A networking perspective on satellite constellations

Ecole d'Hiver des Télécommunications de Sophia Antipolis
17h30-18h00, jeudi 10 décembre 1998

Lloyd Wood (L.Wood@surrey.ac.uk)
http://www.ee.surrey.ac.uk/Personal/L.Wood/

Centre for Communication Systems Research,
University of Surrey
http://www.ee.surrey.ac.uk/CCSR/

We aim to build up a picture of satellite constellations and how we can simulate them as networks. To do that, we'll cover:

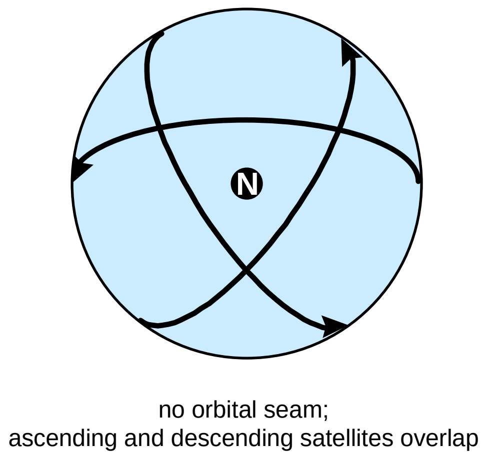
- orbital geometry of satellite constellations
- other physical concepts that are useful to know
- satellite networking before inter-satellite links...
- ...and after; the pros and cons of inter-satellite links
- Manhattan mesh topologies
- multicast
- protocols (IP vs ATM) and issues
- simulation tools

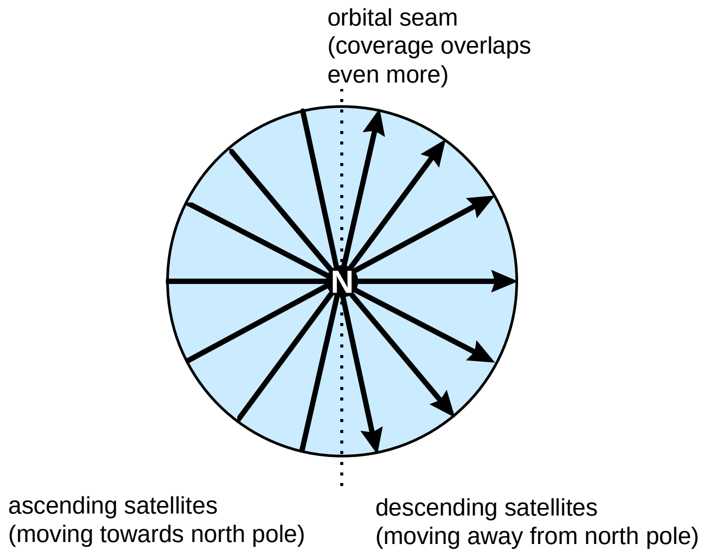
What makes a satellite constellation?

multiple satellites with *co-ordinated* coverage of the earth.

First constellation proposed by Arthur C. Clarke in 1945: three satellites spaced 120 degrees apart over landmasses in the geostationary orbit... these days geostationary orbit is somewhat fuller!

Geometry of low-earth-orbiting (LEO) constellations studied since the 1960s.





Walker Delta or Ballard rosette

Walker seamed polar star

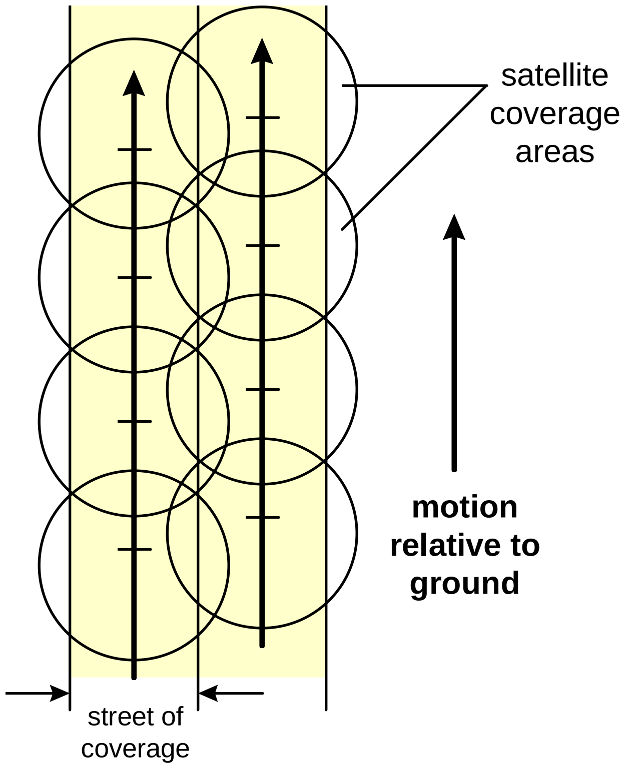
topologically, circular constellations lie on the surface of a torus (doughnut/tyre).

Quick overview of constellation geometry

Walker developed ‘star’ (polar seamed) and ‘delta’ (rosette) constellations, using streets of coverage.

Ballard explored the rosette coverage in depth.

Adams and Rider used ‘streets of coverage’ approach to optimise constellations with polar orbits.



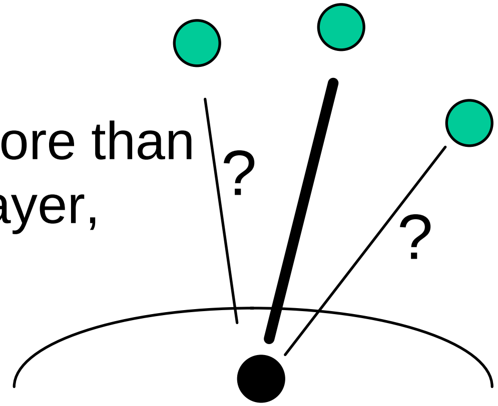
Draim examined optimum *elliptical* orbits; came up with optimum elliptical coverage, since adopted by MCHI’s Ellipso. (But hanging at apogee isn’t that interesting from an intersatellite-link viewpoint.)

This physical design affects magnitude and rate of change of distances and visibility between satellites and between satellites and users. It determines the topology of your network.

Other useful concepts

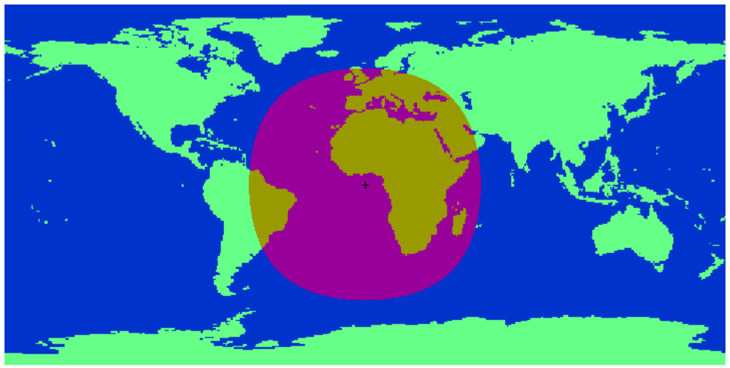
Diversity

is the ability of the ground user to communicate with more than one satellite at once. Usually handled at the physical layer, it can be a major factor in constellation design.

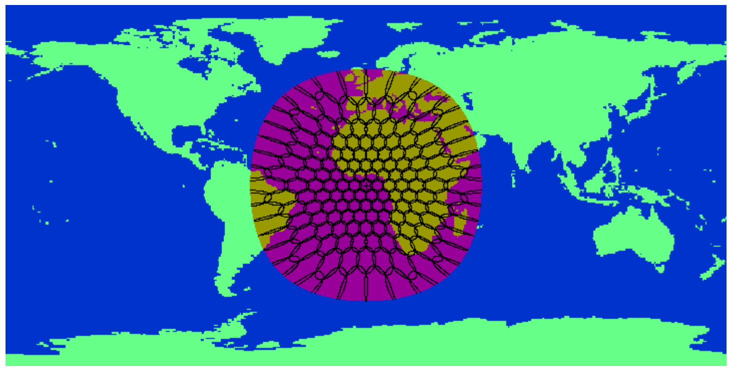


footprints and spotbeams

Satellites of old had a single large coverage area or footprint, but for frequency reuse and increased capacity, spotbeams are now the norm. Overlap of spotbeams and coverage as latitude varies affects constellation traffic.



without spotbeams



with spotbeams

Satellite networking until now

Bent-pipe satellites

These are 'dumb' (but robust!) satellites that simply amplify and frequency-shift whatever is aimed at them, as opposed to more complex satellites with on-board processing, or even more complex ones with onboard switching or routing.

In the bent-pipe approach, all networking equipment stays on the ground (where it's easier to fix). That's well-proven for GEO, and has been extended to VSAT networks, where various slotted-aloha allocation techniques tailored for the application utilise a transponder.

Moving to spotbeams

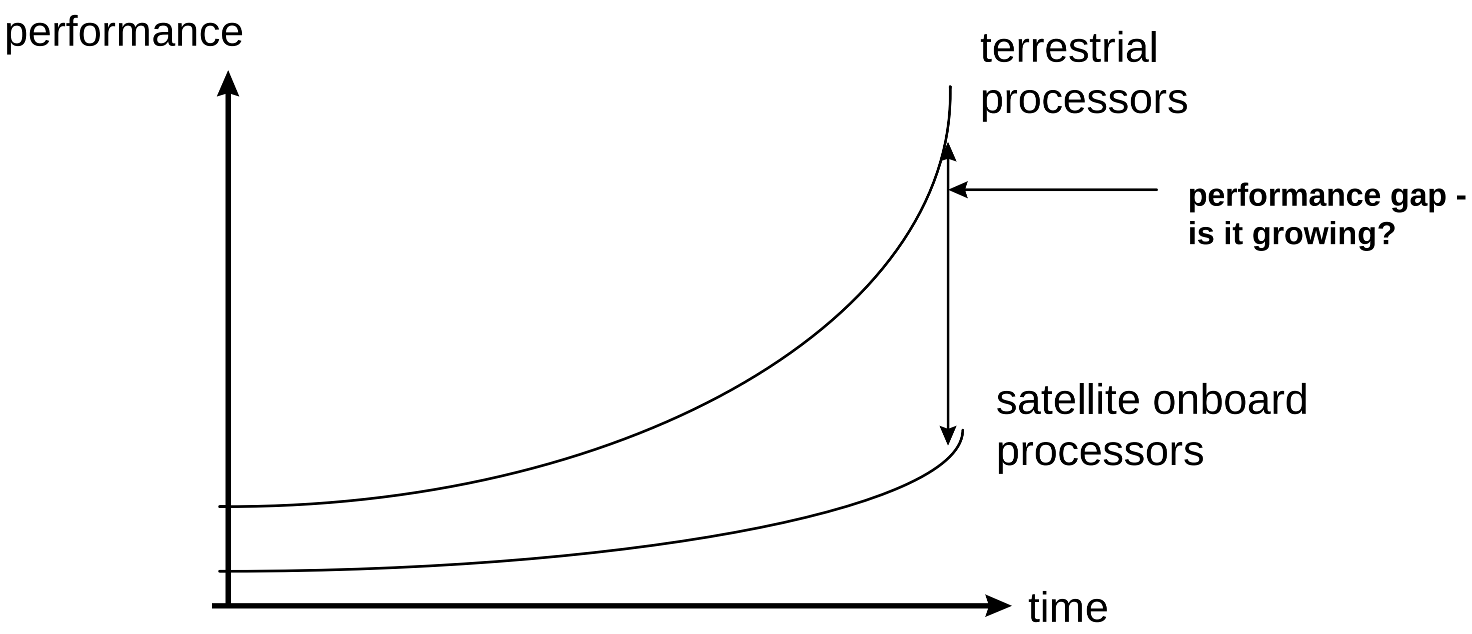
Spotbeams for frequency reuse to increase capacity are fairly new; e.g. Inmarsat-3 satellites, with spot beams for mini-M, launched in the last year or so.

And beyond

Intersatellite links have really only been proven on a large scale by *Iridium*. Some on-board processing. But most work on on-board switching and routing is still not deployed - *Iridium* being the notable exception here.

An argument against intersatellite links consider Moore's Law

Processor performance doubles roughly every eighteen months - but that's performance of *terrestrial* processors.



Satellite processing in a harsher radiation environment has always lagged behind; conservative deployment of stuff that's tested and known to work. Does Moore's Law work the same or differently in space?

Why have intersatellite links between satellites?

Advantages of ISLs

- satellites don't need ground stations in sight to function; permits true global coverage, including oceans, with a small number of ground stations. (*Iridium* has twelve ground stations; *Globalstar* builds a ground station in Iceland for the North Atlantic market...)

- provides redundancy of routing if a ground station fails or is politically out of your control.

- increased independence from the terrestrial networks and control of performance.

- having your users talk to other users entirely within the constellation is locally efficient, possibly important to revenue in the long run.

...these make a lot of sense from a *logical networking* viewpoint

Why *not* have intersatellite links between satellites?

Disadvantages of ISLs

- ISLs need both on-board processing (OBP) and on-board switching or routing to support them.

- increases system and management complexity - payload power requirements and weight, software bugs (*Iridium* testing...) - cost!

- technical issues - slewing of Tx/Rx to track satellites, Doppler shift with latitude. Laser ISLs are researched, but not yet deployed.

- increased reliance on the satellite segment; lots of complex but identical satellites can all fail the same way - one big satellite?

- you must define your system in a lot more detail; it may be obsolete before you've finished designing it, whereas you simply change the ground equipment with simpler bent-pipe systems.

...these make sense from a *physical satellite engineering* viewpoint

The real throughput constraint

The earth-space air interface is the real constraint on throughput here for our hypothetical networks; there's far more available capacity in a single fibre, and you can reuse the frequencies easily by bundling fibres.

How often can you reuse available frequencies to increase satellite capacity? (lower orbits, larger dishes for narrower spotbeams...)

Like the telephone modem, the wireless connection constrains overall throughput to the user.

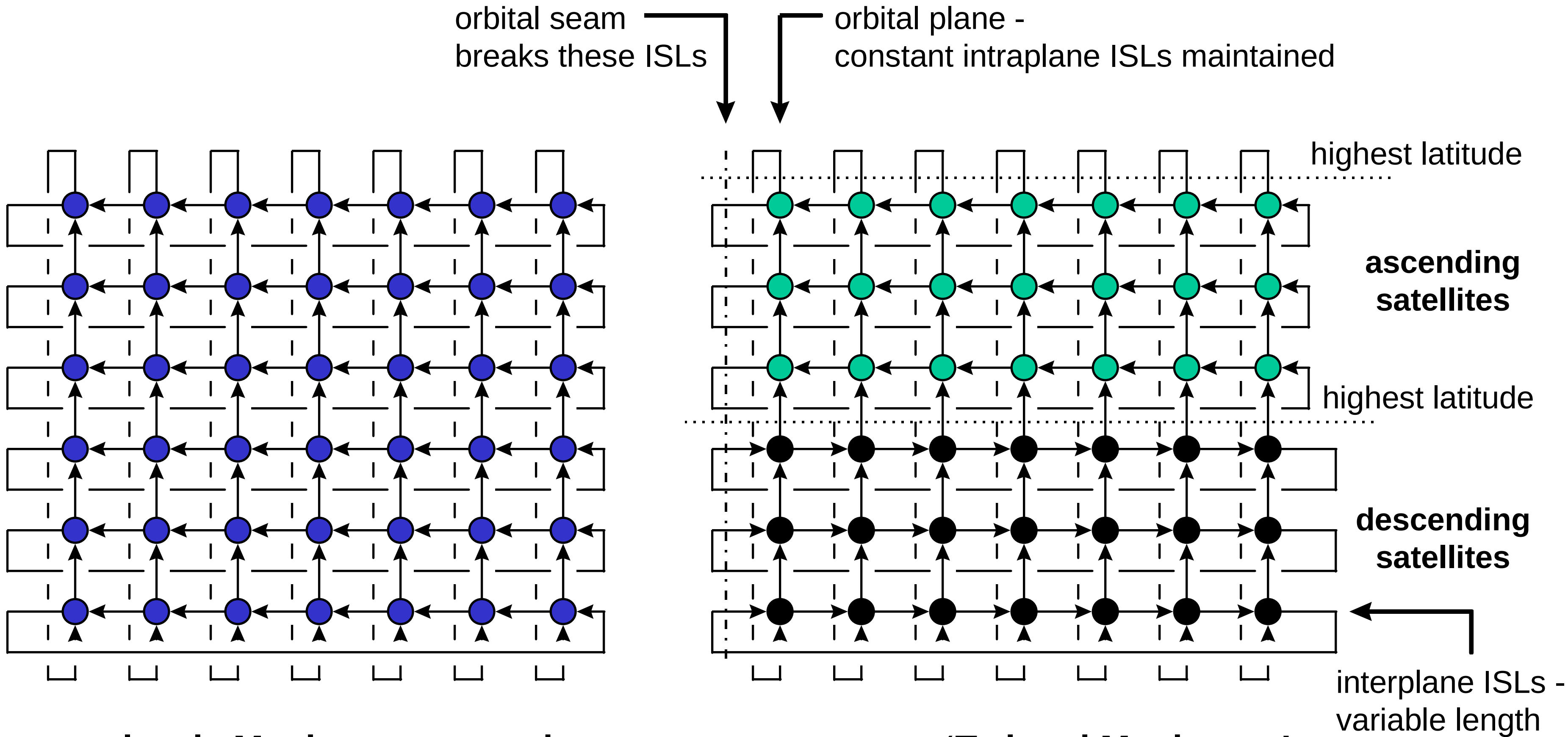
Keeping pace with terrestrial fibre and beyond-gigabit routers will be difficult; satellites will always be an expensive option for those that need it, but they won't be "fibre in the sky".

This is another argument in favour of bent-pipe processing, so that terrestrial upgrades can utilise available frequencies as quickly as possible.

but staying with ISLs and networking...

What network shape do ISLs and geometry give us?

Toroidal; links fail at highest latitudes due to Doppler/slewing/ swapping sides. The basic geometry repeats with time; it's *quasistationary*. We'll pretend that all links are uni-directional to clearly show the effects of crossing orbits on the topology.



classic Manhattan network

'Twisted Manhattan' satellite network variant

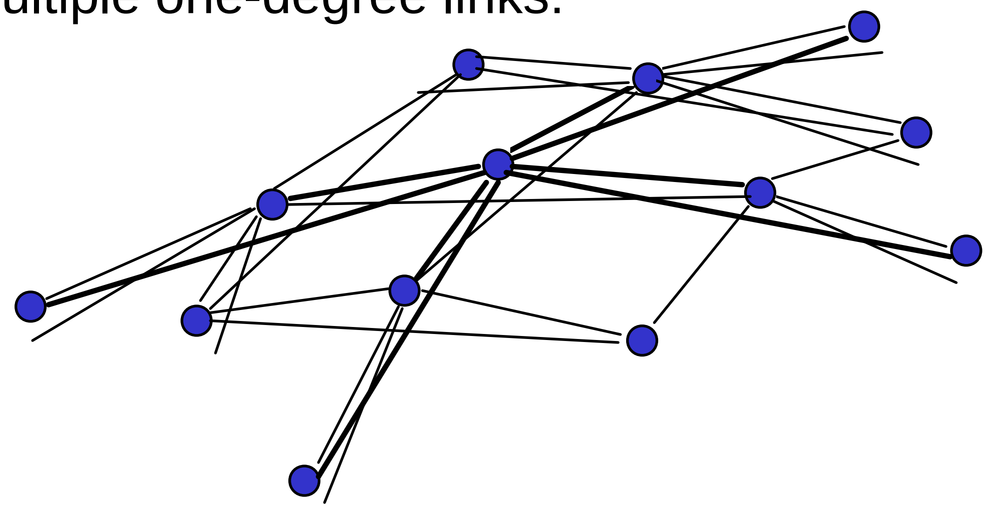
Variations

hexagonal, triangular and other meshes of connectivity are also possible.

Diversity *within* the mesh - the geodesic factor

It’s possible for a satellite to communicate with anything in line of sight (i.e. not over the atmospheric horizon).

Two-degree ISLs were included by Teledesic, where satellites talk to their neighbours’ neighbours, they have a propagation delay and processing advantage over multiple one-degree links:



This method has also been proposed for several geostationary ring constellations, where the delay advantage becomes considerable (bigger delays mean bigger savings).

So what makes this topology interesting?

Manhattan networks are also used in distributed-memory parallel computer architectures, although different terminology is used.

Parallel computers	**Satellite constellations**
flits	packets
wormhole routing	explicit virtual circuits
processing overhead important	routing/switching overhead may be important
but	
processors assumed reliable	satellites may fail
propagation delay insignificant	propagation delay *very* significant!

Results from one domain can be applicable to the other - with some caveats. There’s common ground for reuse of knowledge and results.

Now that end-to-end deadlock-free routing in the Manhattan network is pretty much understood, research here has moved on to looking at multicast - while I’m looking at multicast in Manhattan-like satellite constellations...

Why am I looking at *multicast*?

It's the 'general case' - broadcast (one to all) and unicast (one to one) can be thought of as special instances of multicast.

End-to-end communication, tunnelling through whatever proprietary networks are between the end-points, can always be done. Multicast is *harder*.

Multicast aims to make *efficient* use of network capacity.
Satellite capacity is expensive and always charged for; there's a visible effect on revenue by using capacity efficiently. Profit motive!

There are many types of multicast; IP implementation is *way* ahead of ATM (although ATM is improving, e.g UNI 4.0...)

Dense/sparse protocols - make assumptions about the size of the multicast group, and how its members interact.

Reliable/unreliable protocols - acknowledgement implosion problem

Shared trees (a core-based protocol?) make sense from a satellite viewpoint.

We have a network constellation - so what does it use?

Internet Protocol vs Asynchronous Transfer Mode (IP vs ATM)

ATM
ATM over satellite has received a lot of attention (RACE CATALYST, etc.)
But ATM was intended for telco-owned trees; it makes less sense in a mesh or mobile network, given the overhead of managing VPs and VCs.

Given that a lot of end hosts run IP, it's reasonable to expect a lot of ATM networks to be carrying tunnelled IP traffic,with resulting overheads and difficulties with anything that doesn't fit into an end-to-end tunnel.

IP
IP is packet-based (sensible for meshes), and has interesting multicast support. Since IP is being sent out by the end hosts, why not run IP over satellite as directly as possible, and utilise IP routing for the constellation? Keep things simple...

Neither ATM nor IP provides a Media Access Control (MAC) layer for handling communication via spotbeams etc, so we'll need to design and add one.

IP routing has problems... Problem #1

Global visibility

Satellite networks are mobile global backbones; they interconnect the world.

If every satellite had to store information on how to forward to every network in the world, you'd have routing table overload onboard your satellite, and you'd be overwhelmed by routing updates.

This is often said to be why IP routing onboard satellite "just can't be done".

Solution: Restrict visibility

The constellation can be thought of as a single private network; as a backbone it's a *known cloud*. So, we can do things in private within our network...

IP-in-IP tunnelling is inelegant, but network address translation (NAT), where only the gateways hold the complex tables and the satellites merely know enough to forward to the gateways, reduces on-satellite overhead.

If we translate everything to an abstract multicast address with a loop-free protocol, explicit multicast routing can handle end-to-end communications, too.

Problem #2

Variable packet size

IP packets can be any length; this fits badly with any wireless interface where you have to divide up frequency/time efficiently, and is often cited as a good reason for using fixed-length ATM cells.

Solution: Restrict the packet size

We can fragment IP, but it gets messy; better is to utilise IPv6's implementation of Path MTU (Message Transfer Unit) Discovery.

This dictates the packet size and allows us to map it cleanly to our MAC layer. Constellation users' equipment can default to this size; as a satellite network we need a setup procedure in any case, if only for billing purposes, so we can bring it in here.

This results in wasted space in e.g. acknowledgements, but the resulting inefficiencies can't be worse than those encountered with IP over ATM... the comparison will be interesting.

Summary - Simplicity is its own reward

If we're going to be carrying IP traffic anyway, we may as well *utilise* its attributes and advantages. We want to:

• do routing in only one place - one layer 3 (IP).
Unnecessary duplication of routing and resulting overheads are avoided.

• map IP to MAC as cleanly as possible via Path MTU discovery.
Avoid fragmentation and complex 'we've lost a fraction of of this packet' problems and losses encountered with IP over ATM.

• consider everything a multicast within our backbone.
With a known topology, multicast becomes a more tractable problem; easier than the open inter-domain problem.

Simulation tools I'm using

Since I can't *build* a constellation, I simulate it. I'm using two free open-source unix/X packages:

SaVi
Physical modelling of the constellation only, showing geometry and coverage. Useful for checking that a chosen constellation network makes sense from a physical viewpoint, with a 3D visualisation.

Constellation geometry is easy to script up using embedded Tcl interpreter.

network simulator (*ns*)
The package for simulating IP networks; many papers discussing tweaks to TCP test their ideas in *ns* first. And it supports multicast.

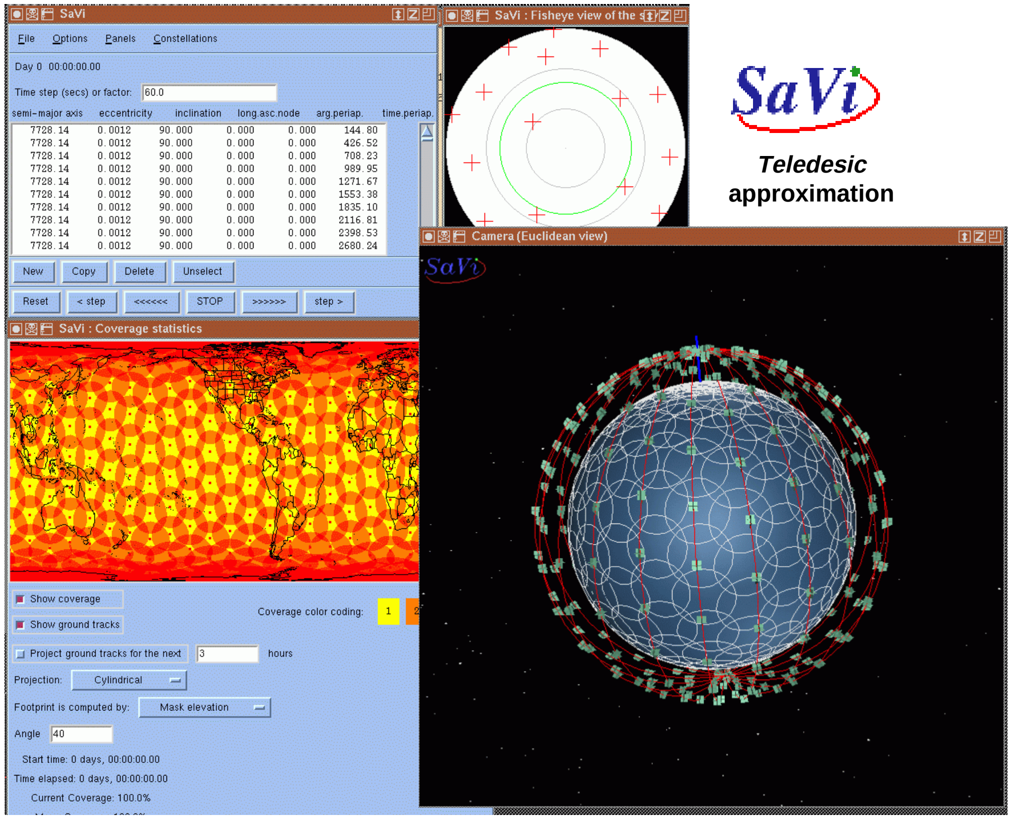
There's also a network animator (*nam*) which lets you view your network simulations from tracefiles generated by *ns*.

Simulation scripts use Object Tcl as an easier way to manipulate C++ objects.

Satellite Visualisation (*SaVi*)

developed at the Geometry Center, University of Minnesota

Relies on *Geomview* (also developed at the Geometry Center) for three-dimensional visualisation. That's quite processor-hungry; an SGI (with texture-mapping support) is best; a Sparc Ultra is adequate.



***Teledesic* approximation**

Not perfect; doesn't do three-line-element sets or spotbeams, ISLs can't be scripted yet, orbit specialists will (always!) want more realism.

But it's far better at showing you the constellation geometry than *Satellite Toolkit* or *SatLab*; scripting is easier than setting parameters, and gives you a good feel for constellations.

ns and *nam*
developed at the University of Berkeley, California

ns simulates IP networks - TCP is its strength, multicast and routing support is improving. Object Tcl interface to existing C++ network objects/agents; easily extensible, and the Object Tcl code is fairly clear and readable.

```
# we set up the topology, now we set up the multicast...
set mproto CtrMcast

set group [Node allocaddr]

# set all nodes to be multicast nodes
set mrthandle [$ns mrtproto $mproto {} ]

set m5 [new Agent/LossMonitor]
$ns attach-agent $g(5) $m5

set m12 [new Agent/LossMonitor]
$ns attach-agent $g(12) $m12

set udp0 [new Agent/UDP]
$ns attach-agent $g(1) $udp0
$udp0 set dst_ $group
 set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0

$ns at 0.1 "$cbr0 start"
$ns at 1.0 "$g(5) join-group $m5 $group"
$ns at 2.0 "$g(12) join-group $m12 $group"
```

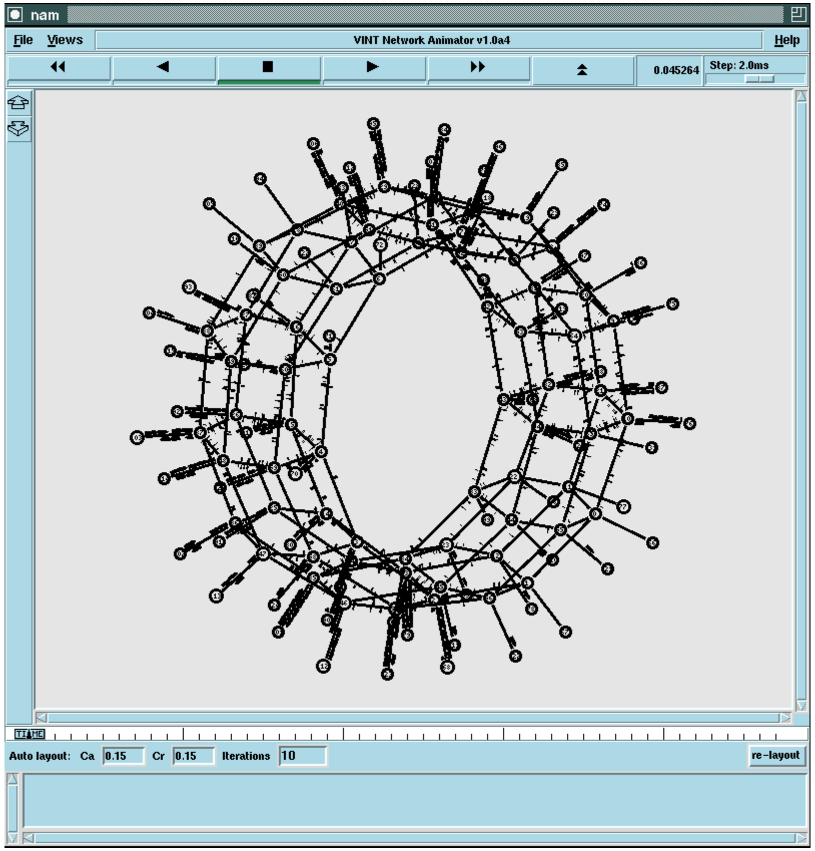
Tried and tested; building on man-years of proven work is better than starting from scratch.

But it's not perfect:
shifting feature set
documentation never keeps pace; you're always better off experimenting with code.
still a steep learning curve
perpetually in beta (daily snapshots are the best way to run it...)
some find the command-line interface discouraging.

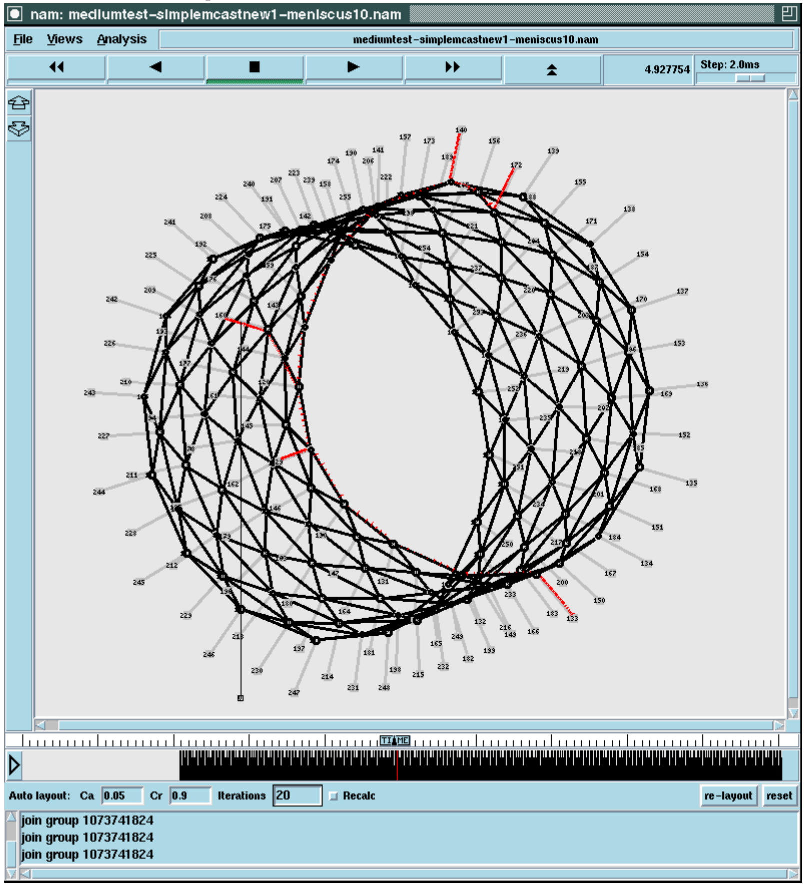
nam

network animator, views *ns* tracefiles

Lets you see what your network is doing; has come a *long* way in the past six months with the public release of the 1.0 alphas, which added a *lot* of visualisation functionality over the previous 0.8 public release.



fourway-connected full-torus rosette constellation exchanging routing information



sixway-connected seamed polar constellation showing a multicast communication in progress

Other simulation efforts around Europe

A number of people are building their own simulation tools to research various aspects of satellite constellations.

They come together under the COST framework to discuss and present their work. Currently running actions that are relevant include:

COST 252
Evolution of Satellite Personal Communication

COST 253
Service-Efficient Network Interconnection via Satellites

COST 256
Modelling and Simulation Environment for Satellite/Terrestrial Networks

Archived COST papers describing these efforts are worth reading

And a last research note...

Since this is simulation, it's only as good as the assumptions you make and the assumptions that you must inherit from your simulation tool.

Producing simulation results is one thing; producing *meaningful* and *useful* simulation results is another!

(Getting the network topology right is simply the first step!)