A SIMULATION SYSTEM FOR ATM AND PACKET-BASED MEO AND GEO COMMUNICATION SYSTEMS

Hartmut Brandt
AIAA member,
Fraunhofer Institute for Open Communication Systems (Fokus)
Berlin, Germany
brandt@fokus.fraunhofer.de

Falco Krepel
AIAA student member
Fraunhofer Institute for Open Communication Systems (Fokus)
Berlin, Germany
krepel@fokus.fraunhofer.de

In this paper we present a simulator for GEO or MEO-based communication systems. This simulator implements variable delay and bit error rate simulation and can be used to analyze communication scenarios involving one or more GEO/MEO or LEO spacecrafts and ground stations. First the paper gives a short overview over the original LEO simulator design from which the GEO simulator was derived. Then it shows the modifications needed for the GEO/MEO simulation, provides a detailed description of the cell and packet processing, together with a detailed description of the simulator structure and measurement results. At the end we present the results of a usage case of ATM-based communication between the ISS and an earth station via a GEO satellite.

INTRODUCTION

Communication systems based on Medium Earth Orbit (MEO) and Geostationary Earth Orbits (GEO) offer a viable alternative to systems using Low Earth Orbits (LEO). They are able to augment LEO and terrestrial systems to offer wide-range, high-capacity and low-cost communication to areas missing the terrestrial infrastructure and to in-orbit objects on lower orbits (for example the International Space Station ISS). GEO systems also may be used for some kinds of communication on their own. While their use for interactive real-time applications is of rather questionable value because of the large delay that is involved, they are able to deliver larger bandwidth to a wider area and more users than LEOs. With these characteristics they are ideally suited to non-realtime applications and data transfer especially for Internet access and bussiness-to-bussiness data communication. For both the communication system designers and the applications designers who use these systems it is essential to prove their designs prior to the launch of the cost-intensive satellites.

Depending on the concrete communication system and the goal of the simulation different simulation types may be necessary. Three effects are off special interest because they substantially differ from terrestrial communication systems:

- Simulation of the delay. Depending on the spacecraft orbits, the number of involved spacecraft and the physical and medium access control layers the delay and the delay jitter may be substantially larger than in terrestrial networks. GEO satellites introduce a one-way delay (ground to satellite) of approximately 120 ms. If inter-satellite links or several hops are involved the delay may sum up to more than 0.5 s. Whith MEO or LEO satellites, periodic changing delays occur and when framed MAC layers and turbo codes are used large delay jitters are to be expected. All these delays and delay jitters have impact especially on real-time applications and on transport protocols.
- Simulation of link errors. Satellite links, especially in the higher frequency bands (Ka and above) are subject to whether influences. Rain and snowfall will introduce errors. Links to MEO and especially LEO satellites are also problematic because of the tight link margin (because of moveable antennas and smaller, low-power satellites). The internet transport protocol TCP was designed with the assumption that packet loss occurs because of network congestion. In satellite networks loss occurs because of link errors, so TCP may perform badly under these circumstances. Link error simulation is also needed to tune the performance of multimedia transmissions (different video and audio have different sensitivity to transmission errors) and protocols like ATM signalling.
- Simulation of shadowing effects. This is mainly needed for LEO and MEO access networks with mobile users. For these cases steerable antennas are

needed and the view to the satellite may become obstructed by buildings and trees.

The goal of the ATMSat project[1] sponsored by the German Bundesministerium für Bildung und Forschung was the development of an architecture and some key technologies for a LEO satellite multimedia communication system. One result of this project was a modular and extensible simulator and rapid protocol-prototyping environment for LEO communication systems[2]. This simulator implements all the mentioned effects. The simulator is accessible remotely through the internet via HTTP, SNMP and secure shell connections.

For a project that involves a high speed communication link to the ISS the simulator has been enhanced to simulate GEO and MEO satellites. The main changes are the support of much larger delays and higher data rates.

ATMSAT LEO DEMONSTRATOR

The ATMSat demonstrator[2] provides an environment to prototype, test and evaluate medium access layer (MAC) and signalling protocols. It is build based on standard PC and ATM equipment as shown in Figure 1.

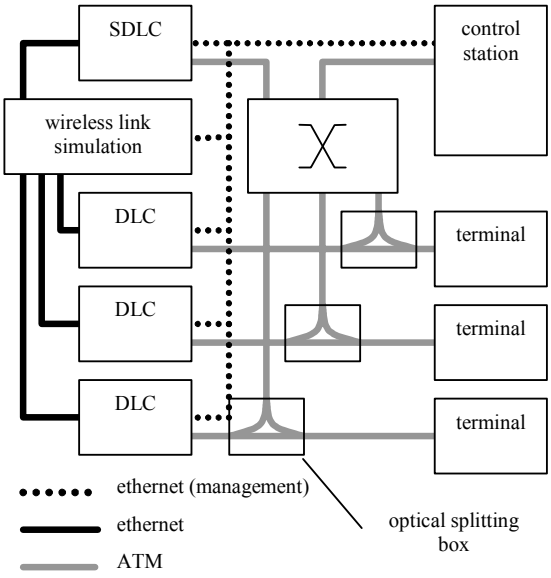

Figure 1: LEO demonstrator architecture

The demonstrator consists of four DLC (data link control) systems, one wireless link system, two or three application terminals, one control station, and an ATM switch.

The ATM switch is the center of the demonstrator. It provides the cell switching functionality, but not the signalling. The signalling stack runs on a workstation (control station) and controls the switch via SNMP over LAN emulation from this control station. The ATM switch can also be used to route monitoring traffic from a number of optical splitting boxes to either the control station or ATM measurement equipment.

The wireless link system is used to simulate the wireless link. It consists of a PC with a four-port 100Mbit/s full duplex Ethernet card. Each DLC system is connected to one port of this Ethernet card. The ethernet links are used as pure data transport vehicles. Because each of the links is a dedicated point-to-point link, no collisions and no unexpected delays are likely to occur.

The simulation assumes a framed MAC layer (Figure 2). Uplink and downlink frames have a size of 24 ms. Each downlink frame starts with the time-burst plan (TBP) that assigns uplink frame resources to the individual terminals. The rest of the frame contains ATM cells each of which has an additional 20bit MAC layer address and empty space. The uplink frames are split in two areas: the normal terminal burst area and the contention area. The splitting point between these areas is variable and is defined in the TBP. The burst area is filled by the terminals with bursts according to the TBP. Each bursts starts with a MAC layer minislot that is used for MAC layer signalling and contains the MAC address followed by zero or more standard ATM cells. The contention area may be used by terminals that have no burst allocated to it to reach the satellite. The burst area is collision free, while in the contention area collisions may occur.

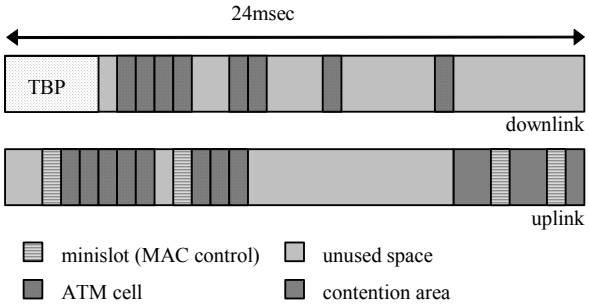

Figure 2: MAC layer frames

In the simulator the TBP, downlink cell bursts, uplink bursts and uplink contention minislots are encapsulated into ethernet frames. Additional there is a synchronisation frame just before the TBP that is used to synchronize the systems.

The simulation process consists of several steps:

- Received packets are timestamped as soon as possible and queued.
- Downlink packets are copied for each of the ground terminals.
- For packets in the contention area: if packets overlap they are dropped and a collision is counted.
- For packets in the uplink burst area: if packets overlap an error is signalled. This generally shows a protocol or implementation error.
- By means of a random number generator and a table with the time dependend error rates errors are introduced into the packets.
- By means of a random number generator and time dependend markov transition probabilities shadowing is introduced (packets are dropped if the satellite view is obscured).
- Packets that have not been dropped are output depending on the current (per terminal) delay and their arrival time.

The MAC protocols are implemented on the DLC (data link control) systems which can be configured to run either S-DLC (satellite DLC) or T-DLC (terminal DLC). Because of this flexibility it is possible to simulate scenarios with different numbers of spot beams and satellites. The protocol implementation is separated and independent from the terminal platform and therefor it is possible to rapid-prototype different MAC layer protocols without affecting the terminal. The DLC system is connected to the terminal with ATM and to the link simulator with a 100MBit/sec ethernet. The DLC systems are standard PCs equipped with two ethernet cards and one ATM card.

The application terminal (terminal) represents an endsystem which communicates over the satellite with another endsystem. The terminal can be any platform (e.g. FreeBSD, Solaris, Windows, Linux) with an ATM interface.

The control and monitoring station is used to boot all other systems and to control, configure and monitor the simulator. It also hosts a web server that can be used to remotly access the simulator. The structure of the control station is shown in Figure 3 and has the following components:

- an NFS and a bootp server that are used to boot the diskless DLC and WLINK systems.
- an SNMP client that connects to all the SNMP servers in the DLC and WLINK systems and translates HTTP requests to SNMP requests and SNMP responses to HTTP responses.
- a number of CGI and perl scripts that implement a web interface to the simulator and uses the above translator.
- command line tools that use SNMP to control the simulator. This may be used to script simulations.
- the ATM signalling daemon. This is implemented as a module for the SNMP daemon and connects to the ATM switch via SNMP over LAN emulation.
- a control tool for ATM and signalling that uses SNMP.
- a number of ATM monitoring tools like protocol tracer.

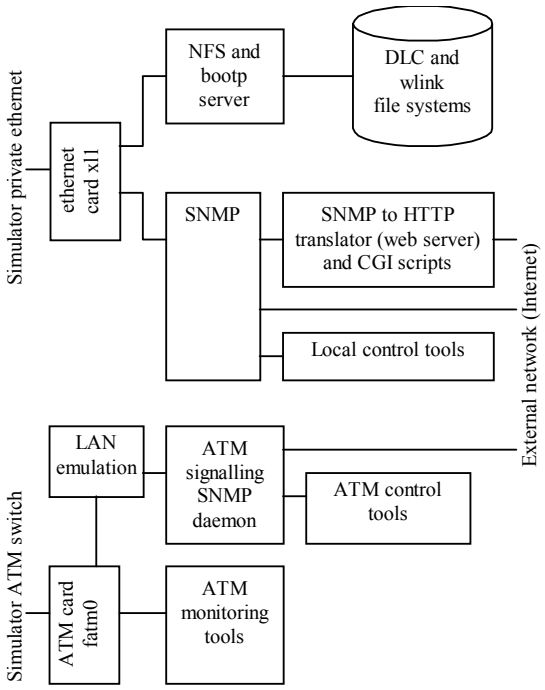

Figure 3: Control station structure

The simulator and prototyping environment has successfully been used to implement the ATMSat MAC protocol and demonstrate a number of Internet and native ATM multimedia applications over the simulated network.

GEO/MEO SIMULATION

For the GEO/MEO simulation only parts of the original simulator are used. The idea is that ATM cell streams are generated and received by equipment outside of the simulator (ATM tester, workstations with ATM cards). The simulator just simulates the link(s) between the two

end points of the communication. This results in the configuration shown in Figure 4.

The simulator consists of the wireless link simulation system, an ATM switch and the control station. Strictly speaking the ATM switch is not necessary, but very helpful if the physical layers used by the external ATM systems and the wireless link simulation are different or the ATM cell streams are to be monitored on the control station.

The link simulation system is a standard PC equipped with two ATM interface cards and one ethernet card. This PC simulates to link(s) between the communication endpoints. The external cell generation equipment is connected to these two cards. This wireless link emulation PC is controlled by the control station PC through SNMP over the ethernet between the control station and the wireless link emulation. Optionally the simulator may be equipped with two optical splitting boxes, an ATM switch to merge the output of these boxes and an additional ATM card with time-stamping capability in the control station (or, for example, an ATM tester). This additional equipment may be used to make high precision timing measurements and monitor the traffic on the link.

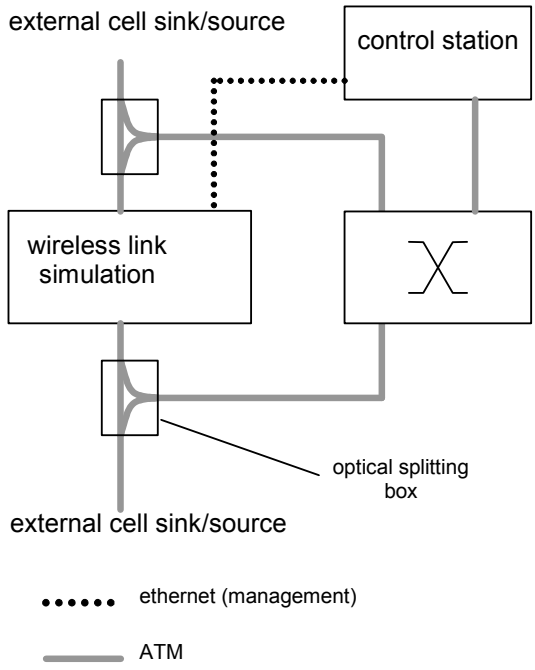

Figure 4: GEO/MEO demonstrator architecture

It must be noted, that, compared to the LEO simulator, no additional equipment is needed—the simulator's hardware is simply reconfigured for the new task and most of the software can be reused without changes.

Wireless Link Simulation

The wireless link simulation PC runs a special tuned FreeBSD kernel[3] with a kernel module that simulates the link and an SNMP daemon for remote configuration (Figure 5).

Uplink traffic is received by the ATM driver on hatm1. It is handed over to the netgraph node `hatm1:` which sends the cells to the `watm:` node. This node inserts the appropriate delay and drops errored cells and sends the cells along the hook to node `hatm0:`. This node in turn hands the traffic to hatm0 to send it onto the wire. Downlink traffic takes the opposite way.

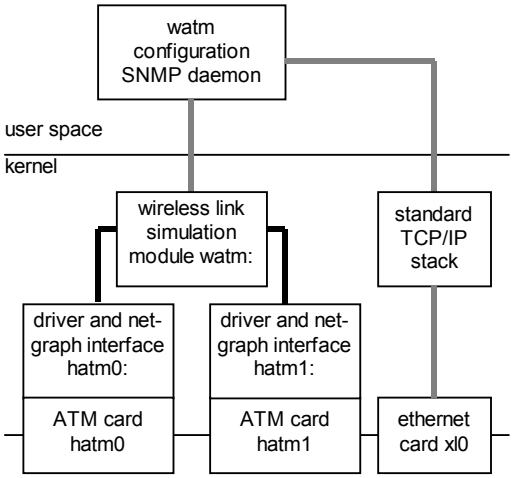

Figure 5: Wireless link simulator

ATM cards and drivers. The link simulation PC is equipped with two Marconi (formerly Fore) HE155 ATM cards[5]. These cards can be configured to run either SDH or SONET on their optical link. They are also available with an UTP physical layer. The drivers for these cards connect to the netgraph subsystem through two nodes call `hatm0:` and `hatm1:`. The driver parameters are tunable before loading the drivers through system tunables. It turns out that for the given application it is necessary to enlarge the receive and the transmit rings for these cards. This is because the cards are used in raw AAL mode—each ATM cell is received by the operating system as a single packet. Normally ATM cards handle AAL5 segmentation and reassembly in hardware so that the burden on the operating system is much lesser. For the target bit rate of 50Mbit/s this means, that the driver must be able to send or receive approximately 120,000 packets (cells) per second. This requires to make the transmit and the receive rings as large as possible.

Another necessary optimisation is interrupt coalescing. With a PC architecture it is in principle possible to have

more than 100,000 interrupts per second. This, however, appears to have serious influence on timekeeping and other functions of the kernel. Therefore, the ATM cards implement a method where an interrupt is generated only after either a given number of packets have been received or the input queue has not been empty for a given amount of time. There is of course a trade-off: the larger one makes these parameters, the more inaccurate will be the delay simulation, because the time a cell is waiting in the input ring for an interrupt cannot be accounted for. In our application we have set this time to approximately 100µs. This generates an interrupt rate of around 10,000 interrupts per second.

Wireless link simulation module. The wireless link simulation is implemented as a netgraph module[4]. Netgraph is a subsystem somewhat akin to System V STREAMS. Noteable differences are that netgraph is much more lightweight, yet employs the multithreading architecture of the kernel, and that netgraph can be used to build arbitrary graphs, not only stacks.

Using netgraph as the implementation base allows interfacing the link simulation module not only to ATM, but to any kind of network equipment, given that the driver for the network card interfaces with the netgraph sub-system. So it is possible to use ethernet cards and ethernet drivers to simulate packet based communication instead of ATM communication without changes to the link simulation module itself.

The link simulator implements two kinds of influences on the communication:

- errors at the cell level
- time variable or constant delays

For all three kinds of effects tables are computed by the configuration tool based on physical parameters (weather conditions, orbits, link budget, coding schemes and so on) and downloaded into the simulation module. Each of the effects may be switched on or off individually or even stopped or started at arbitrary points in the simulation.

The structure of the watm node is shown in Figure 6. The node has two ports: the upper port that connects to the ISS side ATM port and the lower port that connects to the ground station ATM port. Each of these ports consists of a configurable number of hooks (one hook pair is needed for each virtual channel connection (VCC)). Each port has an output queue which holds the cells while they are delayed. The current delay and error probability values are fetched from the corresponding tables.

The node has a number of global configuration parameters that must be set before the node is started:

- `factor`. This is the 'speedup' factor. Simulation time is speed up by this factor plus 1. That means if the factor is zero, simulation happens in real-time, if the factor is 9, simulation runs ten times faster.
- `tticks`. This is the number of ticks that one table entry is to be used. This allows to trade between memory requirements for tables and accuracy of the simulation tracking changes in delay and error rate.
- `length`. This is the number of entries in each of the tables.

So the overall simulation time (in ticks) is

```
T = (factor + 1) * tticks * length
```

When the tables are exhausted (after T ticks), the simulation automatically starts at the begin.

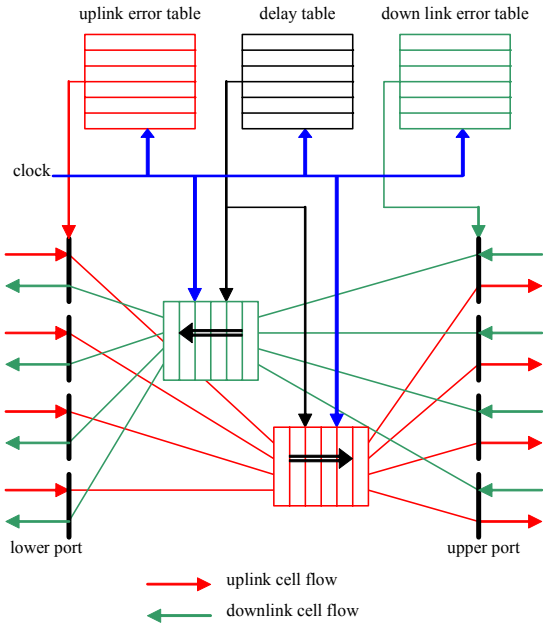

Figure 6: Watm simulation module

Each of the three configurable tables consists of `length` 32bit integers. For the delay table this is the delay in microseconds, allowing for a maximum delay of 2^{32}-1 microseconds or more than one hour. For the error tables this is the cell error probability multiplied with 2^{32}. The tables must be loaded before the simulation can be started.

At any time during the simulation the global status and some statistics can be retrieved from the node:

- `running`. This is a flag that is 1 when the simulation is running.

- `index`. The current index into the tables.
- `ttick`. The current tick count for the current table entry.
- `wraps`. The number of table wraps or complete simulation runs since begin of the simulation.

Per port statistics are:

- `in`. Number of cells received on this port.
- `out`. Number of cells sent on this port.
- `err`. Number of errored cells on this port.
- `no_conn`. Number of cells that where dropped because no node is connected to the hook (this is a configuration error).

The simulation accuracy is heavily influenced by the rate of the operating system kernel clock. Historically FreeBSD is built with the kernel clock rate set to 100/sec or 10ms. For the simulation machine, however, the clock rate is usually set to 10,000 (100µs). The minimum that can be set is currently around 40µs because of some misfeatures in the TCP stack. The trade-off for setting the clock rate is between simulation accuracy and clock interrupt processing overhead.

Wireless link configuration SNMP daemon module. To facilitate remote usability of the wireless link simulation system the entire configuration and status monitoring is done via SNMP. We use a very small SNMP daemon implementation. This daemon implements just the raw SNMP infrastructure plus loadable module support. The watm configuration is just a loadable module for this daemon. The module implements the following MIB variables and tables:

- `begemotWatmNode` and `begemotWatmNodeId`. These are the names and the netgraph node Id of the `watm:` node.
- `begemotWatmFactor`, `begemotWatmTableTicks` and `begemotWatmTableLength`. These correspond to the global configuration parameters (see above).
- `begemotWatmStart`. This is a Boolean flag that starts and stops the simulation.
- `begemotWatmPortConfigTable`. This table has two entries: one for the upper and one for the lower port. Each entry controls which nodes should be connected to these ports and how many hooks to connect as well as the ATM VPI and VCI values for each ATM connection. This is used by the daemon to build the actual netgraph configuration in the kernel.
- `begemotWatmPortStatisticsTable`. This table contains one entry for each port with the statistics that where described earlier.
- `begemotWatmStatus`. Contains the node's status variables.
- `begemotWatmDelayTable`, `begemotWatmUpErrorTable` and `begemotWatmDownErrorTable`. The three tables as octet strings.

The parameters in the configuration table are set via the daemon configuration file. When the daemon starts, it reads this file and builds the netgraph in the kernel (node and hook creation). When this is done a remote application can set the simulation parameters and tables, start the simulation and monitor it via the status variables and statistics table.

Control and monitoring station.

The control station has been described elsewhere[2]. In this paper we describe only additional modules.

Wireless link configuration tool. This command line tool is used to access the wireless link simulation system remotely per SNMP. It can configure the simulation, start and stop it and retrieve the current state and statistics. The tool has numerous options to access this functionality. We provide just three examples of use:

Assume that the following table is contained in the text file `delay.tab`:

```
0     273.225
60    271.865
120   270.487
…
1500  247.663
1560  247.563
1620  247.597
…
3180  272.903
3240  274.136
```

This table assumes communication between a ground station and the ISS via the Artemis satellite. The first column is the current time in seconds since start of visibility and the second is the current one-way delay in milliseconds. The command

```
watm –udelay 3240 \* `cat delay.tab`
```

loads this table into the wireless link simulation.

While the simulation is running the current state can be retrieved with:

```
watm -s
```

an example output would be:

```
factor = 1
table_ticks = 31641
table_length = 1024
index = 292
tick = 21118
wraps = 0
status = running

one sweep is 3240.038 seconds

currently 29% (926.029000 secs)
```

Current statistics can be retrieved with

```
Watm -s
```

the output of which could look like:

```
up:   77505175/0/3281682/0
        (in/err/out/noconn)
down: 3282984/0/77474370/0
        (in/err/out/noconn)
inflight: 1302/30805 (up/down)
```

Scripts. A number of scripts all based on the watm tool have been written to do measurements and experiments. Some of them just configure and start the simulation and then periodically retrieve status and statistics for later analysis. There is also a script '`run`' that additionally starts a number of gnuplot[6] processes that display real-time delay and error graphs.

RESULTS

A number of measurements have been done to show, that the link emulation works as expected. Figure 7 shows the configuration used for these measurements.

In addition to the simulator there are two PCs, brittain and honshu, that are equipped with ATM cards and connected to the simulator. They are also connected to the control station via the intranet for time synchronization and controlling purposes. The time on all four machines is synchronized to a stratum-1 clock with ntp. In a constant temperature laboratory time synchronization is better than 50µs between all machines after several hours of uptime.

ATM level measurements

A series of measurements have been done to verify the delay and error behavior of the simulated link. These have been done with two tools: atmsnd and atmrcv. Atmsnd generates AAL5 traffic and inserts timestamps and sequence numbers into the AAL5 frames. Atmrcv receives these frames.

The first measurement has been done to verify the delay behavior of the simulation. Honshu has been used as the ISS side of the link and brittain as the ground station (see the delay table above).

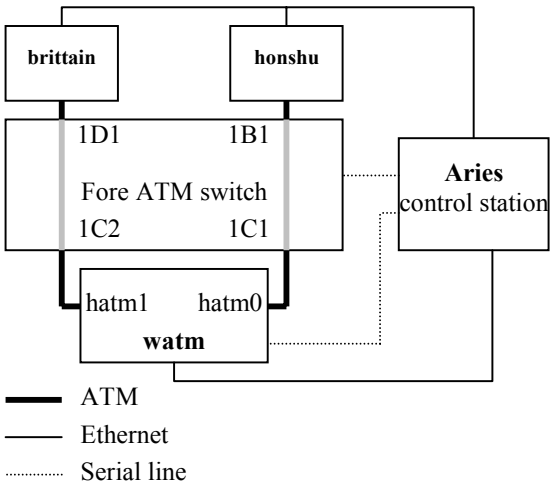

Figure 7: Measurement scenario

Honshu has been set up to generate AAL5 frames of 17000 bytes every 3 milliseconds and send them at a peak cell rate of 120000 cells per second CBR on VCI 34. These parameters give an actual mean cell rate of 118347 cells per second. This is a net downlink bandwidth of 45.44Mps and a gross bandwidth of 50.18Mps. The atmrecv command on brittain reports the mean delay and standard derivation as well as losses every second.

Brittain has been set up to generate AAL5 frames of 700 byte every 3 milliseconds and send them at a peak cell rate of 5500 cells per second CBR on VCI 35. This results in an actual uplink mean cell rate of 5001 cells per second. This corresponds to a net bandwidth of 1.92Mps and a gross bandwidth of 2.12Mps.

The resulting delays are shown in Figure 8. The lower line is the theoretical delay.

The measured delay is about 4.3ms larger than the theoretical delay. The main reason for this is the segmentation and re-assembly delay. In the sender the AAL5 frame is timestamped just before the **first** cell of the frame is transmitted. In the receiver the receiving timestamp is obtained just after the **last** cell was received. Thus there is an inherent additional delay in these type of measurements that is just the time between the first and the last cell. For the 17000 byte downlink frames (355 cells) this is 2.96ms and for the

uplink frame (15 cells) the delay is 2.72ms. Other components of the additional delay are the application delay (the time it takes to copy the data between user space and kernel), the transmission delay on the links and the switch delay.

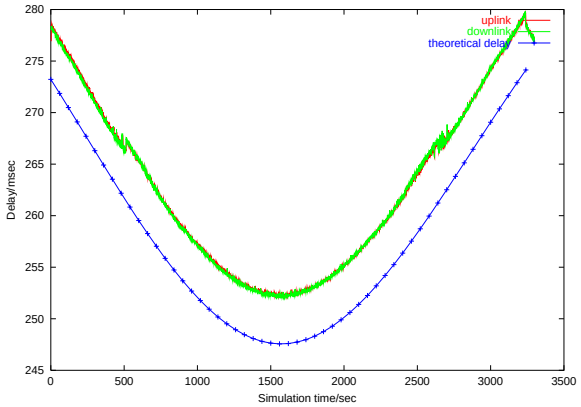

Figure 8: Up and downlink delays

The fuzzy zones at a delay of 267 milliseconds occur because of interferences between the delay and the packet segmentation time. For different AAL frame sizes these fuzzy zones are at different points. This is actually an artifact of the measurement method not of the link simulation itself.

To get an estimate of the delays which are not simulated by the link simulator the same measurement as above has been done, but with a through path between switch ports 1D1 and 1B1. This eliminates the transmission delay of the links to the link simulator itself and the delays in the link simulator. The resulting delay is approx. 4ms in the downlink and 3ms in the uplink. These measurements leave an additional delay of approximately 0.5-1.0msec that is obviously caused by the link simulator itself (device drivers, bus speed and others). That means an error in the delay simulation of below 0.5%.

The error insertion function has been tested by using a linear error function on both links (that is an error rate that linearly increases with time). The AAL5 frame size has been made much smaller, because a lost cell means, that the entire frame is lost. The measurements have been done with a speed factor of 4. As an example Figure 9 shows the theoretical (straight line), inserted (lower fuzzy line) cell and measured packet error rates.

The generated-loss line shows the actual error rate inserted by the link simulator measured every 5 seconds. This is obviously quite near to the theoretical error rate. The upper line is the application-observed packet error rate. This error rate is 3 times the cell error rate. This is because the AAL5 frames are 3 cells long and a lost cell means one lost frame if it is the first or second cell of a frame or two lost frames if it is the last cell of the frame. The last effect is compensated by the fact that there is no difference whether the the first or the second cell of a frame or both are lost.

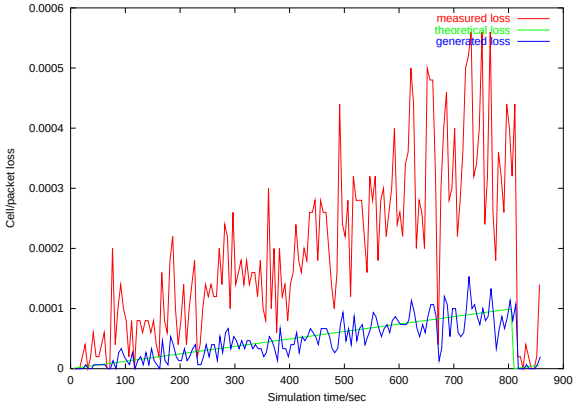

Figure 9: Downlink errors

TCP measurements

A number of TCP performance measurements have been done with different TCP flavors and parameters. The measurements have been done with a re-written version of the distributed benchmark DBS[7].

Full rate downlink data transfer. This test is configured to transfer data per TCP at the largest possible data rate. Data is red and written in 64k packets as fast as possible. The receive and send socket buffers have been configured to be 4.8MByte. The test is run with no errors on the link and speed-up factor 4 over 1200 seconds.

The resulting throughput of 46MBit/sec is quit near to the theoretical throughput. For 120000 cells/sec the net bit rate is 46.1MBit/sec (not counting approx. 40kbit/sec AAL5 overhead). The application observed delay is somewhat larger than the link delay (402msec versus 247-274 msec) because of queuing delays and the large send buffer. The measured RTT is also larger than the two way delay for the same reasons.

Figure 10 shows the current send window (straight line), the throughput (left line) and the congestion window at start up. Note, that such a fast startup is possible only with the large send and receive buffers for this long delay. A normal, untuned TCP will get a throughput of a couple of 100kbits/sec only after a much longer startup time.

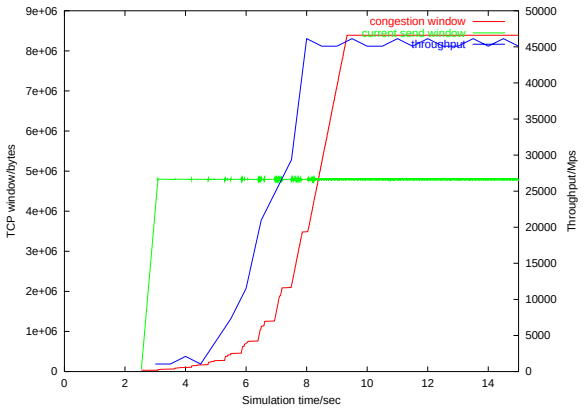

Figure 10: Downlink TCP measurements

Another interesting result is shown in Figure 11. The upper line is the delay measured at the application level and the lower line is the link delay.

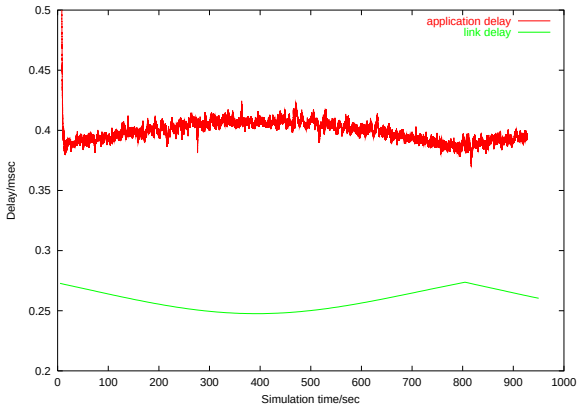

Figure 11: Downlink link and application delay

USAGE CASE: CONNECTING THE ISS TO EARTH VIA A GEO SATELLITE

The simulator has been used to explore different options of communication between the ISS and the earth. One of those scenarios is the communication between the ISS via a Ka-Band link to a GEO satellite and from the GEO satellite via Ka-Band to a fixed earth station (Figure 12).

Several parameters are used as input to the simulation:

- the orbit of the ISS and the positions of the GEO satellite and the earth station
- the link budgets of both links in both directions
- the weather conditions in the geographical region of the earth station
- the coding parameters on the link

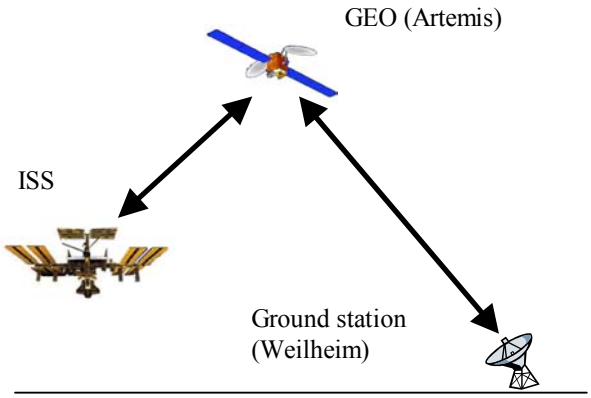

Figure 12: ISS example

The simulation setup is shown in Figure 13. Both ATM links of the simulator is connected to equipment that multiplexes an artifical HSM (high speed multiplexer) cell stream with a bidirectional AAL5 stream that comes from an Ethernet interworking function. On one side the Ethernet is connected to a laptop (which simulates an on-board laptop) and the HSM part is configured to generate HSM cells. On the other side a small local network is connected to the Ethernet port and the HSM part is configured to receive the HSM cells.

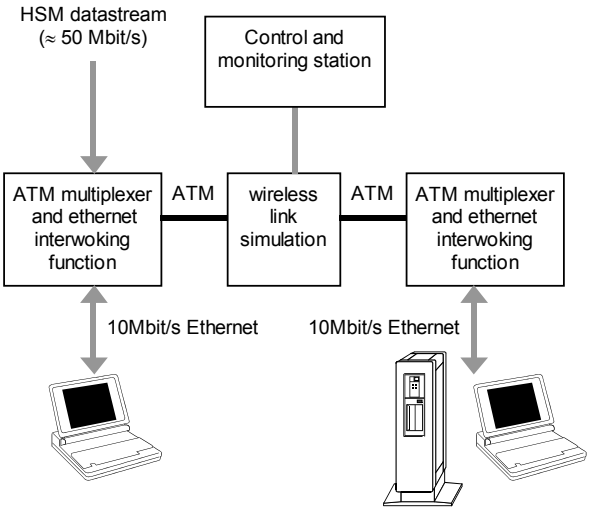

Figure 13: Simulation setup

Using this setup several applications have been demonstrated. It could be shown, that it is possible to use IP based applications between ISS and ground while also sending the HSM data stream to ground.

During these experiments it has also been shown, that it is absolutely necessary to tune TCP based applications with regard to the delay and possible errors. Out-of-the-box TCP is usually tuned to typical small terrestrial bandwidth-delay products and to the assumption that network loss is the result of congestion. In the given case, however, the bandwidth delay product is very large and, because of the dedicated link, loss is a result of link errors.

FUTURE WORK

There are several directions for future work with regard to the simulation.

First of all the simulator should be better integrated with other tools. Two steps when setting up a simulation involve computing the delays from the orbit and ground station parameters and computing cell loss rates from the link budget. Both of these steps use special tools and it should be possible to directly get the output from these tools into the simulator.

The ATMSat demonstrator web interface should be adapted to support also the watm simulation.

Some minor changes are necessary to support packet based networks instead of ATM.

The characteristics of the FEC scheme should be taken into account when simulating errors. When an error happens with a turbo code, for example, several cells will actually be affected.

ACKNOWLEDGEMENTS

The work on ATMSat was supported by the german Federal Ministry of Education and Research and the German Aerospace Center (DLR). The work on the GEO simulator was supported by Astrium and the German Aerospace Center (DLR). Thanks also to Dr. Hermann Bischl from DLR for the delay and error tables.

REFERENCES

[1] "*Target Systems Report*", ATMSat: ATM-Based Multimedia Communication via LEO-Satellites. BMBF Project, Oct. 2001

[2] Brandt H., Krepel F., Tittel C. *A Multiple Access Layer and Signalling Simulator for a Ka-Band LEO Satellite System*. 20st International Communications Satellite Systems Conference and Exhibit. Montreal, Canada, May 2002

[3] http://www.freebsd.org

[4] Archie Cobbs, "*All About Netgraph*", Daemon News, March 2000 [journal online]; available from http://www.daemonnews.org/200003/netgraph.html; Internet; accessed 09 September 2002.

[5] http://www.marconi/com

[6] http://www.gnuplot.info.

[7] Murayama Y., Yamaguchi S. DBS: A powerful tool for TCP performance evaluation. Performance and Control of Network Systems, Proceedings of SPIE, Volume 3231, November 1997