

IpexT: Integrated Planning and Execution for Military Satellite Tele-Communications

Christian Plaunt Kanna Rajan

Caelum Research Corporation

NASA Ames Research Center

Mail Stop 269-2 • Moffett Field, CA 94035

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

Abstract

The next generation of military communications satellites may be designed as a fast packet-switched constellation of spacecraft able to withstand substantial bandwidth capacity fluctuation in the face of dynamic resource utilization and rapid environmental changes including jamming of communication frequencies and unstable weather phenomena. We are in the process of designing an integrated scheduling and execution tool which will aid in the analysis of the design parameters needed for building such a distributed system for nominal and battlefield communications. This paper discusses the design of such a system based on a temporal constraint posting planner/scheduler and a smart executive which can cope with a dynamic environment to make a more optimal utilization of bandwidth than the current circuit switched based approach.

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 military satellite communications.

Satellites are limited and expensive communications resources and ATM's offers greater flexibility and capacity than existing circuit-switched systems. As a first step however, 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.* 1998; 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)). The system we have designed based on these components, is called **IpexT**.

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), **IpexT** clearly distinguishes between a *deliberative* and a *reactive* layer. In the current context the Planner/Scheduler develops a schedule based on requested bandwidth allocations known a priori. As in the DS1 scenario, the Exec issues commands to the low-level real-time control software according to the scheduler's directives and it locally reacts to sensory information which monitors the runtime network performance. Unlike the DS1 scenario however, the environment is substantially more dynamic and uncertain. Requests to use the bandwidth can come at any time from any geographical location. Additionally environmental factors can conspire to fluctuate the actual bandwidth at any time. Therefore a schedule generated by PS not only has to be as optimal as possible, but must also be robust, anticipating run time changes. In addition, the Exec must not only cope with dispatching these plans in a highly dynamic environment but it needs to make the most of the available bandwidth at run time.

In this paper, we describe the overall problem in greater detail in the "Domain" section, followed by the **IpexT** architecture in the "System Architecture" section, with details of the approaches taken in the "Planning/Scheduling Component" and "Run-Time Execution" sections. We conclude with the current status and future work envisioned in "Open Issues and Future Work".

The Domain

Motivation

We are motivated by the requirements of a complex military communications systems. In this domain, there are several conflicting goals which influence many

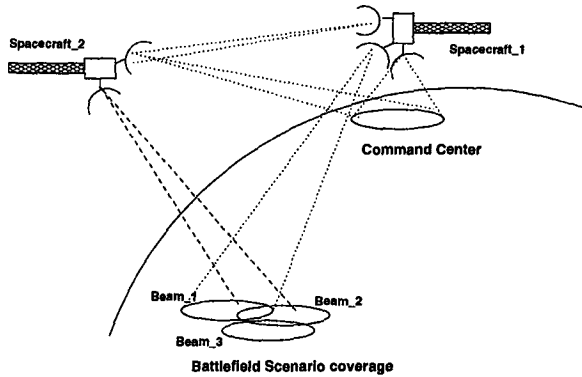


Figure 1: A simplified battlefield scenario supported by an ATM networked constellation

levels of design choices (for instance, guaranteed connections versus maximal network throughput, fluctuating bandwidth, conflicting demand patterns, quality of service). These considerations — an inherently unstable environment which allows for a variable level of autonomy — 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 analysis and design tool capable of simulating and analyzing multiple configurations, topologies, and environments in the unstable environment of military communications with the purpose of designing a future generation satellite based telecommunications system. In the design phase, the planner/scheduler would generate output for designers to evaluate operations policy by providing flexibility in the operational constraints modeled. Rapid iteration of the system design would be possible by comparison of throughput performance results for candidate designs. Moreover, network planners can optimize the policy for users and potential customers can be advised in planning for network usage.

After design optimization, the planner/scheduler has the potential to migrate to operational use. At present, satellite communications network planning is a computation and labor intensive element of operations. The model-based planning agent would improve efficiency and reduce cost and effort.

A Brief Background on ATM's

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 requests. 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 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 guarantees¹.
- 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. For instance, critical calls that need to get through under all circumstances will have the CBR contract type. Calls that might not be so critical could request VBR while low priority or non-critical data would be categorized as using ABR with the expectation that they can be shed at run time.

Currently, the military manages communication by restricting who, when, and the bandwidth of people and equipment that communicate. Multiple high priority channels are reserved just in case an important message needs to be sent. In this approach not only is the complete bandwidth 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.

System Architecture

The IpexT architecture implemented, consists of a cluster of several modules, as shown in Figure 2. The

¹We 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.

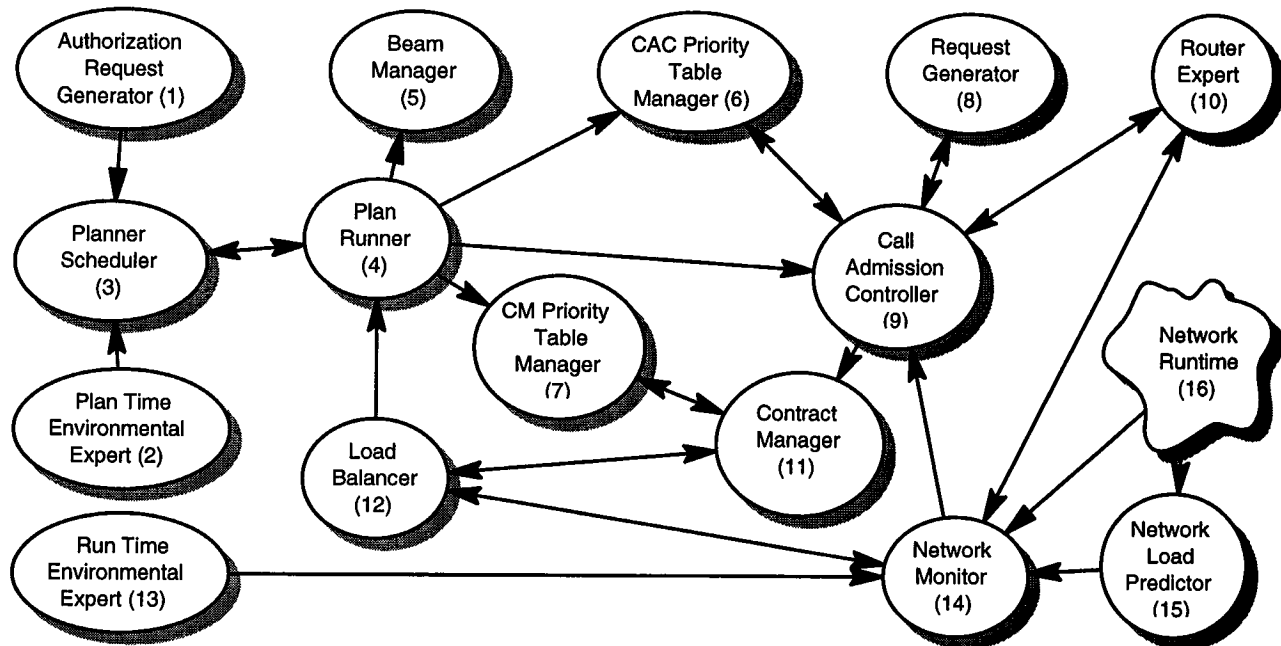


Figure 2: The **IpexT** architecture for a Modular High Level ATM Network Controller

architecture is based on the components of the Remote Agent architecture (Pell *et al.* 1998), plus several domain specific components and 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.

Plan Time Components

The Authorization Request Generator (1) is a simulator that generates call “reservations” for the Planner/Scheduler (3) at plan time. The plan time Environmental Expert (2) supplies the Planner/Scheduler (PS) with estimates of the effects of environmental conditions on bandwidth capacity, which PS uses during plan generation. The PS (3) generates a schedule of bandwidth demand for authorized calls based on the input for a specific planning horizon. The schedule produced from these inputs is supplied both to the users of the system (to order to regulate usage) and to the run-time execution system.

Run Time Execution Components

The major tasks of the run-time execution system, the Exec, are (i) to determine whether a call should be admitted to the system, and (ii) to administer those calls which have been admitted.

The Plan Runner (4) executes the plan provided by PS (3). This consists of commanding the Beam Manager (5) in order to provide uplink/downlink coverages, and commanding changes to the Call Admission

Control (CAC) Priority Table Manager (6) and the Contract Manager (CM) Priority Table Manager (7). These priority tables control the behavior of the major run-time execution components of the system, the Call Admission Controller (9) and the Contract Manager (11).

The run time Request Generator (8) is a simulator which sends real time call request traffic to the Call Admission Controller (9). These requests are a variable mixture of scheduled and unscheduled calls designed to simulate various probability distributions.

The Call Admission Controller (9), the central component of the process, receives requests for calls at run time from the Request Generator (8), and 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, handles the requests. The call requests can be (i) serviced as requested, (ii) serviced but with some alternative contract type (as allocated by the Router Expert (10)), or (iii) denied. When a call request is accepted, the call is passed, along with the allocated contract to the Contract Manager (11).

The Router Expert (10) is a run-time simulator which allocates connection contracts in response to requests from the Call Admission Controller (9). It de-

cides 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 Simulator (16), availability of point to point virtual circuits, and so on.

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 contracted calls received from the Call Admission Controller (10), and based on the priority policy in the CM Priority Table Manager (7), and the current state of the network from the Load Balancer (12), controls the in progress call traffic.

Network Simulation 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 Simulator (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 Simulator (16) is our "real" network. Primarily, it feeds the Network Monitor (14) with run time fluctuations in network capacity.

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.

The Planning/Scheduling Component

The objective of the Planner/Scheduler (PS)² is to schedule traffic allocations which are known before hand, as optimally as possible. The Exec then takes this generated schedule and in trying to place the scheduled calls, also has to meet the demands of a dynamic real-time traffic that can request the same band-

²Note that while we refer to a *planner/scheduler*, scheduling of bandwidth resources is the primary activity that the PS does. The planning component is restricted to deciding when to slew the spacecraft to ensure beam coverage for a priori requests. Our use of the term *planning* as a result, is somewhat loose in this paper.

width. The objective of having the PS in the loop is to ensure that run time allocations are not sub-optimal.

The PS is a timeline based non-linear temporal constraint posting planner which uses chronological back-track 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).

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. A conglomeration of such requests will then allow the planner to search through all the possible ways of specifying where the limited set of beams need to provide 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.

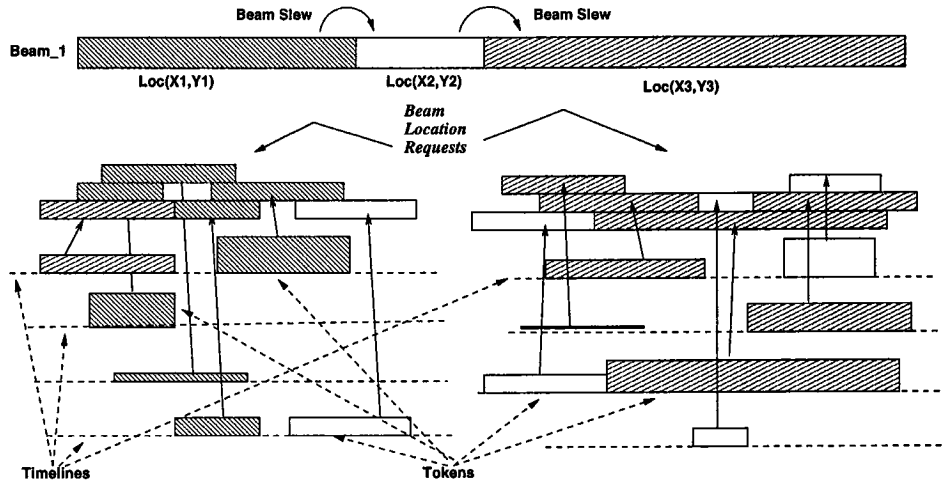


Figure 3: A partial schedule with calls requesting bandwidth and beam coverage. Height of a token indicates amount of bandwidth requested while shading corresponds to a specific beam coverage location. Conglomeration of beam location requests results in the PS scheduling beams as shown in the top of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

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 to this table can then be adjusted to allow selection of different scheduling strategies by the user.

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) (Muscettola 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.

Run-Time Execution

Dynamic Policy Enforcement

The run-time execution system's objective in **IpexT** is to enforce a small number of communication policies

```

(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
;; preceeds 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 **IpeX**T Planner model.

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 execute the scheduled allocations 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 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 dy-

namic 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.

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 assigned beam(s) at the granted bandwidth, and the call and contract

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.

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, or to terminate calls. For example, if bandwidth becomes unexpectedly restricted, the CM can migrate or 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 to use the extra bandwidth.

Open Issues and Future Work

Open Issues

There are a number of open issues this domain has brought out. While we have addressed how to deal with an unpredictable environment with an open architecture, 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 to ensure that the PS generates a dispatchable schedule for the Exec is another, something we are currently in the process of addressing (Muscettola *et al.* 1998a).

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.

Acknowledgment We wish to thank Nicola Muscettola, Barney Pell and 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

- 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. To Appear.
- Bonasso, R. P., D. Kortenkamp, D. Miller, & M. Slack (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical AI*, 9(1).
- Dechter, R., I. Meiri, & J. Pearl (1991). Temporal Constraint Networks. *Artificial Intelligence*, 49:61–95.
- Department of Defense, Space Command (1997). *Advanced Satellite Communications Capstone Requirements Document*.
- Drabble, B., A. Tate, & J. Dalton (1996). Oplan project evaluation experiments and results. Oplan Technical Report ARPA-RL/O-Plan/TR/23 Version 1, AIAI.
- Muscettola, N. (1994). HSTS: Integrating planning and scheduling. In M. Fox & M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann.

```

[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.

- 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 *Proceedings of Sixth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, Trento, Italy. To appear.
- Muscettola, N., P. P. Nayak, B. Pell, & B. Williams (1998b). Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*. To Appear.
- Musliner, D., E. Durfee, & K. Shin (1993). Circa: A cooperative, intelligent, real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6).
- Pell, B., D. E. Bernard, S. A. Chien, E. Gat, N. Muscettola, P. P. Nayak, M. D. Wagner, & B. C. Williams (1998). An autonomous spacecraft agent prototype. *Autonomous Robotics*, 5(1). To Appear.
- 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.
- Varma, A. (1997). Tutorial on Traffic Management in ATM Networks.
- Wilkins, D. E., K. L. Myers, J. D. Lowrance, & L. P. Wesley (1995). Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI*, 7(1):197-227.