A SIMULATION TOOL TO VALIDATE AND COMPARE SATELLITE TDMA ACCESS SCHEMES

Nedo Celandroni N.Celandroni@cnuce.cnr.it
Erina Ferro E.Ferro@cnuce.cnr.it
Francesco Potortì F.Potorti@cnuce.cnr.it

CNUCE, Institute of National Research Council
Via S. Maria 36 - 56126 Pisa - Italy
Phone: +39-50-593111 (operator)
Fax: +39-50-904052

ABSTRACT

This paper proposes an architecture design for a tool suitable for emulating DA-TDMA (demand-assignment-time division multiple access) satellite access schemes. The tool presented, named FRACAS[1], is particularly suitable for comparing the performance of different satellite channel allocation policies. Using FRACAS, a service provider can choose from different policies for sharing a satellite channel among a number of users. Some allocation policies, selected from those available in the literature, are built-in, while others can be designed from scratch and added without much effort. The parameters of the built-in allocation policies can easily be changed in order to exploit the full potential of the allocation schemes. FRACAS's features permit the optimisation of satellite resource usage in accordance with the traffic pattern supported. FRACAS enables research teams and students to explore and compare different multiple access schemes, and to develop simulation runs for various kinds of service-induced traffic, including aggregate traffic, which is typical in a local area network (LAN) interconnection environment.

Keywords: emulator, simulator, satellite access schemes, performance comparison, TDMA, aggregate traffic

1. INTRODUCTION

Complex communication systems where satellite links are involved are difficult to test and tune up without the help of simulation tools. First of all, using satellites is very expensive - the satellite time spent in testing and tuning-up the system must thus be as short as possible. Second, during the performance evaluation in a real environment it is not always possible to find the right amount of traffic, and the most appropriate traffic

[1] FRAmed Channel Access Simulator

pattern and data aggregation that will put the system under the maximum amount of stress so that its limits can be validated.

In the last few years, many methods for accessing a data satellite channel in TDMA have been proposed in the literature, such as FODA/IBEA [7], CFRA [12], FEEDERS [10], DRIFS [11]. Older ones include SS-ALOHA [17], 2D-ALOHA [18], DAMA SCPC, FODA [19], C-PODA [20], F-PODA [20], R-ALOHA [21], RAN [22], CRRMA [23], CRIER [24], RACER [25, 26], CFDAMA [27, 28], BSA [29], ABCS-Dornier [30]. Few of these have actually been implemented as prototypes, and even less have come onto the market. It is therefore difficult to compare their performances, mainly because of a lack of a common testbed for the various solutions.

Our past experiences during the test phase of satellite access schemes in real environments [1] convinced us of the need for a simulation tool for evaluating the performance of satellite networks access schemes that would enable us to define the network topology, the traffic carried out by each individual station, the allocation request policy, the bandwidth allocation policy, and the statistics to be collected in order to evaluate the performance of such a network. Such a tool would facilitate changing the allocation policy in order to make comparisons and find the most suitable one. In order to avoid unnecessary approximations, it should be possible to specify exactly the mechanism of a given allocation policy, in order to get a precise emulation of a system's behaviour. Existing simulators are generally bulky and expensive. Moreover, most of them are general-purpose tools, so they do not exploit the particular structure of TDMA access methods.

We present the architecture of a high speed, lightweight emulator that is useful for simulating framed channel allocation schemes. We describe a proof-of-concept implementation, which is completely functional and easily extendible. The source code of our prototype implementation is freely available to students and researchers for downloading from:

ftp://fly.cnuce.cnr.it/pub/fracas.tgz.[2].

The prototype is an easily portable software package, written in C[3]. The allocation methods considered by FRACAS may have either a centralised or a distributed control. The stations receive data traffic from a number of different simulated sources, which are representative of the actual traffic. For example, our prototype implements the generation of fixed rate, Poisson, Markov modulated Poisson, fractal traffic and other traffic models, both with predetermined or freely selectable parameter values. Also, the architecture makes it possible to adapt the simulation time and the accuracy of results to the user's

[2] Users who download the FRACAS code are kindly requested to notify any of the authors of this paper.

[3] In our current implementation it runs under UNIX.

purposes, both for a preliminary network dimensioning, and when a precise exploration of the network performance is required. FRACAS produces some key performance figures, which are useful for studying the performance of an allocation policy, for example average and peak packet delay, traffic impulse delay, and packet loss due to queue overflow.

The rest of the paper is organised as follows. Section 2 presents an overview of some of the simulators available on the market. Section 3 describes the FRACAS architecture, with reference to our prototype implementation. Section 4 discusses the reliability and the accuracy of this simulator. In Section 5 we compute bounds to the discretisation errors. Section 6 describes some aspects of our prototype implementation. Section 7 concludes the paper with a summary and an outline of future work.

Technical details on the FRACAS prototypal implementation are not reported in this paper, but can be found in [14].

2. AN OVERVIEW OF NETWORK SIMULATORS

Most of the simulation tools which can be found in the literature are *discrete-event* simulators, i.e. they model a certain system as it evolves over time by a representation in which the state variables change only at a countable number of points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence which may change the state of the system. Let us examine some of the most popular tools.

OPNET[4] is a simulation tool for analysing communication networks by using models [2]. It is based on an extended finite state machine and is written in C. OPNET models are specified in terms of objects, each with configurable sets of attributes. These attributes can be specified either by a graphical process, an associated text file, or as variable parameters in the simulation description file. The specification of the models is organised into a hierarchy of four different levels: network, node, process, and parameter. When the model is specified, OPNET generates a simulation program written in C. Although C is not a particularly suitable language for modelling and simulation, the choice of UNIX/C guarantees the portability of the system.

BONeS[5] is a simulation system for studying communication network models [3]. BONeS SatLab is a flexible, special software package for the design, animated visualisation and analysis of satellite-based communication systems. SatLab can be used to model

[4]OPtimised Network Engineering Tool

[5] Block-Oriented Network Simulator

mobile/satellite systems, which may include satellites, fixed earth stations, and moving vehicles/persons. It provides multiple animated views of global satellite systems, uplink, crosslink and downlink analysis, jamming, and adjacent satellite interference analysis representation of fixed, mobile and portable earth stations. Three types of simulation can be performed: positioning, design and communication simulation. The positioning simulation lets the user model analyse and animate various configurations of satellites, fixed earth stations and moving stations. The design simulation automatically performs simulation for a range of parameter values. For example, the user can determine the interference of communication satellite systems on other communication systems or the probability of interference between satellite systems for selected frequencies. The communication simulation is used to track the source and route of data packets and to determine their best route based on relative distance, velocity, angle, visibility, traffic congestion, and interference between two or more nodes.

RESQ[6] is a software package developed at IBM Research for defining and solving extended queuing network models [4, 5]. Problems which have been analytically solved typically fall into the class of queuing systems for modelling the performance of computer communication systems. For the analytical solution of a queuing model, assumptions must be made about the system modelled. Typically these assumptions relate to the process of arrivals at the queues, the service processes at the queues, and the scheduling disciplines. RESQ provides a numerical solution component, QNET4, which uses the convolutional algorithm for product form networks, and a simulation component, named APLOMB. RESQ is especially strong in the statistical analysis of simulation outputs and the determination of appropriate simulation run lengths.

AMS[7] is an environment which integrates facilities and tools to build a communication system, to study its performance, and to validate the connectivity [6]. The user of the “atelier” can construct a concise system in a graphical environment and execute it, starting from models of several standard networks, such as LANs (Ethernet, Token Ring, FDDI), WANs (X25, TCP/IP), satellite (TDMA, FDMA) and radio-networks available in a specific library. AMS was designed on the basis of existing and proven packages, such as QNAP2, GSS (graphical support system), MODLINE and S-PLUS.

[6] RESearch Queuing package

[7] Atelier for Modelling and Simulation

3. ARCHITECTURE OF FRACAS

FRACAS was conceived in a research environment, with the goal of being highly specialised, and therefore efficient, so that it can be used even on CPU bound machines. A very essential but robust prototypal implementation exists, which has been used in research projects [9, 13]. As it can analyse the performance of demand assignment access schemes that have some sort of frame structure, it is well suited for satellite TDMA access schemes, which are always framed. Unlike the products cited in the previous section, FRACAS is a *discrete-time* emulator, so it is not based on events. This is the most important feature of FRACAS's architecture.
In our prototypal implementation, the time granularity is equal to the *frame duration*, which is the basic *time unit* of any action in FRACAS. This implies that everything that happens inside a frame appears to happen at the end of it, and time measurements with a resolution better than a frame time are not possible.
We argue that, when emulating TDMA allocation schemes, event-based simulation is an overkill because, most of the time, all that is needed to get reasonably accurate emulations is to study what happens at the end of each frame. In order to obtain such a behaviour, it would certainly be possible to use an event-based simulator to generate regularly time spaced events, one per frame, but this would mean that the event-based simulator would not be used efficiently.

The architecture of FRACAS is maximally efficient when the time resolution is equal to one frame. However, this choice is not limiting, therefore any specification of the desired time granularity is also possible for each emulation run. Although our prototypal implementation does not allow it, the time granularity could be made as small as desired with respect to the frame duration. This feature would be useful when, after having run a series of tests and having tuned the system to their liking, experimenters require and get the maximum possible precision from the simulation, by reducing the discretisation effects at the expense of running time.

FRACAS assumes that a frame of fixed duration is accessed by a number of stations in conformance with the allocations they have received. At each frame, each station receives a number of *traffic units* (TRU)[8] to be transmitted on the framed channel, then it makes a request for an allocation. When all the stations have made their requests, the allocations are computed. The frame where these allocations will be used by the stations is delayed with respect to the frame where the requests have been made by an *allocation delay*, which generally depends on the allocation policy. At each frame, each station queues the

[8]A TRU is the unit of measure of the traffic in FRACAS.

traffic units it has to transmit on the framed channel, and dequeues a number of traffic units equal to the allocation received (provided the queue does not get empty). Each pair *requester-allocator* defines a specific *allocation policy.*

In summary, at each frame the following actions take place inside FRACAS:

- for each station, a number of traffic units is generated, depending on the traffic pattern that feeds the station, and the input queue is lengthened accordingly;
- for each station, a request for allocation is made;
- all the allocation requests are analysed and the allocations for a future frame are computed;
- for each station, the allocation for the current frame, computed beforehand, determines how many traffic units can be transmitted, and the input queue is shortened accordingly.

3.1. Modelling a network with FRACAS

Modelling a network with FRACAS involves choosing an allocation policy, from those implemented, which defines the *allocator* and *requester* to be used. Then, for each station, a set of traffic *generators* must be chosen and, finally, a set of *workers* which will output the desired statistics. Figure 1 shows the logical flow of data used to compute the allocations inside FRACAS, starting from the data produced by the *generators*.

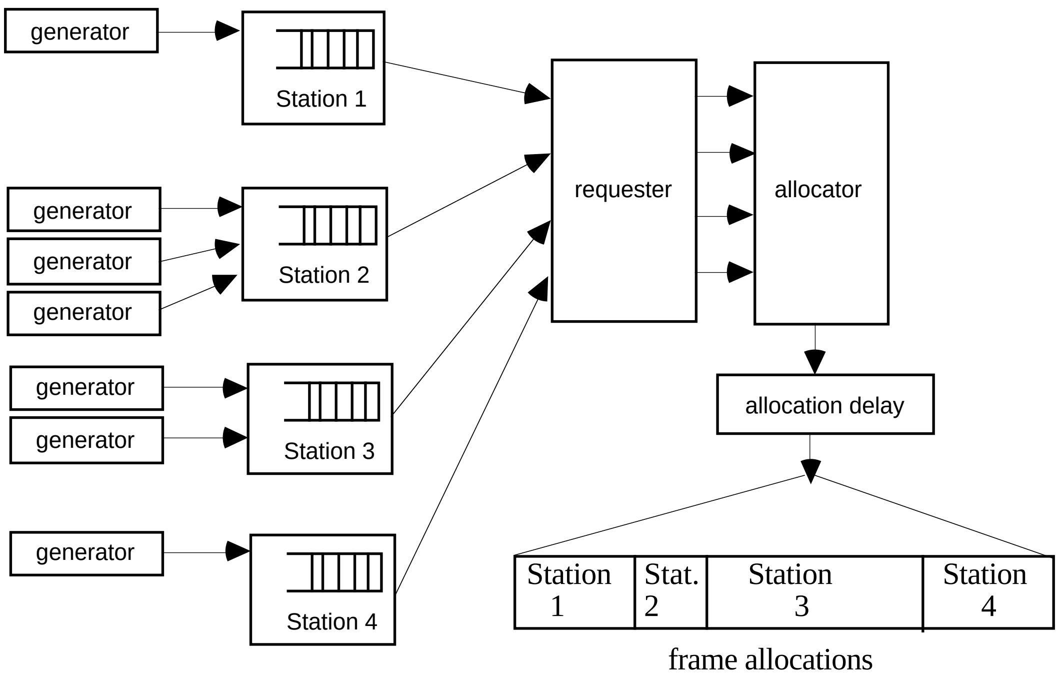

Fig. 1. Flow of data inside FRACAS

The traffic *generators* are responsible for computing the number of TRUs entering a station in each frame. All TRUs have the same length, which is chosen according to the granularity required. The output of a generator is the number of TRUs generated in the current frame. At each frame, a station queues the TRUs received from all its generators, and dequeues at most a number of TRUs equal to its allocation. The TRUs produced by a generator are collectively identified by how many they are, that is, they are not individually distinguishable. In order to queue them, the station at each frame just sums the current number of computed TRUs (queue length) to the number of TRUs received as input from the generators. Each station is fed by an arbitrary number of generators, each defined by individual generation parameters. In practice, a generator is a function that returns a single number (number of TRUs generated) each time it is called.

The *requester* is the object responsible for computing the number of TRUs requested by each station (station request) in each frame. The output of a requester is the number of TRUs requested in the current frame. There is only one requester for all the stations, that is, all the stations use the same criterion for making their request. At each frame, each station calls the requester in order to compute its allocation request. There is a specific requester for each allocation policy. An example is the *queue* requester, used mostly for testing, which issues a request equal to the length of the station’s input queue.

The *allocator* is the object, invoked once per frame, which computes the allocations for all the stations. It is invoked after all the stations have called the requester. The output of the allocator is an array of numbers representing the number of TRUs that will be allocated to each station after the allocation delay. The allocations computed are used after an allocation delay which is characteristic of the allocation policy used.

The *workers* are used to gather the statistics of interest. An arbitrary number of workers can be defined, each of which has a specialised job. When the emulator runs, at each frame many quantities inside the emulator can be probed in order to compute statistics on them. The candidate quantities are termed *observables*. An example of an observable is the number of TRUs per frame that is produced by the generators of an individual station. Each worker makes a single operation on the observables, such as listing them in a file, and computing their mean, variance, quantiles, mass distribution, and so on. All the observables can either be computed per station, or globally, considering the system as a whole.

The basic observables are expressed in *numbers of TRUs per frame* per station, and for each station it is possible to monitor the number of TRUs in *input* (that is, those produced by the generators), the number of TRUs *sent*, the available *allocation*, and the *queue* length.

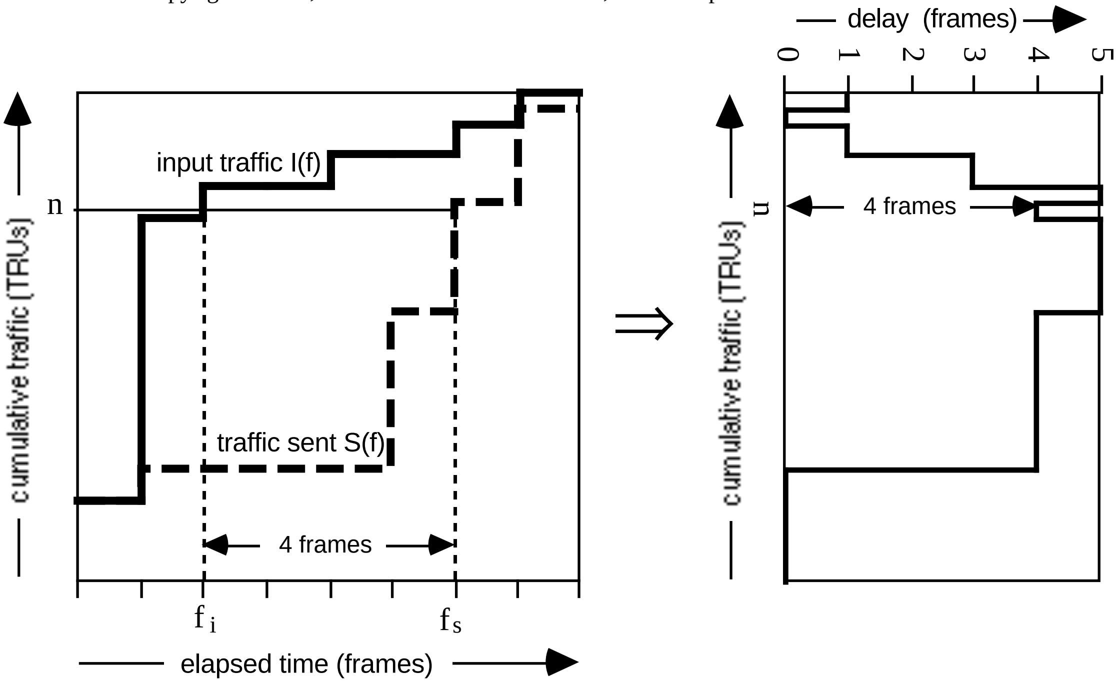

Fig. 2. Computation of the queuing delay.

The *delay* is the only observable which denotes a measure of time, expressed in numbers of frames. It is computed as the difference between the frame number when a TRU is generated and the frame number when it is sent (*queuing time*), plus the satellite round trip time. The *delay* observable is also special because it can be computed either per frame or per TRU. In the former case, for each frame a delay is computed which is the mean of the delays of the TRUs sent in that frame; in the latter case a delay is computed for each TRU sent.

Denoting by I(f) the cumulative number of TRUs in input to a given station at frame f, and by S(f) the cumulative number of TRUs sent by the same station, FRACAS computes the *queuing delay* of the n^th TRU (which we call Δ_n) as $\Delta_n = f_s - f_i$, where $S(f_s) = I(f_i) = n$. Figure 2 illustrates the concept. The leftmost graph shows the behaviour of cumulative input and output traffic. When the two lines overlap, there is no queuing delay, that is, traffic arriving at the station from generators is immediately sent, because there is enough allocation available for that frame. When this is not the case, traffic is enqueued and therefore it experiences a delay. The rightmost graph is depicted sideways in order to illustrate how it is computed from the leftmost graph. It depicts the delay experienced by each TRU. The scales of the axes labelled “cumulative traffic” are numbered in TRUs, so that each TRU produced by generators can be individually identified on that scale.

Figure 3 illustrates, with C pseudo code, the dynamic behaviour of the emulator and the role of the workers.

```
do {
    frame_number += 1;
    for_all_stations_do {
        input = run_generators (this_station);
        queue += input;
        sent = min (queue, allocation);
        queue -= sent;

        request = compute_request (this_station);
    }
    compute_allocations (frame_number +
allocation_delay);
    gather_statistics (frame_number);
} while (! stop_condition ());
run_workers_and_print_results ();
```

Fig. 3. The core of the emulator.

The core of the emulator is a loop which, for each frame, computes the traffic produced by the generators of each station, the length of the queues, and the traffic sent by the stations. It then calculates the allocations for a future frame and gathers the statistics needed by the workers. When the emulation ends, the workers elaborate the gathered data and produce their output.

3.2. Policies available

A number of *requesters* and *allocators* are included in FRACAS, which implement some allocation policies. The policies included in our prototype implementation are:

• fixed TDMA	a fixed assignment to each station;
• FODA/IBEA	a centralised control system developed at the CNUCE Institute in the framework of the Olympus project, validated by simulative and experimental results [7];
• V2L-DA	a centralised control system which supports variable bit rate video traffic [8];
• FEEDERS	a partially distributed control protocol derived from FODA/IBEA [9, 10];
• DRIFS	a fully distributed control protocol derived from FODA/IBEA [9, 11];

• CFRA a slotted centralised control allocation policy developed at ENST-Toulouse [12].

3.3. Output statistics

FRACAS can collect statistics of given quantities. Those quantities that are eligible for this purpose are called *observables*. As already mentioned, all observables are measured in the number of TRUs per frame, apart from the *delay observables*, which are expressed in *frame duration* units (the time length of a frame). Statistics on observables are available per station or globally. In our prototype implementation, observables include the traffic in input to a station (created by *generators*), the station's input queue length, the input data dropped because of the queue overflow, the requests made and the allocations given to a station, the traffic sent, the allocation unused, and the traffic delay (queuing plus round trip time).

Statistical calculations are performed by *worker* objects. In our prototype implementation, some workers are built in (listing, mean, variance, percentile, distribution), and others can be added as an external program. This feature gives FRACAS great flexibility, as it is easy to interface it with external statistic packages.

3.4. Types of traffic

In the most common classification, access methods provide different services for *stream* traffic and *bursty* traffic. According to the traffic categories as defined in the ATM Forum TM4.0 ("ATM service categories") [31], the former type of service is intended for real-time, fixed-throughput applications, such as telephony or videoconferencing which are coded at a constant bit rate (CBR) and which require a guaranteed bandwidth and a fixed delay. Recently, the need for the transmission of variable bit rate (VBR) video traffic has emerged. This kind of traffic has some real-time requirements, and is very bursty, thus requiring allocation policies that are different from both CBR and bursty traffic. The second class includes all the jitter-tolerant applications, i.e. unspecified bit rate (UBR), and available bit rate (ABR) service categories. This latter class is intended for best-effort traffic, such as the one flowing between interconnected LANs. Bursty traffic typically has no hard real-time requirements but the burstiness of the sources requires sophisticated allocation methods in order to make efficient use of the expensive satellite bandwidth. Bursty traffic can be further classified into *interactive* and *bulk* traffic. Although interactive traffic typically represents only a small fraction of the total bursty traffic, it nevertheless requires particular attention because it is much burstier than bulk traffic, and requires small and possibly constant delays. In fact, sites which use satellite links for interconnection often generate all these types of traffic.

A DA-TDMA satellite access scheme is then faced with a great variety of traffic types, whose interaction is usually very complex.

The FRACAS architecture assumes that all the traffic inside the network is divided into a fixed number of classes. Our prototype includes four classes, namely *stream*, *vbr*, *interactive*, and *bulk*. The names of the classes reflect the type of traffic they are supposed to be used for. According to the traffic classification previously introduced, we think that most categories of traffic are represented in the four classes implemented in FRACAS, but other classes of traffic can be added if necessary**.** All the parts of FRACAS keep different structures for each of the four classes of traffic. The input queue of each station is thus composed of four counters, the request issued by the requester is composed of four numbers, and the allocations computed by the allocator are expressed by four numbers per station. There are observables for each class so, for example, the mean value of the VBR input queue at a specific station can be computed.

The reason for having different classes of traffic is that the allocation policy can treat them differently. Moreover, the *generators* may be different for each traffic class. The FRACAS engine makes it natural to define a hierarchy on the classes of traffic, which is used when the station's input queues are shortened according to the allocations received. In our prototype, *stream* has the highest priority class, so the *stream* allocation is subtracted from the *stream* input queue first. Then, if the *stream* allocation is not entirely used, an *extra space* is added to the *VBR* allocation (the second highest priority class). The *VBR* allocation thus updated is then subtracted from the *VBR* input queue. If the updated *VBR* allocation is not entirely used, an extra space is added to the *interactive* allocation, and so on. The *bulk* traffic has the lowest priority. Each station maintains the stream traffic allocation constant for the entire duration of the allocation, while the allocator assigns a VBR allocation between a minimum and a maximum value. Interactive and bulk traffic receive allocation amounts which depend on the allocation policy chosen. This hierarchy of priorities on the classes of traffic guarantees that the different delay characteristics of each type of traffic are respected for the protocols currently implemented. From the FRACAS designer's point of view this hierarchy can be easily changed or rearranged according to possible new allocation policies (for example, a new hierarchy where some classes have the same priority). Once the data hierarchy has been chosen, the user is not allowed to change it from the input file which defines most parameters used in FRACAS.

3.5. Modularity and interface with external programs

As already mentioned, allocation policies can be built into FRACAS by writing standard C code routines. It is then possible to test new allocation policies with any number of stations, each one fed by an individual traffic pattern. The implementation of a new allocation policy requires some work, but the ones that are part of our prototype implementation are good starting points, so it is possible to use them as templates.

Traffic generators and workers can be incorporated inside FRACAS, or they can be built as external applications. Indeed, there is one ad hoc generator (*external generator interface*) which just reads from an external program (or file) a list of numbers, which are interpreted as the number of TRUs generated per frame, and there is one worker (*external worker interface*) which writes a list of observables to an external program (or file). This feature is particularly useful if other programs already exist that can generate numbers representing traffic or that can make particular statistical computations on lists of numbers.

Our prototype implementation features a number of built-in generators, including constant-bit rate, Poisson, periodic impulsive with constant rate or Poisson bursts, Markov-modulated impulsive, and fractional Gaussian noise.

4. VALIDATION OF THE PROTOTYPE IMPLEMENTATION

The prototype emulator was tested during its development in order to validate it. This phase is very important and delicate because if the developed tool does not give a valid representation of the system under study, or the user cannot be confident about the data produced, no useful information about the actual system is given. The goodness of a simulation model is measured by the closeness of the model output to that of the real systems. We had the opportunity to compare the results obtained transmitting data with a real system realised in the framework of the Olympus satellite utilisation program [1], with those obtained using FRACAS. The comparison of the packet delays, for a particular traffic pattern feeding the station used in the test, showed no significant difference between the plots obtained using FRACAS and those obtained using the real system (Figs. 4-6).

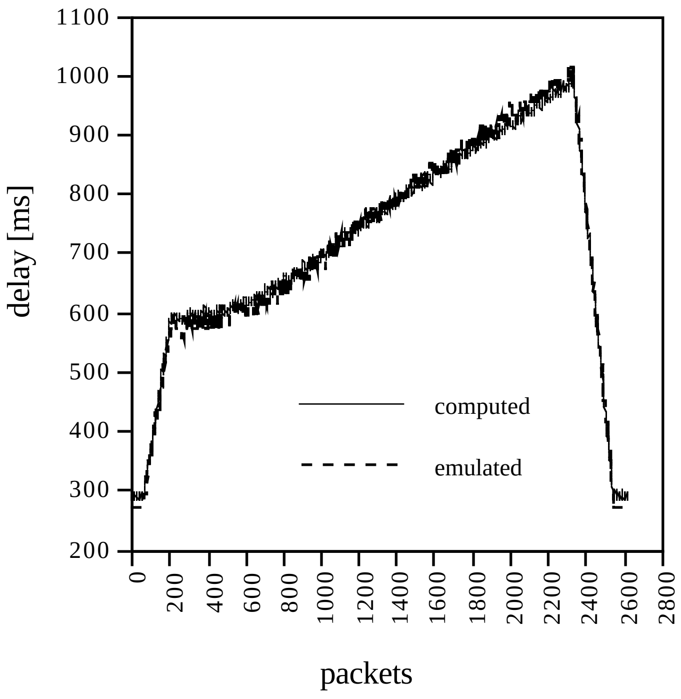

Fig. 4. Packet delay of a station during a transient of the traffic load. Measurement and emulation results.

The traces refer to an individual station whose input is loaded with a traffic step with a constant rate higher than the bandwidth of the satellite channel. The allocation method used in this test is FODA/IBEA, and the test run lasts for 7s, for a total of 80,000 packets generated. The real system data are measured using MTG [15], a traffic generator and measurement system that we developed for testing satellite access schemes, while the emulated data was obtained by running FRACAS on an IBM Risc 6000 530H, using 0.15s of CPU time. While both the prototype emulator and the real system use the same allocation algorithm, the results are not identical, because the real system uses many implementation and hardware-dependent tricks, which are difficult to reproduce exactly in FRACAS. Moreover, the resolution of the emulated packet delay is one frame (20 ms in this case), which introduces a further error with respect to the measured delay. This granularity error on the packet delay will be discussed in Section 5. These effects notwithstanding, the emulated traces reflect the measurements well, considering that we are comparing an emulator with a real system, not merely a mathematical formula. The relative errors are in the range of [-5%; 5%] for the packet delay, and in the range [0-7%] for the queue length. It should also be pointed out that this test was chosen because it is particularly severe, in that it highlights possible problems in the emulator. Indeed, the traffic that feeds the station saturates it, that is, the station is never allocated enough capacity to empty its input queue. In such a situation all the emulation errors accumulate.

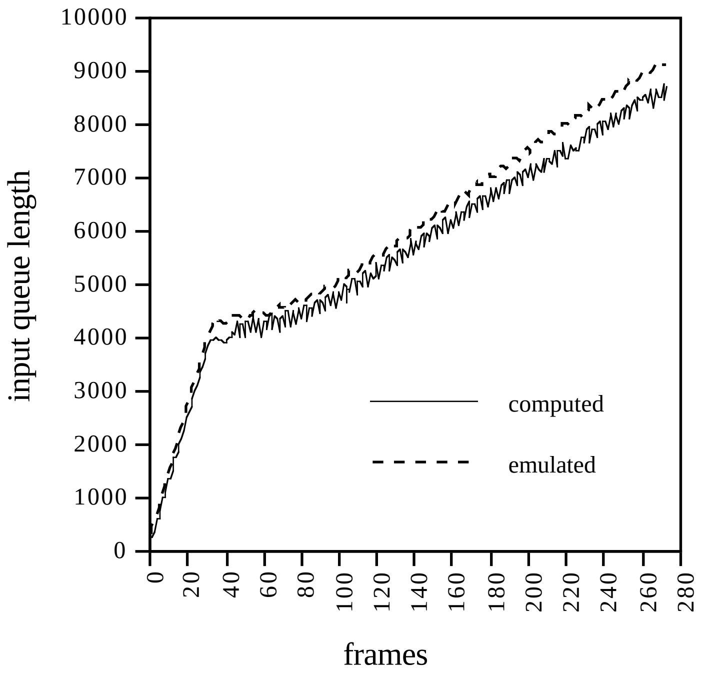

Fig. 5. Input queue length of a station during a transient of the traffic load. Measurement and emulation results.

Figure 6 shows another test comparing the performance of a real network composed of four stations running the FODA/IBEA protocol, and the emulated behaviour. All stations are fed with Poisson traffic, with stations from 1 to 4 receiving 42%, 32%, 20% and 6% respectively of the overall network load. Three runs are performed, with three different overall network loads, up to a load of 85%, which saturates the allocation method and results in queues at station 4. The real data are collected over periods of 30 seconds (1500 frames), using MTG, for a total of 360s (18,000 frames) and 270,000 packets, and the emulation used 35s of CPU time. Even with this complex set-up and the rather short running period, the relative differences between measured and emulated delays are in the range of [-6%; 2%], with only one point where the difference exceeds one frame length (20 ms). Once more, we stress the fact that we are comparing a software tool with a real network, and we argue that the differences are remarkably low.

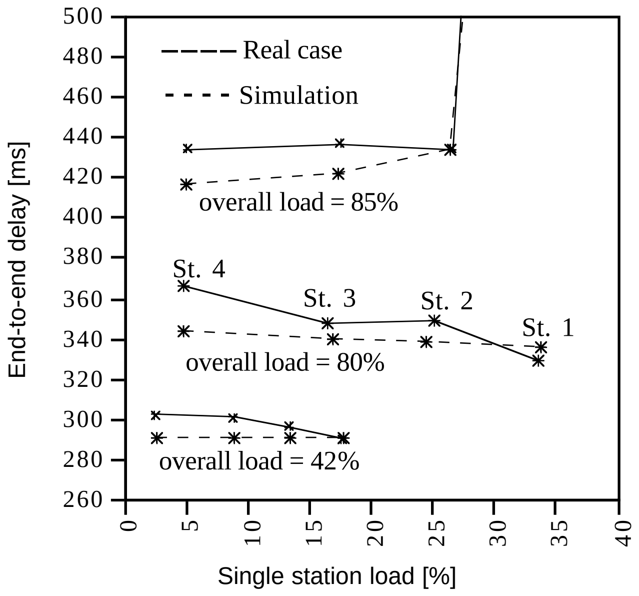

Fig. 6. Mean packet delay for four stations under medium and high loads. Measurement and simulation results.

5. ERRORS DUE TO TIME DISCRETISATION

FRACAS is a *discrete time* emulator with a granularity of one frame. Since all the timings are multiples of one frame, no delay shorter than this quantization unit can be resolved. This implies, for example, that null delays in a real system are rounded up to a half frame delay. The delays obtained with FRACAS generally have an error in the range of [-1; 1] frames. This means that the mean packet delay has a maximum error of 1 frame, though the error is effectively halved for emulation runs of any significant length. Let us examine this issue in closer detail.

When FRACAS runs an emulation with a time resolution set to one frame, as in our implementation, its concept of time is limited to the current frame number. The traffic generators in FRACAS compute how many packets are generated in the current frame, but no knowledge exists either about the exact instant in the frame when each packet is generated or about the instant in the frame when the packet is sent. FRACAS's idea of delay in this case is that, if a packet is generated in a frame (i.e. the packet arrives at the station) and there is an allocation available after N frames (i.e. the packet leaves the station), the packet has a delay of N frames. In reality, the packet's delay would lie in the range of [N-1; N+1] frames, depending on the point in the frame when the packet arrives and the point when it leaves.

Figure 7 illustrates this point. Only the queuing times are considered, as the delays are simply computed by adding the round trip time to the queuing time. In both cases a) and b), FRACAS computes a delay equal to three frames, because the packet leaves three frames after the frame where it has arrived. However, the actual packet's delay ranges from 2 to 4 frames, because FRACAS has no notion of where in the frame the packet arrives or leaves.

Thus each packet's delay, as computed by FRACAS, has an error in the range [-1; 1] frames with respect to the emulated system. The *delay error* is the sum of two errors: the error on the position of the arriving packet in the frame (*arriving error*), and the error on the position in the frame of the leaving packet (*leaving error*), each in the range [-0.5; 0.5] frames. We will show that the former becomes negligible in practical emulation runs, so the total error is limited to the range[-0.5; 0.5] frames.

The leaving error is not easily estimated, because it depends on the position of the allocation inside the frame, which can have any distribution (and thus any mean) inside the range. Consider, for example, a fixed allocation policy, where each of *N* stations gets 1/N of the frame space, always in the same order (station 1 first, station 2 second, etc.). The packets transmitted by station 1 always arrive at the beginning of the frame, and the uncertainty on their position inside the frame is limited to the width of the allocation itself. The arriving error thus lies in the range [0.5-1/*N* ; 0.5] frames. In this case, the range where the leaving error lies is reduced by a factor *N*. For a given allocation policy, it may be possible to compute a distribution of the allocation position, and thus get an estimate of the range of the leaving error which is more accurate than the generic [-0.5; 0.5] frames, as in the case of a fixed allocation. However, this is generally not possible.

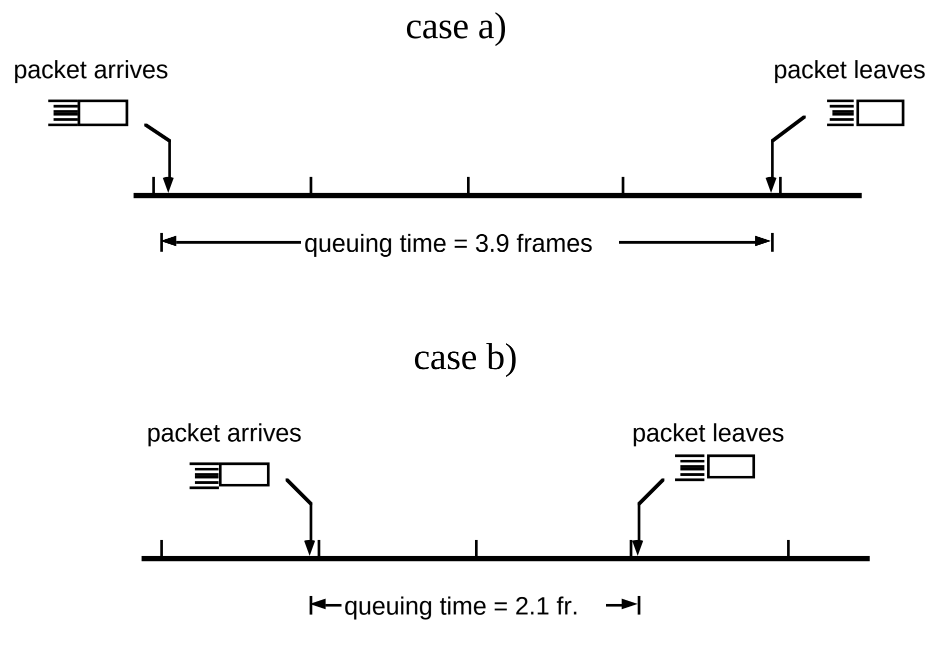

Fig. 7. How the positioning of the packet inside a frame affects the delay measurement error.

The arriving error, on the other hand, can be estimated with high accuracy if we assume that the instants when the packets arrive have no correlation with the frame repetition period, which is generally the case. Under this assumption, a packet's delay error has a uniform distribution in the error range. If we assume an emulation run lasting a number of frames k (say, k »100), the error of the mean is well approximated by a Gaussian distribution with a null mean and variance equal to $\frac{1}{12k}$ frames, which is negligible for any practical purpose. For example, in an emulation run lasting 10000 frames, the 99.7% confidence interval of the mean of the packet delay would be less than ±1% of the frame size.

Thus, for all meaningful emulation runs for which the packet generation rate is not synchronised with the frame rate, we can assume that the error of the mean delay computed by FRACAS has an error in the range [-0.5; 0.5] frames, whose distribution depends on the allocation policy.

6. SOME ASPECTS OF OUR PROTOTYPAL IMPLEMENTATION

The main strengths of the FRACAS architecture are its speed, extendibility, and the existence of a working prototype whose portable source code is freely available to students in the field of satellite framed allocation methods. The prototype has knowledge of leading-edge satellite allocation methods, and we hope that researchers will contribute to extend its knowledge, so that a common testbed for this kind of access methods can be made available to the scientific community. Work is in progress towards enabling the prototype to run independent replication tests, in order to accurately assess the reliability of the results obtained.

Another area of improvement will be the implementation of time resolutions smaller than one frame. Although we believe that smaller time resolutions generally add nothing to the significance of the results, it may be important in some corner cases, where the behaviour of the emulated system appears to be unstable, and changing the time resolution may provide deeper insight into its response to traffic loads.

To get an informal feeling of the performance of our prototypal implementation of the FRACAS architecture, we ran the same emulation test using both FRACAS and OPNET. The choice of OPNET for the comparison rather than other simulators (such as BONeS,) was dictated by the fact that an OPNET version is already installed in our laboratory, and we used it in a comparison study between different access schemes. The test was a simulated 2000 seconds run of a complex traffic generator configuration feeding 16 ground stations which access a common satellite channel using CFRA [12] as the allocation policy. The duration of this test yields a 95% confidence interval of ±7% on the packet delay. The OPNET sources were written by researchers at the ENST-Toulouse, and the FRACAS sources by one of the authors of this paper in the framework of some research on the relative methods of different satellite access schemes [13]. The double platform on which the simulation was run was intended to make the emulation results more reliable. Both programs were run on an IBM Risc 6000 530H workstation, with very similar results.

OPNET ran the emulation in 3h 10m 8s, while the FRACAS prototype ran it in 1m 29s 33. The ratio of speeds is a massive 128:1 in favour of FRACAS. Such a great difference of speed can easily be explained by looking at the number of "elaboration units" corresponding to the same run for the two programs: while OPNET, a discrete event simulator, generated 23,700,000 events, which is an average of more than three events per packet, FRACAS computed 68,900 frames, each containing an average of about 100 packets. From another point of view, OPNET generated 2700 events per second, while the FRACAS prototype computed 771 frames per second. These are comparable numbers, and are certainly more comparable than the run times.

We don't claim any generality for the above result, because by no means can the described test be considered as a true performance comparison. Moreover, it must be pointed out that OPNET is a commercial product and is far more flexible than our prototype, which is an academic program confined to framed access schemes. Indeed, since OPNET is a discrete event emulator, it follows each packet individually, allowing a potentially greater precision and detail of the resulting statistics.

7. CONCLUSIONS AND FUTURE WORK

We have presented the architecture for a performance evaluation tool, which is especially suitable for research teams and students to explore and compare different satellite access policies in TDMA. The strengths of this tool are: a) its execution speed; b) its knowledge of allocation methods which are the subject of current research; c) the free availability of a proof-of-concept implementation.

We are investigating the possibility of augmenting the FRACAS architecture to enable it to consider the behaviour of allocation policies when faced with severe casualties, such as signal fades. This would allow the analysis of recovery from disastrous events (power and component failure) or poor signal reception (deep signal fading).

REFERENCES

[1] N. Celandroni, E. Ferro, F. Potortì, A. Bellini, F. Pirri: "*Practical experiences in interconnecting LANs via satellite*", ACM SIGCOMM Computer Communication Review, Vol. 25, No. 5, October 1995.

[2] OPNET Tutorial Manuals. MIL3, Inc.

[3] K. Sam Shanmugan: "*BONeS Designer: Introductory overview*", COMDISCO Systems, Inc.

[4] Sauer C.H., Mac Nair e.a.: "Queuing network software for systems modelling", Software-Practice Exp. 9, 5, (1978).

[5] Sauer C.H., Mac Nair E.A., Kurose J.F.: "*The research queuing package: past, present and future*", Proceeding of the National Computer Conference AFEPS, Arlington (Virginia), USA, 1982.

[6] A. Cohen, R. Mrabet: "*An environment for modelling and simulating communication systems: application to a system based on a satellite backbone*", International Journal on Satellite Communications, Vol. 13, 1995, pp. 147-157.

[7] N. Celandroni, E. Ferro, N. James, F. Potortì: "*FODA/IBEA: a flexible fade countermeasure system in user oriented networks*", International Journal of Satellite Communications, Vol. 10, N. 6, pp. 309-323, November-December 1992.

[8] N. Celandroni, M. Conti, E. Ferro, E. Gregori, F. Potortì: "*A bandwidth assignment algorithm on a satellite channel for VBR traffic*", International Journal on Satellite Communications, Vol. 15, No. 6, pp. 237-246, November-December 1997.

[9] N. Celandroni, E. Ferro, F. Potortì: "*Comparison between distributed and centralised demand assignment TDMA satellite access schemes*", International Journal on Satellite Communications, Vol. 14, No. 2, pp. 95-112, March-April 1996

[10] N. Celandroni, E. Ferro, F. Potortì: "*FEEDERS-TDMA: a distributed-control algorithm for satellite channel capacity assignment in a mixed traffic and faded environment*", International Journal on Satellite Communications, Vol. 15, No. 4, pp. 185-195.

[11] N. Celandroni, E. Ferro, F. Potortì: "*DRIFS-TDMA: a proposal for satellite access distributed control algorithm for multimedia traffic in a faded environment*", International Journal on Satellite Communications, Vol. 15, No. 5, pp. 227-235, September-October 1997.

[12] T. Zein, G. Maral, T. Brefort, M. Tondriaux: "*Performance of the Combined/Fixed Reservation Assignment (CFRA) scheme for aggregated traffic*", Proceedings of the COST-226 Final Symposium, pp. 183-198, Budapest (H), 10-12 May 1995.

[13] N. Celandroni, E. Ferro, F. Potortì, G. Maral: "*Delay analysis for interlan traffic using two suitable TDMA access schemes*", International Journal on Satellite Communications, Vol. 15, No.4, pp. 141-153.

[14] N. Celandroni, E. Ferro, F. Potortì: "*FRACAS: FRAmed Channel Access Simulator. User Manual*", CNUCE Report C95-27, September 1995.

[15] N. Celandroni, E. Ferro, F. Potortì: "*A traffic generator for testing communication systems: presentation, implementation and performance*", Real-Time Systems, Vol. 13, no. 1, Jul. 1997.

[16] Olympus Users' Guide UG-6-1 Part 4: “*20/30 GHz Communication Payload*”, CCB/52182/HHF/CMM, Issue 3, February 1988.

[17] S.N. Crozier: “*Sloppy-Slotted ALOHA*”, proceedings of the International Mobile Satellite Conference, Ottawa, 1990, pp. 357-362.

[18] Gilat product brochures.

[19] N. Celandroni , E. Ferro: “*The FODA-TDMA satellite access scheme: presentation, study of the system, and results*”, IEEE Trans. Communications, vol. COM-39, pp. 1823-1831, Dec. 1991.

[20] I.M. Jacombs, R. Binder, E.V. Hoversten: “*General purpose packet satellite Networks*”, proceedings of the IEEE, vol. 66, no. 11, pp. 1448-1467, Nov. 1978.

[21] S.S. Lam: “*Packet broadcast networks - A performance analysis of the R-ALOHA protocol*”, IEEE transactions on Computers, vol. C-32, no. 8, pp. 717-726, Aug. ‘83.

[22] C.J. Wolejsza Jr., J. McCoskey, D. Taylor: “*The Random Access with Notification Multiple Access Protocol: performance and implementation*”, ICDSC 8, pp. 131-138, Jan. 1989.

[23] H.W. Lee, J.W. Mark: “*Combined Random/Reservation Access for packet switched transmission over a satellite with on-board processing: Part I - Global beam satellite*”, IEEE Transactions on Communications, vol. COM-31, pp. 1161-1171, Oct. 1983.

[24] Tho Le-Ngoc, Yu-Dong Yao: “*CRIER, a multiaccess protocol for on-board processing satellite systems*”, Canadian Conference on Electrical and Computer Engineering, pp. 26.1.1-26.1.4, Sept. 1991.

[25] “*DOC Advance Satcom Mission system concept and hardware definition program., Task 1 - Final report*”, Vol. 2 Architecture Derivation, Spar Program 3300-C, Issue A, Sep. 1992.

[26] “*DOC Advance Satcom Mission system concept and hardware definition program., Task 1 - Final report*”, Vol. 3 Simulation and Modelling, Spar Program 3300-C, Issue A, Sep. 1992.

[27] Tho Le-Ngoc, Jahangir I Mohammed: “*Combined Free/Demand Assignment Multiple Access (CFDAMA) Protocol for packet satellite communications*”, 2nd IEEE International conference on Universal Personal Communications, ICUPC'93, Ottawa 12-15, 1993.

[28] Tho Le-Ngoc, V. Krishnamurthy: “*Performance of Combined Free/Demand Assignment Multiple-Access schemes in satellite communications*”, International Journal of Satellite Communications, Vol. 14, 11-21 (1996)

[29] S. Agnelli, A. Tonoli, V. Trecordi: “*An efficient bandwidth-sharing technique for a satellite ATM user-oriented network*”, proc. ICCS ‘90, pp. 4.1.1-4.1.5, Nov. 90.

[30] Dornier Deutche Aerospace: “*Advanced Business Communications via satellite*”, System Description, July 92.

[31] The ATM Forum: “ATM service cayegories: the benefits to the user”, White paper, European Market Awareness Committee.