejected.

**Bandwidth Scheduling** We currently use a simple forward dispatching strategy which is adequate to schedule all calls. As a result calls scheduled on a specified channel take up the 'real estate' on that channel. Any subsequent call also requiring capacity on that channel and intersecting temporally with a previously scheduled call will currently be rejected at the scheduling phase. Such rejected calls however have the opportunity to request bandwidth at run time where lower priority and contract type calls can be shed. In the future however, the problem that needs to be tackled is complicated by the introduction of contract types and priority. In that event, contract types and priority schemes will allow preemption of scheduled calls already placed on the timelines. So for instance if a CBR request is posted to a temporal duration  $[t_1, t_2]$  and if the bandwidth capacity exists, this call could be accommodated within the temporal duration. If not, any previously scheduled ABR or VBR calls would need to be rescheduled to accommodate this incoming CBR call. Correspondingly if a non-CBR request comes in after a CBR call capacity is satisfied, then depending on the request type, its duration and requesting range, the new call request could be either moved or rejected outright. This strategy will have to ensure that a CBR will always have the capacity reserved for it when scheduled, while a ABR could be shed at execution time. Effectively this calls for a CAC (9) style priority table manager, *but at schedule time*. Policies for this table can then be adjusted to allow selection of different scheduling strategies by the user.

## 4.2 Model Representation

The plan model consists of definitions for all the timelines, definitions for all the tokens that can appear on those timelines, and a set of temporal constraints that must hold among the tokens in a valid schedule. The planner model is described in a domain description language (DDL) (Mussettola 1995), and is represented as part of the planner's TDB.

Temporal constraints are specified in DDL by *compatibilities*. A compatibility consists of a *master token* and a boolean expression of temporal relations that must hold between the master token and *target tokens*. An example is shown in Figure 4. The first constraint specifies that a call request master token can only be satisfied if its peak bandwidth capacity is satisfied, and it is within the confines of some beam which provides coverage. Additionally, another call is to follow (precede) it on this channel.

Heuristics tell the planner what decisions are most likely to be best at each choice point in the planner search algorithm, thereby reducing the search. In HSTS, the heuristics are closely intertwined with the model and can be used to specify which compatibility to place on the planners agenda mechanism to focus its search. In the current system acquiring good heuristics to make the planner search computationally tractable is still an issue.

## 5 Run-Time Execution

### 5.1 Dynamic Policy Enforcement

The run-time execution system's objective in **RAST** is to enforce a small number of communication policies in a variety of environmental network loading situations in order to analyze their effects on the system. That is, the Exec's job is (1) to enforce policy on priority-based bandwidth allocation, (2) within that policy, to service the scheduled allocations and configuration changes generated by PS, and (3) to service unscheduled bandwidth allocation requests for bandwidth dynamically as (1) and (2) allow.

In particular, this means that the active run-time policy will determine the default behavior of the Exec (and the behavior of the communications system) when (1) there is no plan available (for whatever reason), (2) between the time when a plan is broken and a new plan is received, and (3) when there is not enough bandwidth to satisfy the current plan, etc.

Currently, the communication policy of interest is (1) to service all dynamic communication requests, scheduled or not, in highest priority first order until either all are serviced or bandwidth capacity is reached; and (2) when bandwidth capacity is exceeded, shed

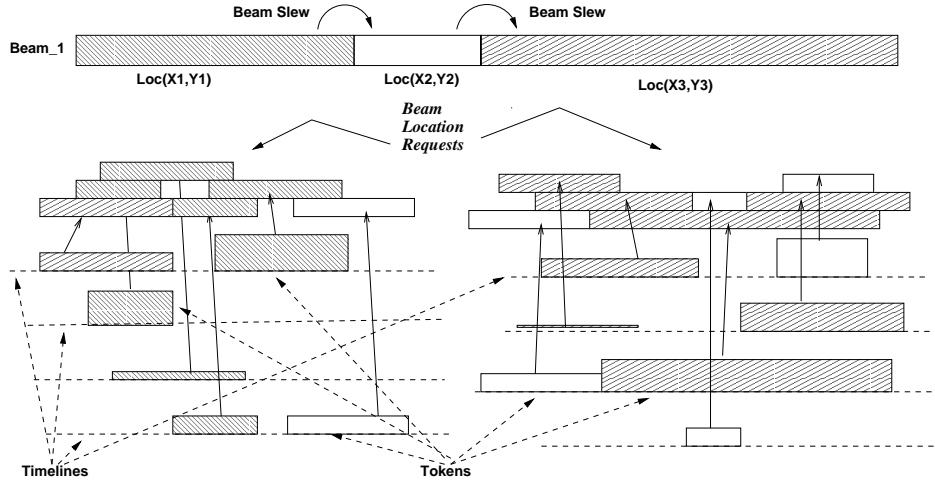


Figure 3: A partial schedule with calls requesting bandwidth and beam coverage. Height of a token indicates amount of bandwidth requested while shading corresponds to a specific beam coverage location. Merging of beam location requests results in the PS scheduling beams as shown in the top of the figure.

```
(Define_Compatibility ;; compats on Call Request
(Call_Request ?ID ?Contract_Type ?Priority ?Cap ?Call_Source
 ?Call_Dest ?Est ?Lst ?Duration ?Beam)
:parameter_functions ( ((?duration_ <- ?Duration)) )
:compatibility_spec
(AND
;; requires a specific amount of bandwidth capacity
(AND (equal (DELTA MULTIPLE (Capacity) (+ ?Cap Used)))
;; needs to request a beam location based on the call source
(contained_by (MULTIPLE ((Beam Beam_1_Pointing_SV)) ((Beam_Loc (?Call_Source))))
(contained_by (SINGLE ((Beam Beam_1_Pointing_SV)) ((Beam_Loc (?Call_Source)))) )
;; is followed either by another call immediately or a NOOP
(OR (met_by (SINGLE ((Call_UL CALL_1_SV)) (Call_Request)))
(met_by (SINGLE ((Call_UL CALL_1_SV)) (No_Call_Activity))) )
;; is preceded either by another call immediately or a NOOP
(OR (meets (SINGLE ((Call_UL CALL_1_SV)) (Call_Request)))
(meets (SINGLE ((Call_UL CALL_1_SV)) (No_Call_Activity))) )
;; and allocates an equivalent bandwidth for the downlink phase
(equal (SINGLE ((Call_DL CALL_1_SV)) ((Call_DL_Request (?Id ?Contract_Type ?Priority
?Cap ?Call_Source ?Call_Dest
?Est ?Lst ?Duration ?Beam)))))) )

(Define_Compatibility ;; compats on beam pointing/location
(SINGLE ((Beam Beam_1_Pointing_SV)) ((Beam_Loc (?Call_Source))))
:compatibility_spec
(AND
;; precedes and succeeds another beam pointing token
(met_by (SINGLE ((Beam Beam_1_Pointing_SV)) (Beam_Loc)))
(meets (SINGLE ((Beam Beam_1_Pointing_SV)) (Beam_Loc))) )

(Define_Compatibility ;; compats on no activity fillers
(SINGLE ((Call_DL CALL_1_SV))(No_Call_Activity))
:compatibility_spec
(AND
(meets (SINGLE ((Call_DL CALL_1_SV)) (Call_DL_Request)))
(met_by (SINGLE ((Call_DL CALL_1_SV)) (Call_DL_Request))) )
```

Figure 4: An example of a compatibility constraint in the **RAST** Planner model.

communication allocations in lowest first priority order until it is no longer exceeded.

At run time, whenever a conflict arises over bandwidth allocation in either the Call Admission Controller or the Contract Manager, they consult a dynamic table of priorities to determine which call(s) are accepted, migrated, denied, or shed. An example of such a table is shown in Table 1.

Two such tables are maintained for use by the CAC and CM, which consult them in order to increase or decrease bandwidth usage. These tables each have a "manager" which the Plan Runner commands in order to set and reset these tables.

Given a clear policy on such priorities, the run-time system will work even in the absence of a plan. Further, there can be multiple policies which the Exec can enforce, perhaps depending on various environmental or experimental circumstances.

## 5.2 Run-Time Execution

At run-time, the Exec accepts a stream of call requests, some scheduled in advance, others not. Requests are either to start or release a connection. Start requests contain data about the call's requested contract, assigned priority, origin, destination, and so on.

When the CAC receives a call, with the help of the Router Expert and the Network Monitor, either a route and a contract are granted or denied. If the contract is granted, the call is connected via the route (uplink beam, downlink beam, etc.) assigned at the granted bandwidth and QoS contract, and the call and contract are passed on to the Contract Manager. In the case of a release request, the relevant parts of the system are notified, and the call (and its associated resources) are released.

The Contract Manager administers all of the "in progress" traffic in the system. In order to keep bandwidth usage within capacity, it has the ability to migrate calls among beams, reduce (or "squeeze") the bandwidth usage of calls with certain contracts, or to terminate calls. For example, if bandwidth becomes unexpectedly restricted, the CM can migrate, squeeze, and shed calls in reverse priority order to preserve as many virtual circuits as possible within the bandwidth available (Figure 5) or reduce ABR rates to keep usage within network load capacity. Conversely, when usage falls below capacity, ABR rates can be increased ("unsqueezed") to use the extra bandwidth.

## 6 Related Work

This paper is among the first work concerned with the problem of integrating planning and execution to support both advanced reservations and dynamic requests

for quality-of-service (QoS) style resource-allocation problems. QoS requirements have emerged mainly in the telecommunications domain and have come to the forefront in the context of Asynchronous Transfer Mode (ATM) communication networks. Major areas of research on intelligent agents in telecommunication applications (Albayrak 1998) include *network configuration*, *call admission control*, and *routing*.

Hayzelden (Hayzelden & Bigham 1998) describes a heterogeneous multi-agent architecture for ATM networks. The architecture is similar to **RAST** in that it integrates a deliberative planning layer with a reactive execution layer. The problem focus is somewhat different, however, as they address the problem of network configuration (dynamically adjusting the network topology), while **RAST** addresses call admission control and load balancing (accepting and shedding calls). Also, their approach does not deal with advanced scheduling based on call reservations; rather, their planning agents watch over the network and plan modifications to it based on observed usage patterns. Similar comments pertain to the ARCHON multi-agent system (Jennings *et al.* 1996), which was also applied to network monitoring and configuration.

One aspect of network configuration that does fall within the present scope of **RAST** is *beam management*. Unlike the context of terrestrial ATM networks, where every source and destination have automatic coverage, the satellite context especially requires advanced reservations as supported by **RAST**, as the beams must be pointed to cover an area to enable call initiation from that area. Optimizing beam positioning in both planning and scheduling present interesting problems in computational geometry that also differ from network configuration problems addressed in standard ATM networks. Nielsen (Nielsen *et al.* 1997) addresses the problem of obtaining maximally efficient coverage given a set of antennas and regionally varying load requirements. This could enhance our approach to beam planning and beam migration in **RAST**.

Brown and Tong (Brown & Tong 1998; Tong & Brown 1998) address the problem of call admission control and QoS guarantees by means of reinforcement learning. Verma (S. Verma & Garcia 1998) approaches the problem by means of distribution and mathematical optimization. Both these approaches could be used to enhance the priority and policy table update mechanisms in **RAST**, although the advanced call scheduling of **RAST** is still necessary for a full solution to our problem.

Much research on load management addresses the problem of packet and call routing in telecommunication applications. Approaches include reinforcement

Rank	1	2	3	4	5	6	7	8	9	10
Contract	CBR 1	CBR 2	VBR 1	CBR 1	CBR 2	CBR 1	ABR 1	ABR 2	ABR 1	ABR 2
Priority	high	high	high	medium	medium	medium	high	high	medium	medium

Table 1: An example of the first several entries in a priority table. Priority rank is determined as a function of assigned priority and the QoS contract. That is, the number one ranked calls are high priority CBR 1, the second rank are high priority CBR 2 calls, and so on.

```
[Setting Beam Capacity for BEAM-2 to 15]
Handling network event at 5050.00 for <BEAM-2 18/15 in use (120.0%), 5 calls>
Migrating <CALL 6 :LOW_PRIORITY (13) :VBR_1 (4 s/s) :AREA_C to :AREA_D (BEAM-2)> to BEAM-1 at 5050.01
<BEAM-1 26/100 in use (26.0%), 9 calls>
Done handling network event at 5050.02 for <BEAM-2 14/15 in use (93.3%), 4 calls>
[several transactions elided]
Looking for 2 s/s on BEAM-1 for <CALL 26 :HIGH_PRIORITY (1) :CBR_1 (2 s/s) :AREA_A to :AREA_B (BEAM-1)>
Looking for 3 s/s on BEAM-2 for <CALL 12 :MED_PRIORITY (6) :VBR_1 (4 s/s) :AREA_C to :AREA_D (BEAM-1)>
Can't find 3 s/s on BEAM-2 to reclaim.
Shedding <CALL 12 :MED_PRIORITY (6) :VBR_1 (4 s/s) :AREA_C to :AREA_D (BEAM-1)> at 5077.96
<BEAM-1 26/30 in use (86.7%), 10 calls>
Accepted <CALL 26 :HIGH_PRIORITY (1) :CBR_1 (2 s/s) :AREA_A to :AREA_B (BEAM-1)> at 5077.97
<BEAM-1 28/30 in use (93.3%), 11 calls>
```

Figure 5: A trace of the run-time execution system which demonstrates call migration and shedding. CALL 16 is moved when network capacity changes, and CALL 26 is accepted after CALL 12 is shed.

learning (Boyan & Littman 1993; Tong & Brown 1998), market-based routing (Gibney & Jennings 1998), and ant-colony optimization (Bonabeau *et al.* 1998). Since our current work operates at a higher level of abstraction (call admission and modification, not packets and routing), this work could be plugged into our architecture in a modular fashion. It would be interesting to see whether the advanced reservations managed by **RAST** could be exploited by these routing mechanisms for performance improvements.

Much of the emphasis of QoS resource allocation involves reasoning about real-time cpu, bandwidth, latency, and jitter requirements, often in the presence of geometric constraints. Boddy (Boddy & Goldman 1994) addressed many of these issues in generating a scheduler to produce real-time schedules for the BOEING 777 aircraft. The CIRCA system (Musliner *et al.* 1993) also generates plans with real-time execution guarantees. Boddy and Musliner (Boddy 1996) describe a constraint-based distributed scheduling process for air traffic control. Each designated region of airspace is managed by a separate resource manager that allocates spatio-temporal windows to pilots requesting the resource. They applied similar ideas to task distribution and data volume management for distributed processing (Musliner & Boddy 1997).

Finally, several AI systems have been developed to support closed-loop plan execution. In contrast with **RAST**'s current plan execution component, the ap-

proach taken in 3T (Bonasso *et al.* 1997) has the planner watch over each step of execution. Hence the planner itself serves as an integral participant in the plan execution capability. Bresina (Bresina *et al.* 1996) describes APA, which has separate components for generation and execution of temporal plans, in which the executive is competent to carry out activity in the absence of plans, similar to the approach in **RAST**. Reece and Tate (Reece & Tate 1994) developed an execution agent for the O-Plan (Currie & Tate 1991) planning system. The combined system supports a plan repair mechanism (Drabble *et al.* 1996) that is more sophisticated than that supported by **RAST** at present, as it allows the planner to edit any unexecuted portion of the currently executing plan. Our redesigned plan execution component (Pell *et al.* 1998b) will support a similar editing capability, based on the work in O-Plan and also in Cypress (Wilkins *et al.* 1995). Finally, Lockheed's Tactical Planning and Execution System (TPES) (Mitchell 1997) is an interesting related system that supports many execution and replanning capabilities with a high level of human interaction.

## 7 Open Issues and Future Work

### 7.1 Open Issues

There are a number of open issues this domain has brought out. While we have addressed how to reconcile advanced reservations and dynamic requests within

an unpredictable environment, we have as yet to determine how our design scales up to a constellation of spacecraft. Traffic patterns and routing efficiencies are bound to affect the performance of the system. One interesting issue to explore would be to perform Machine Learning for load prediction and apply it to the Network Predicting (15) component. Determining schedule quality and ensuring that the PS generates a dispatchable schedule for the Exec (Muscettola *et al.* 1998a) are two other interesting tasks.

## 7.2 Future Work

What we have described in this paper is, in part, work in progress. We have developed the PS models and the Exec interfaces to most of the run time monitoring and execution software, and are running the Exec in a standalone mode with no planner input.

We are currently only demonstrating a modest scenario with 2 beams and 20 channels per beam, though we subsequently plan to increase the number of beams and hence the number of call requests this system can handle. In the near term we will be injecting various failure scenarios into both the plan-time and run-time environment (e.g. restricting the bandwidth because of jamming or atmospheric phenomena) and modeling the uplink and downlink segments separately. The latter would allow us to analyze throughput rates for each spacecraft which is acting as an ATM switch by changing the on board buffering capacity that each spacecraft provides.

## 8 Conclusion

We have reported here on a partially implemented architecture for managing satellite tele-communications network resources. We have used an approach which supports advance reservations and dynamic requests, negotiation and fulfillment of prioritized quality of service (QoS) contracts, graceful degradation in the presence of dynamic tasks and environmental changes, and optimization of geometrically constrained resources. Our integration of planning and execution addresses multiple types of quality-of-service contracts, with resource sharing (statistical multiplexing), preemption, and even reconfiguration (in the case of beam migration and repositioning).

We have explored an interesting integration of planning and execution, which combines several techniques, including plans as advice, coordination routines and task networks. Finally, our system can be used in routine operations or as a simulation-based design tool.

**Acknowledgment** We wish to thank Gregory Dorais of NASA Ames for useful technical discussions.

Scott Sawyer, Laura Plice, Tom Fall, Marilyn Golden and Gregory White of Lockheed Martin provided the necessary domain knowledge and the framework necessary for our understanding of this problem.

## References

- Albayrak, S., editor (1998). *Intelligent Agents for Telecommunications Applications*. Lecture Notes in Computer Science. Springer-Verlag.
- Bernard, D. E., G. A. Dorais, C. Fry, E. B. G. Jr., B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, & B. C. Williams (1998). Design of the remote agent experiment for spacecraft autonomy. In *Proceedings of the IEEE Aerospace Conference*, Snowmass, CO. IEEE.
- Boddy, M. S. (1996). Contract-based distributed scheduling for a next generation air traffic management system. Technical report, Honeywell Technology Center.
- Boddy, M. S. & R. P. Goldman (1994). Empirical results on scheduling and dynamic backtracking. In D. Atkinson, editor, *Proceedings of the Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, Pasadena, California. Jet Propulsion Laboratory.
- Bonabeau, E., F. Henaux, S. Guerin, D. Snyers, P. Kuntz, & G. Theraulaz (1998). Routing in telecommunications networks with ant-like agents. In (Albayrak 1998).
- Bonasso, R. P., D. Kortenkamp, D. Miller, & M. Slack (1997). Experiences with an architecture for intelligent, reactive agents. *JETAI*, 9(1).
- Boyan, J. A. & M. L. Littman (1993). Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, & J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 671–678. Morgan Kaufmann, San Francisco CA.
- Bresina, J., W. Edgington, K. Swanson, & M. Drummond (1996). Operational closed-loop observation scheduling and execution. In L. Pryor, editor, *Procs. of the AAAI Fall Symposium on Plan Execution*. AAAI Press.
- Brown, T. X. & H. Tong (1998). Reinforcement learning for admission control. In *Procs. of Snowbird 98*.

- Currie, K. & A. Tate (1991). O-plan: the open planning architecture. *Art. Int.*, 52(1):49–86.
- Dechter, R., I. Meiri, & J. Pearl (1991). Temporal constraint networks. *Art. Int.*, 49:61–95.
- Department of Defense, Space Command (1997). *Advanced Satellite Communications Capstone Requirements Document*.
- Drabble, B., A. Tate, & J. Dalton (1996). O-plan project evaluation experiments and results. Oplan Technical Report ARPA-RL/O-Plan/TR/23 Version 1, AIAI.
- Gibney, M. & N. Jennings (1998). Dynamic resource allocation by market based routing in telecommunications networks. In (Albayrak 1998).
- Hayzelden, A. & J. Bigham (1998). Heterogeneous multi-agent architecture for atm virtual path network resource configuration. In (Albayrak 1998).
- Jennings, N. R., E. Mamdani, J. M. Corera, I. Laresgoiti, L. F. Perriolallat, P. Skarek, & L. Z. Varga (1996). Using archon to develop real-world dai applications, part 1. *IEEE Expert*, 11(6):64–70.
- Mitchell, S. W. (1997). A hybrid architecture for real-time mixed-initiative planning and control. In *Procs. of AAAI-97*, pages 1032–1037, Cambridge, Mass. AAAI, AAAI Press.
- Muscettola, N. (1994). HSTS: Integrating planning and scheduling. In M. Fox & M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann.
- Muscettola, N. (1995). *HSTS Domain Description Language v1.2 User Manual*.
- Muscettola, N., P. Morris, & I. Tsamardinos (1998a). Reformulating temporal plans for efficient execution. In *Proc. of Sixth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR '98)*.
- Muscettola, N., P. P. Nayak, B. Pell, & B. C. Williams (1998b). Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1/2). To Appear.
- Musliner, D. & M. S. Boddy (1997). Contract-based distributed scheduling for distributed processing. In *Working Notes of the AAAI Workshop on Constraints and Agents*, Providence, RI.
- Musliner, D., E. Durfee, & K. Shin (1993). Circa: A cooperative, intelligent, real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6).
- Nielsen, F., P. Calégari, F. Guinec, & P. Kuonen (1997). Combinatorial optimization algorithms for radio network planning. In *Proc. 10th Franco-Japanese Franco-Chinese Conf. Combin. and Comp. Sci.* Palaiseau, FRANCE.
- Pell, B., D. E. Bernard, S. A. Chien, E. Gat, N. Muscettola, P. P. Nayak, M. D. Wagner, & B. C. Williams (1998a). An autonomous spacecraft agent prototype. *Autonomous Robots*, 5(1).
- Pell, B., G. A. Dorais, C. Plaunt, & R. Washington (1998b). The remote agent executive: Capabilities to support integrated robotic agents. In A. Schultz & D. Kortenkamp, editors, *Procs. of the AAAI Spring Symp. on Integrated Robotic Architectures*, Palo Alto, CA. AAAI Press.
- Reece, G. & A. Tate (1994). Synthesizing protection monitors from causal structure. In *Procs. AIPS-94*. AAAI Press.
- S. Verma, R. K. P. & A. L. Garcia (1998). Call admission and resource reservation for guaranteed qos services in internet. *Computer Communications Journal*, 21(4). Special issue on Building Quality of Service into Distributed Systems.
- Simmons, R. (1990). An architecture for coordinating planning, sensing, and action. In *Procs. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292–297, San Mateo, CA. DARPA, Morgan Kaufmann.
- Tong, H. & T. X. Brown (1998). Optimizing admission control and routing while ensuring quality of service in multimedia networks via reinforcement learning. In *IEEE Infocom '99*, New York. Submitted.
- Varma, A. (1997). Tutorial on Traffic Management in ATM Networks. In *MILCOM 1997*.
- Wilkins, D. E., K. L. Myers, J. D. Lowrance, & L. P. Wesley (1995). Planning and reacting in uncertain and dynamic environments. *JETA1*, 7(1):197–227.