The DynaSat Information Architecture for Fractionated Satellite Systems

Joe Touch[1], Gregory G. Finn[2], Goran Scuric[3]
USC/ISI, Marina del Rey,CA,90292 USA

Michael Elkins[4]
SPARTA, Inc., El Segundo, CA 90245 USA

and

Richard Shiffman[5]
Digital Graphics Assoc, Los Angeles, CA., 90025 USA

DynaSat is a dynamic satellite information architecture that supports adaptive, fault-tolerant, and secure communication and resource sharing for distributed (fractionated) satellite systems. It supports safe, robust, and secure sharing and virtualization of both cluster and ground capabilities by combining a COTS secure hypervisor and trusted resource control and multiplexing software with secure overlays. DynaSat leverages USC/ISI's X-Bone overlay network and NetStation distributed PC architecture systems to deliver a space-based Internet for distributed satellite data collection applications. The result is a configuration-based approach that enables rapid implementation, simple operation, and provides programmers a familiar Internet-based environment. The architecture also supports service import/export to external ground users. This document provides an overview of the DynaSat architecture, discusses its design and resulting capabilities and advantages for fractionated satellite systems.

I. Introduction

DynaSat is an information architecture developed for the Future, Fast, Flexible, Fractionated, Free-Flying (F6) DARPA satellite research program system[1]. F6 is a cluster of satellites, each running a multiuser operating system, in which the cluster communicates as a LAN and a subset of satellites also have ground links. An information architecture describes how a subset of cluster resources can be interconnected and partitioned to provide a platform for a distributed service. Such services might include a single application that links an infrared camera, a ground imaging radar, a processor to integrate the two images, and a downlink to transfer the result to the ground. DynaSat leverages USC/ISI's previous DARPA projects to develop the X-Bone[2] virtual network system and the NetStation[3] distributed PC architecture system to deliver a space-based Internet for distributed satellite data collection applications.

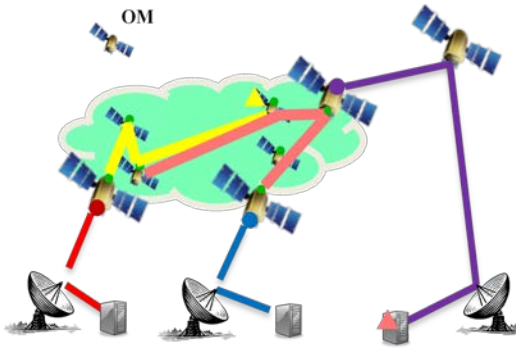

Figure 1 DynaSat system showing two overlays.

[1] Postel Center Director, Computer Networks Division, 4676 Admiralty Way, non-member.
[2] Senior Scientist, Computer Networks Division, 4676 Admiralty Way, non-member.
[3] Senior Programmer, Computer Networks Division, 4676 Admiralty Way, non-member.
[4] Senior Principal Scientist, Network Security Sector, 2450 E El Segundo Blvd STE 100, non-member.
[5] Design Engr., Programmer & Owner, Digital Graphics Assoc., 10318 Dunkirk Ave., non-member.

A. Motivation

F6 avoids the need for multiple large, integrated space platforms, custom to each deployed service. Instead it dynamically groups component capabilities, allowing each satellite to be simpler, additional satellites to be deployed incrementally, and new services to be created out of existing components in new ways as needed. It is a space-based variant of ISI's dynamic LAN-based workstation called NetStation. NetStation replaced a computer backplane with a 640-Mbps LAN, which enabled component devices such as processors, storage, cameras, and displays to be shared and dynamically grouped to support a variety of sometimes concurrent uses. F6 is the same concept applied to individual satellites and resources of those satellites as components.

Figure 2 The NetStation architecture.

DynaSat has a primary additional goal of leveraging modern, commodity space hardware and COTS operating systems to provide an information architecture for F6. This approach reduces both development time and risk. It provides a management system that coordinates resource sharing, ensures security and provides isolation between deployed groupings, and virtualizes resources and their interconnection to support a robust solution.

B. Overview

DynaSat supports existing Internet protocols and services using components that are assembled, configured, and extended as needed to operate in the F6 environment to support a variety of expected uses. The resulting system deploys an end-to-end secure network overlay for each mission, such that network and distributed systems services are localized within that overlay. To accomplish this goal, DynaSat assembles the following components:

- The X-Bone overlay configuration and management system…
  - with extensions for network QoS, device virtualization, scheduling, and hypervisor management
- A distributed set of operating systems, configured with login access and a distributed file system, activated. by the X-Bone's application deployment system…
  - using MooseFS with scripting extensible to other file systems and services.
- Controlled external access using configured NAT and DNS services…
  - providing a gateway to publicly available services through a guard gateway process.

The X-Bone system is publicly-available software that has supported network virtualization experiments in the US, Asia, and Europe since March 2000. It deploys and manages end-to-end virtual networks, and provides separate discovery and configuration operations. It utilizes a robust configuration protocol of two-phase commits nested idempotent transactions with rollback recovery and a heartbeat mechanism for fault recovery. Its virtual network architecture includes recursion and revisitation, which natively supports the necessary F6 networking.

The X-Bone system application mechanism supports installation and management of both node service installation and user applications[4]. This scripting system supports common Unix shell commands in an environment that provides the context of the overlay, enabling full user control within the confines of the deployed overlay using its assigned services. It previously supported web cache deployment[5], Active Network testbed configuration[6], and peer-to-peer experiments[7], and here it is used to configure distributed services such as the MooseFS file system and to initiate user programs.

File fault tolerance is supported by MooseFS using the X-Bone's application deployment script, which can be altered to support other file systems instead or additionally. Fault tolerance for management system processes is already supported within the X-Bone system by heartbeat monitors with cold-spare backups and distributed configuration files. Fault tolerance for user processes is implemented internal to the user applications using this same approach, with tools developed as part of DynaSat.

Components of a mission communicate securely with each other using conventional Internet protocols and interfaces, using the addresses within their individual overlay. Services exported for external use are made available through a ground gateway based on a configured NAT, including service announcement using the DNS[8]. Per-mission dynamic routing provides network fault tolerance and supports multipath communication, using the X-Bone's isolated secure overlay routing with embedded host virtual routers supporting end-to-end path selection.

Note that DynaSat deploys overlays based on resource availability, but does not itself optimize that deployment. Such optimizations, including least-loaded processor selection, optimal camera sharing, *etc.*, can be supported by plug-in extensions, *e.g.*, between the resource discovery and configuration operations. DynaSat treats provisioning

as optimized by external functions, and operation as optimized using internal dynamic routing algorithms or within user applications running within a mission.

The following section presents the DynaSat architecture and explain how it achieves its goals, discuss its key capabilities of security and fault tolerance, and review the performance of key aspects of the resulting system.

II. DynaSat Architecture

The DynaSat information architecture presents each mission with its own securely isolated IPv6 overlay network. The mission sees a conventional (OSI Layers 3-7) Internet protocol suite with its popular existing programming interfaces. Mission programmers use the same Internet programming environment they are already used to. DynaSat ensures that each mission overlay is robust and secure, and includes common services such as the DNS and distributed file systems. The overlay also supports external access (where activated) *via* a NAT gateway through services advertised in the public DNS. This section outlines this architecture as it exists on top of physical (OSI Layer 1) and link (OSI Layer 2) services.

In DynaSat, we use the following definitions:

- Node – a single physical system, *e.g.*, a satellite or ground computer.
- Cluster – a group of nodes, subsets of which can participate in one or more missions.
- Mission – an integrated computation, communication, and sensing task, *i.e.*, a distributed system that provides a desired result.
- Module – a node that hosts computation, communication, or sensor resources that are used in missions.

The following are the network components of a DynaSat system:

- Space nodes:
  - F6 cluster modules – space nodes with only F6 network links
  - F6 BGAN router – F6 cluster modules that include both F6 and Inmarsat BGAN[9] network links
  - F6 Relay routers – F6 cluster modules with additional direct ground network links
- Ground nodes:
  - F6 Ground modules – ground nodes that participate directly in F6 clusters
  - External nodes – ground nodes that access publicly-available F6 cluster services, but do not participate directly in F6 clusters

A. Base Layer Configuration

DynaSat assumes that the F6 space and ground Layer 2 (link) and Layer 3 (network) layers are preconfigured as separate subnets and that basic routing is available between these subnets (Figure 3). A separate, non-F6 external network may exist for nodes that import or export services to the F6 system (also Figure 3). The BGAN and relay routers forward packets between the two F6 subnets.

IPv6 addresses are preinstalled on all F6 modules. On space modules, these can be modified after launch by the network administrator through conventional secure network management or potentially through the telemetry tracking and command (TT&C) channel. IPv6 self-configuration presents a complexity and vulnerability not necessary in this environment.

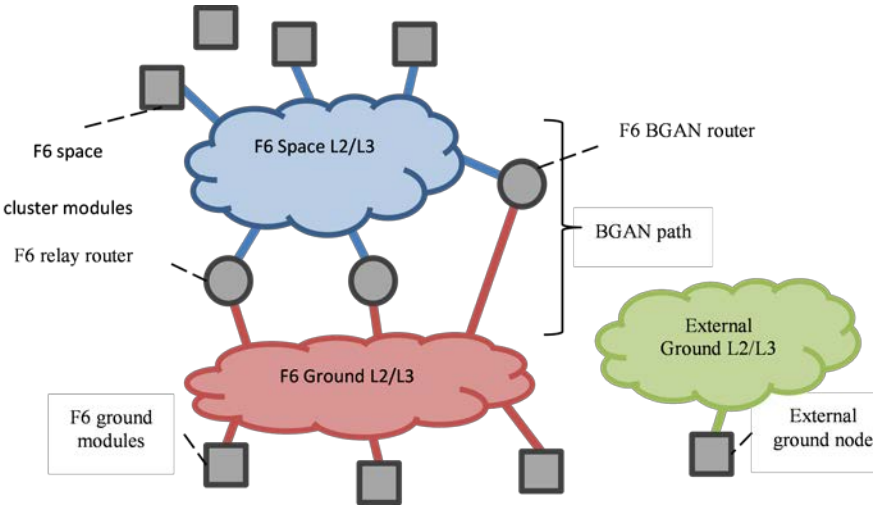

Figure 3 Space and Ground Layer 2 and Layer 3 networks.

The base network needs to exchange packets between the ground and space IP subnets to support monitoring functions as well as to enable DynaSat configuration.

This path can be supported one of two ways:

- *Static routing* – all space and ground nodes use the BGAN router as their default route

- *Dynamic routing* –space and ground nodes use dynamic routing
  - *Secure dynamic routing* – protected using secure multicast IPsec
  - *Insecure dynamic routing* – not protected

Dynamic routing allows base network interactions to operate over alternates to the BGAN path, either for efficiency (lower delay, higher throughput) or to operate through BGAN link outages. Dynamic routing also requires network manager monitoring to reduce the impact of an active attack on the base network's routing system. This is needed with secure dynamic routing to protect against a rogue member of the multicast IPsec group, or with insecure dynamic routing to protect against an arbitrary rogue module.

B. Layer 3 Overlay Configuration

DynaSat uses the X-Bone software system which is capable of deploying overlays of any indicated topology. DynaSat overlays are always deployed with a specific topology – a fully connected mesh among the space subnet members and a fully connected mesh among the ground subnet members (Figure 4, left). Not all modules are a member of any single overlay. Each host includes a virtual embedded router (shown on only one node, for simplicity); this internal virtual router uniquely enables dynamic routing with host virtualization[10].

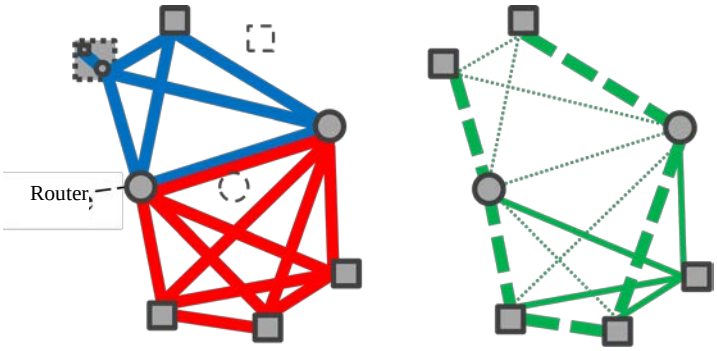

Figure 4 DynaSat space and ground overlays.
Left shows full mesh, right shows backup paths (path shown dashed; down links shown dotted).

These meshes emulate shared-access subnets using unicast (point-to-point) IPsec protection. Although IPsec supports group keying for multicast links, the DynaSat overlay architecture uses only direct links to reduce the potential vulnerability of needing dynamic group key management, which is a known difficult challenge.

The effect of these two separate meshes is that every space node has a direct overlay link to every ground relay module and every ground node has a direct overly link to those same relay modules. This allows routing in the overlay to control its egress to the ground, even when its choice is inconsistent with dynamic routing in the base network. Together they provide similar control that would be achieved with two separate Autonomous Systems (ASes), where routing within the ground node and space node could separately determine the preferred AS transit. F6 subnets are sufficiently small that an AS-based solution is not required.

In DynaSat, these two meshes are configured as a single topology (Figure 4, left). This avoids redundant space/ground tunnels. When dynamic routing is supported in the base network, it also allows paths that cross the ground/space interface multiple times. This might be the case if either (but not both) the ground or space subnets became temporarily partitioned (*e.g.*, as shown by the highlighted path). This latter argues for a single mesh over an AS-based solution, which would prohibit such a path because it would cross a single AS boundary multiple times.

DynaSat uses the X-Bone distributed overlay deployment and management system. The X-Bone system uses two basic components:

- *Resource Daemon (RD)* – a software service deployed once on each network node that coordinates resource allocation and translates generic commands into OS-specific equivalents to configure resources and networking inside that node; such resources can be on space nodes or ground nodes
- *Overlay Manager(OM)* – a software service that deploys and manages one or more overlays by interacting with RDs on located on the nodes

These components are managed by users, either through a web interface or directly using the X-Bone-API[11] TCP port (Figure 5). Users direct the OM to deploy and manage an overlay; the OM configures the components of that overlay through RDs on individual devices.

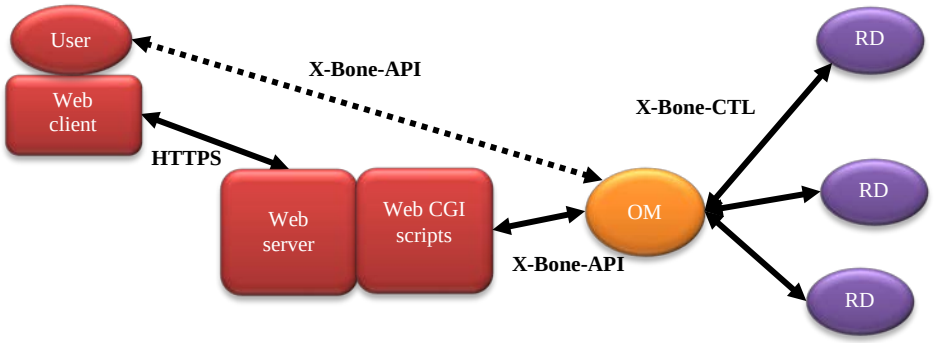

Figure 5 User overlay deployment.

The system employs OM cold spare backups using distributed state files and RD recovery using local storage. It also uses heartbeats to ensure that RD resources are released when all OMs associated with an overlay have failed. The system configures local network resources and deploys and manages node services and user applications using a user-provided script.

DynaSat applies the X-Bone architecture as an overlay on top of a particular module configuration. DynaSat F6 modules are nodes with the following internal architecture (Figure 6):

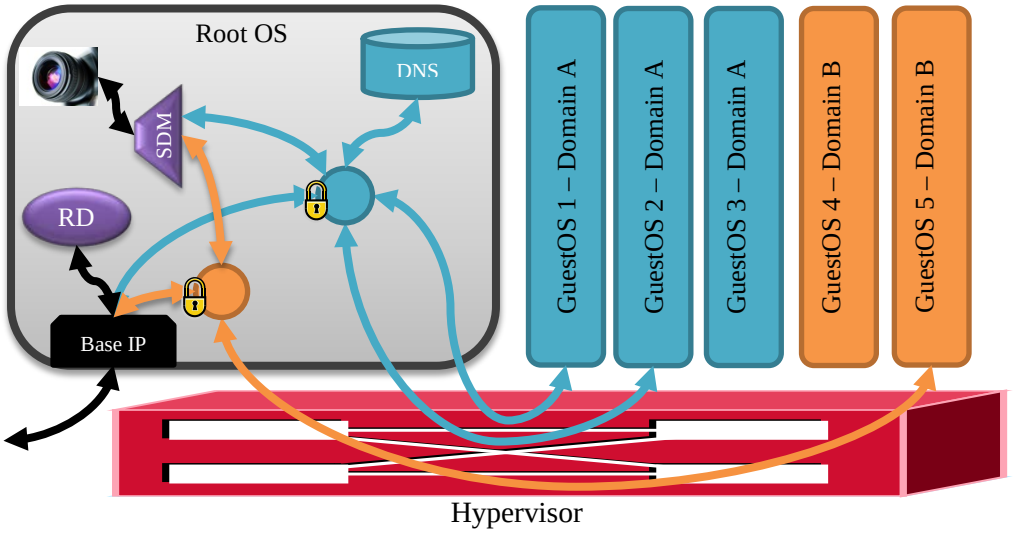

Figure 6 DynaSat F6 module internal network architecture.

- Devices are networked and replicated by a custom scheduled device multiplexer as virtual interfaces
- Guest OSes are replicated by a hypervisor and represent virtual hosts
  - In hypervisors supporting dynamic guest OS creation, guest OSes may be created/destroyed for each overlay.
  - In hypervisors requiring static boot-time configuration, guest OSes are pre-assigned to a particular security domain, and can be reused for different overlays within that domain only.
- Network interfaces are replicated by a virtual router.
  - There is one such virtual router for each overlay, and it interconnects the guest OSes, virtual device interfaces, and internal services (*e.g.*, DNS) of that overlay.
  - All traffic from that virtual router that leaves the module is IPsec encrypted; among virtual devices within the module, hypervisor switching and the use of a secure root OS ensure protected communications without the overhead of internal IPsec.

In DynaSat, a hypervisor and guest OSes assist in partitioning the network interfaces and routing tables. In the X-Bone, this partitioning is typically supported inside a single OS using Clonable Stacks[12]. All these components are realized inside the root OS except for virtual hosts, which are implemented as guest OSes managed by a hypervisor.

Given this module architecture, a DynaSat overlay is a conventional X-Bone overlay network composed of devices (virtual hosts), guest OSes (virtual hosts), and virtual routers interconnected by tunnels[13]. That overlay is specified with a topology of two fully-connected subnets of tunnels as a single end-to-end overlay (Figure 4).

DynaSat benefits from the X-Bone's ability to support a complete end-to-end overlay using IPsec transport-mode IP-in-IP tunnels, which uniquely supports use of conventional dynamic routing within the tunnels. The result provides a complete, compartmentalized security domain with its own network services, including dynamic routing.

Further properties of aspects of this architecture are:

- *Overlay deployment system:*
  - Nodes are configured *via* a single trusted Resource Daemon inside the root OS, which is part of the Trusted Computing Base (TCB).
  - Overlays are deployed by a coordinating Overlay Manager that resides in the same security domain(s) as the target overlay (the OM is not shown in Figure 6; it would function like the RD, but operates only within given security domains inside the root OS).
- *Security:*
  - DynaSat deploys each mission as a secure, independent end-to-end network overlay with associated services running internally.
- *OS virtualization and hypervisor:*
  - OSes are virtualized by a hypervisor that also provides constrained internal interconnection.
  - The root OS needs support for IPsec, dynamic routing, QoS, the RD, and associated RD services (*e.g.*, virtual routing, the SDM, *etc.*); guest OSes need only support IPv6, basic network auto-configuration (*e.g.*, IPv6 auto-configuration, DHCPv6 stateless auto-configuration), and remote execution (if automatic distributed shared network configuration, user configuration, and/or mission application deployment is desired).
  - OSes are either deployed as needed for each security domain, or pre-assigned to pre-defined security domain pools.
- *Device virtualization:*
  - Devices are virtualized by a custom scheduled device multiplexer (SDM).
- *Resource reservation:*
  - QoS reservations are managed by the OM so that each tunnel reserves base network capacity as needed, or shares available capacity.
  - CPU and OS resources are reserved *via* the hypervisor scheduling mechanism
  - Device resources are reserved *via* the SDM.
- *Dynamic routing:*
  - Dynamic routing in the overlay operates independently of dynamic routing in the base network. The former allows per-mission prioritization of available paths independently of the base network's support for persistent routing.
- *Exporting/importing overlay services (via a NAT):*
  - A DynaSat overlay is a complete end-to-end overlay network in which all services are local to the overlay. If external interaction is needed, DynaSat deploys a guard similar to a NAT gateway, so external nodes can exchange services with an overlay individually or as a single IP address.

C. User View

DynaSat presents each mission with a network view of a single, private IPv6 subnet implemented as an overlay (Figure 7). Modules in that subnet are reachable only over IPsec-protected tunnels. Paths through the overlay are determined by dynamic routing, configured by the DynaSat system. All other services are presented inside the overlay and visible only internally, *e.g.*, distributed files, mission DNS, *etc.* Services accessible externally are presented *via* a NAT gateway that serves as a security guard, where such services are announced at the NAT in the public DNS.

DynaSat allows the use of dynamic routing in the base network in which the particular space/ground gateway can be selected using conventional cost metrics (Figure 4, right). Separate overlay routing selects paths using explicit intermediate points inside the overlay

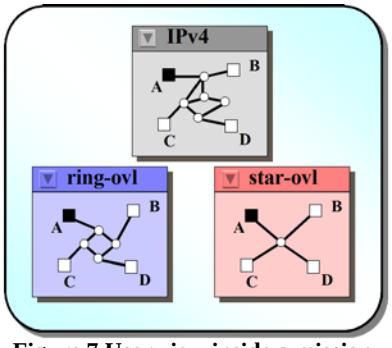

Figure 7 User view inside a mission.
Three missions are depicted here.

routing processes independently of the routing in the underlay.

The remainder of the user view is conventional IP networking using conventional network protocols and APIs. Multicast is supported using serial copy, in which a single packet is copied and emitted on each outgoing IPsec'd link. Within an overlay, user processes sit behind an IPv6 address, as do module devices.

D. Hypervisor issues

To provide a full separation of execution environment, DynaSat assumes a MILS[14] hypervisor, initially focusing on the Wind River MILS Hypervisor product. Like all hypervisors, it allows multiple guest operating systems to run concurrently on host hardware. The hypervisor provides a virtual platform for guest OSes and manages their execution[15]. DynaSat's current target hardware platform is the Space Micro processor board, a 1.2GHz PowerPC single board computer with 512MB RAM and 8GB flash that is available in both space-qualified and engineering models (the latter electrically identical and for ground-only use).

Wind River MILS Hypervisor is a Type 1 embedded hypervisor, known as bare-metal hypervisor, with a very small code and memory footprint, minimal latency for device access, and deterministic capabilities and optimizations for performance. It runs directly on the host hardware (Figure 8). A guest OS runs above the hypervisor.

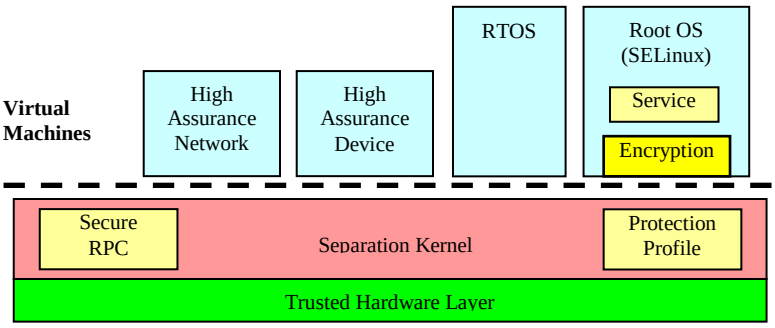

Figure 8 Wind River hypervisor architecture.

DynaSat extends Wind River's MILS support to create securely separated mission-specific overlay networks. The hypervisor enforces CPU and memory separation, and controls network interaction. Communication between guest OSes uses a controlled internal virtual Layer 2 Ethernet switch for TCP/IP communication. Switching uses a socket-like API and shared memory for fast, zero-copy communications between OSes.

The hypervisor limits communication between OSes as specified in the Virtual Machine (VM) XML Schema Description (XSD) file. Some hypervisors can dynamically vary this configuration under control of the root OS, and can deploy OSes dynamically. The Wind River MILS hypervisor supports only boot-time static configuration. To support use of either approach, DynaSat configures each guest OS with a single (virtual) network interface and connects each to the root OS directly (Figure 9). The root OS serves as the relay (and guard) between any guest OS and the (real) network interface of the module.

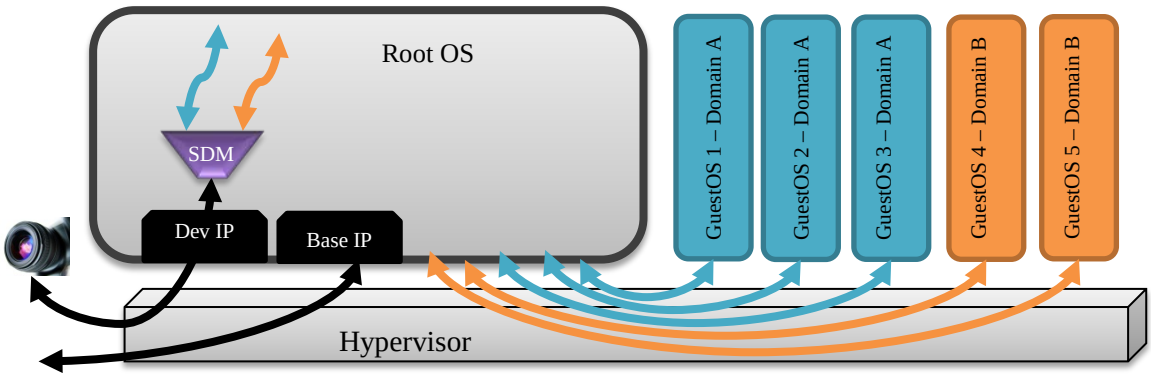

Figure 9 Hypervisor communication paths.

Device networking is modeled the same way as a single guest OS, *i.e.*, each has a device network interface connected only to the root OS. For devices, this interface is treated as a trusted channel, and its access is multiplexed

to support dynamic sharing inside the root OS using DynaSat's Scheduled Dynamic Multiplexer (SDM). The SDM provides the dynamic scheduling that could be supported in a dynamically-reconfigurable hypervisor, but avoids the need to rely on this capability.

For each mission, the RD (running inside the root OS) creates a virtual router to interconnect resources, including the guest OSes and SDM ports specific to that a single overlay. In a statically-configured hypervisor, guest OSes can be reused for successive missions within their assigned domains, providing a pooled resource. In a dynamically-configured hypervisor, guest OSes would be created or destroyed as needed.

The root OS interacts with the hypervisor to manage the amount of shared resources available to each guest OS, and for some hypervisors can request that a guest OS be moved across the network to a different module. The benefit of this approach is not clear *vs.* user-level process migration – especially in MILS hypervisors where dynamic guest OSes are not generally available – so DynaSat is continuing to explore possible VM migration.

Any security and access violation are logged and may invoke specific hypervisor actions based on the type of violation. Both MILS kernel and each VM (including the root OS) can have audit logs, which can be filtered and collected by the security audit subsystem.

E. Support for shared devices

Devices such as sensor packages and cameras attached to a F6 module need to potentially be shared temporally among multiple users. DynaSat assumes devices are already – or can easily be converted to be – IPv6 network addressable. In order to enforce the exclusive use of the device at a particular time, DynaSat adds a Scheduled Device Multiplexer that connects the device to the appropriate overlay at the appropriate time and/or location (Figure 10). The SDM allows a single networked device to be securely shared by multiple missions in separate overlays. This sharing can be temporal (based on GPS time), spatial (based on GPS location), or some combination of the two. Sharing can also be simultaneous, where a single camera can allow separate high- and low-resolution images to be obtained concurrently by missions in different domains, but that level of sharing is typically easier to manage as a separate (concurrent) network interface, *i.e.*, emulating separate virtual devices.

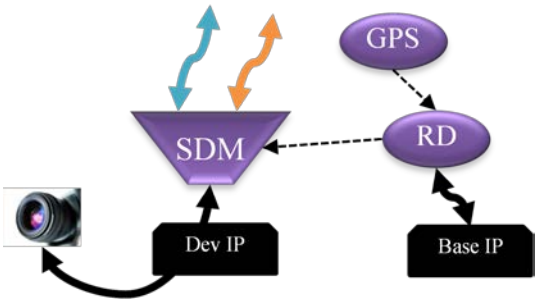

Figure 10 SDM architecture.

Applications not on the same overlay as the device, or applications whose time or location does not allow will have no access to the device; they will exchange packets with an unconnected socket. The RD running in the base host for the F6 module is responsible for configuring the SDM and scheduling it using GPS information.

For network management purposes, the device may also present a control interface. This is assumed on a separate network interface so that it could be isolated to a separate overlay specifically for the purposes of management.

F. Infrastructure

DynaSat relies on several infrastructure components, not all of which affect a given F6 module (Figure 11). The Overlay Managers may be deployed on a subset of F6 modules or outside the F6 system. A DNS may be deployed inside each overlay, but there also is a base network DNS that supports the hostnames used inside X.509 certificates. There may also be a DNS on the external network used to allow external nodes to access public mission services inside the F6 cluster. The X-Bone also relies on an address server to allocate independent IP address spaces for each overlay, although this is less important for IPv6-only deployments where large ranges can be pre-allocated to each OM. Finally an external gateway process may be

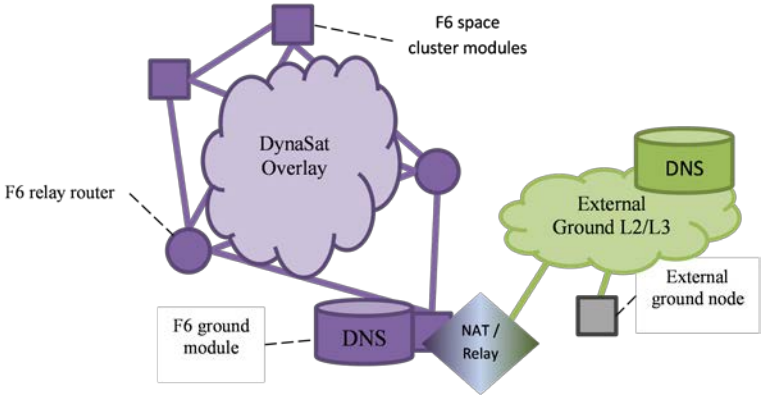

Figure 11 Space and ground network interconnection.

required to support relaying between a mission overlay and the external nodes to support public services or to allow the mission to access public resources, so DynaSat includes a pre-deployed NAT system that OMs configure as needed to enable exchanges across the mission/public network boundary. This NAT is deployed together with a configured public DNS where publicly available services are registered if desired.

DynaSat uses NATs and NAT-like relay services to enable mission services to use services outside the F6 cluster, as well as to allow nodes outside the cluster to access desired services of the cluster. In either direction, this process relies on both traditional network address and port translation (NAT/NAPT), as well as proxy relays that may serve as endpoints of transport protocol connections in both directions[16].

NATs and their associated relays serve as guards to manage communication between the F6 cluster and external nodes. They are deployed and configured inside the root OS on F6 ground nodes and rely on advertised services as indicated in the DNS on either side of the cluster boundary.

III. Analysis and Discussion

DynaSat provides its distributed information architecture with two key features – security and fault tolerance. The following is an analysis and discussion of each of these considerations.

G. Security

The DynaSat architecture extends the MILS security model to distributed systems[17,18]. This is accomplished by creating 'separation networks' that securely contain distributed applications. Separation networks are a class of virtual network that strictly limit what an application can access. This is achieved by creating secure proxies for network naming, routing and device access for each application. DynaSat constructs separation networks using standards-based tool suites: TLS, X.509 and IPsec. These suites were developed under the control of the Internet Engineering Task Force (IETF). IPsec development in particular is partially overseen by the National Security Agency (NSA), which has judged Suite B IPsec encryption to be sufficient for protection of classified material. An F6 network consists of both ground-based and space-based sub-networks. Each module (host) in an F6 network provides one or more virtual machines for use by applications. DynaSat permanently associates each virtual machine with a particular security domain (level). An F6 application is distributed across a set of virtual machines that reside in the same security domain.

DynaSat presumes the existence of a base IPv6 network that interconnects all F6 modules, both terrestrial modules and space modules. Base network management security uses TLS, which relies on X.509 certificates. DynaSat does not provide the following COMSEC security features on the base network: emission security, traffic-flow security and transmission security; they are generally provided by the link layer. DynaSat does provide crypto-security for all messages sent across the base network by F6 applications and by all DynaSat control messages. DynaSat is responsible for configuring the routing and addressing of the base network and ensuring its function.

Base routing can use either static or dynamic routing, with static being more secure but less robust. If present, dynamic routing can be run either insecurely or can be secured using multicast IPsec using GSAKMP. GSAKMP presents an additional overhead, but group membership is expected to be stable enough that additional communication overhead is not a concern. All dynamic routing requires network manager oversight, to detect when anomalies occur, such as when a rogue module renders other modules unreachable for extended periods. Upon detecting such a routing error, secure routing would need to execute a group key change (omitting the rogue element). In such cases, insecure routing would reconfigure to the static BGAN default route. Both require the intervention of an external participant, *e.g.*, a network manager (human or automated) to detect and signal the event. This base layer routing does not interfere with or interact with overlay routing, which is distinct because it forwards traffic directly to virtual routers on the BGAN or relay routers under control of an independent dynamic routing mechanism. Base network management security is protected using TLS, which relies on these X.509 certificates. Conventional network management software can be used to manage the base network configuration, either in-band or over the TT&C channel (the latter if present and supported).

A distributed F6 application runs in a dedicated overlay that is securely isolated from other application overlays. The application overlay lies in the same security domain as the application that uses it. DynaSat constructs each application overlay network complete with its own names, addresses and routes. Logically, a router exists for each application running concurrently in a module. Routing tables are set up by the RD when each overlay is created. Routing functionality is located solely in the RD virtual machine, which is isolated from application virtual machines by the underlying separation kernel (hypervisor). Each packet going out of an application virtual machine must pass through its associated router in the RD before being placed onto the network or being sent to a local device. Each packet that enters an application virtual machine must also first pass through that router. This ensures

that only those routes constructed for it by the RD are usable by an application virtual machine. The RD applies MAC routing constraints appropriate for an application prior to that application's startup. Changes to routing can be controlled by RDs. For example, overall priority in using BGAN versus other specific modules with ground relays. This is accomplished by changing virtual link weights in an overlay's routing topology. In the event that a particular application must perform 'write down' or 'write up' operations, DynaSat requires that application-specific guard processes be installed. A guard process resides simultaneously in multiple application overlays and it monitors information transfer between overlays and so between security levels.

The various virtual machines running in a module at any given time are securely isolated from one another by adopting a MILS-style hypervisor, also called a separation kernel[14]. For prototype development, the Wind River MILS hypervisor was selected. It provides secure separation of virtual machines. Other hypervisors exist, such as the open source Xen project[15]. These are more widely used, but require modification to guarantee secure separation.

A file system in DynaSat is a service made available by virtual machines to applications. By construction DynaSat ensures that an F6 file system is permanently associated with a single security domain. As a consequence, an F6 file system does not require labeling. When an application concludes, it can leave state behind in a file system. That does not constitute a security breach, since any following application must reside in the same security domain. DynaSat does not allow applications in differing security domains to share file systems, although guard processes can transfer file data from one security domain to another.

Devices in DynaSat are directly addressable *via* IP and are given DNS names. This simplifies both application writing and process relocation. All an application process requires is the name of the device, which is then translated by the application's overlay DNS service into the proper address. Routing functionality in that overlay now takes over and directs application packets to the correct module and internal device network interface. Devices are traditionally directly accessible only to the host in which they reside. To make devices truly network addressable requires three things:

- The physical device driver no longer presents an API to local applications.
- A corresponding network device driver presents to local and remote users a network API. The network driver utilizes the physical device driver.
- Device schedule- and access-control must allow for both local and remote use.

Because devices may be used by applications having multiple security domains, the network device drivers are considered to be part of the TCB and logically located in the RD of the module where the device is physically located. Device access and scheduling is a function under control of the RD, which applies MAC constraints when an application is instantiated, and manages the Scheduled Device Multiplexer (SDM) accordingly.

H. Fault tolerance

DynaSat is capable of handling a variety of faults in the system without significant operational impact. The goal of the F6 program is to develop a system that can autonomously reconfigure the network routing paths, and move processes or applications to other modules. This section describes the major categories of potential faults and how DynaSat can mitigate them.

Link failure is the inability of two modules to directly communicate. This can occur for several reasons, such as hardware faults, wireless signal disruptions such as interference or loss of line of sight, or lack of sufficient signal strength due to range. When the link failure is intermittent, DynaSat has the option of working around the impairment in several different ways. Intermittent failures are likely to be somewhat common, both among the modules in the space cluster, as well as the ground links. During the course of an orbit, the modules will move relative to each other, causing the signal strength of the wireless networks to vary, and the cluster itself will move out of range of ground stations for the LEO ground links. DynaSat will mitigate these disruptions using the techniques of *striping*, *pause/resume*, and *dynamic routing*. *Striping* is a method where the receiver makes use of multiple network links simultaneously to transfer data in parallel by requesting chunks either from multiple sources, or where the source is dual-homed on multiple network links. The receiver tracks which chunks have been received, and can request the same chunk from a different source, or different network link, if that chunk is not received in a timely manner. The BitTorrent protocol is one widely used example, but for the purposes of DynaSat is likely too heavyweight. Instead, applications can make use of the FTP and HTTP/1.1 protocol features that allow for clients to request a chunk of data at a specified byte offset. *Pause and resume* is somewhat similar to striping in that the FTP and HTTP/1.1 protocols allow a client to start a transfer at a specified byte offset rather than at the beginning of a file. The file transfer application can save persistent state information about progress, perhaps by saving to disk or maintaining it in memory, and resume the transfer where it left off at some later point. This technique is also useful in the case where a file grows over time, such that the entire file need not be transferred—just the new portion of the

file. *Dynamic routing* hides the link failure from the application by selecting an alternate path to the destination, without the need for the application to take action. DynaSat uses the X-Bone overlay management system to provide this service. The F6 cluster will consist of several LEO ground links, which can be used when the ground stations are in range, and Inmarsat BGAN as a fallback. DynaSat will route traffic through the BGAN when no other path is available in order to ensure connectivity to the ground. Additional, dynamic routing can be used among the nodes in the on orbit cluster in the case where L2 connectivity between two nodes is lost but there is an indirect path.

Node failure is when a module or virtual machine running in a module stops functioning properly. DynaSat is able to recover from faults of this nature either by attempting to restart the node/VM, or through the use of cold and hot spares. The X-Bone Resource Daemons running in the root OS save their configuration to disk so that the configured overlay networks can be recreated after a node reboot. The DynaSat application setup script is used to restart applications on the node. When a node/VM is unable to be restarted, DynaSat supports application migration to hot or cold spares, as the user desires. When the overlay is configured, the user has the option of allocating additional nodes. From DynaSat's perspective, it does not distinguish active from inactive nodes. It is at the discretion of the user to craft the application setup script to select which nodes to activate, and which to save as spares. The application setup script can utilize the IP address assigned to the node as a unique identifier to aid the selection. This allows the same application setup script to be used on all nodes in the overlay. Software component failure is largely a subset of the node failure issue. In the DynaSat system, software component failure is handled in a nearly identical manner as described above for the node failure case. The typical approach to mitigate router failures is via the use of multiple routers linked together using a protocol such as Common Address Redundancy Protocol (CARP) or Virtual Router Redundancy Protocol (VRRP). Using this technique, a set of routers use the same IP address such that nodes utilizing the routers as a gateway need not be concerned with attempting to failover when a single router fails. In that situation, one of the other routers receives and forwards packets. DynaSat's overlay architecture largely supplants the need for this type of approach since it creates overlays above L3 and uses dynamic routing to move packets over available network links. Thus, even though the ground link routers may all have different L3 network addresses, the endpoints are isolated from the actual network addresses.

IV.Performance

Basic measurements verify the volume of message exchanges for various X-Bone functions. Our approach does not modify the kernel OS packet handling mechanisms. Each packet traverses the kernel at least two times, however, so the maximum packet rate is reduced by a factor of around 2. The performance of both IPv4 and IPv6 traffic is 922 Mbps in the base network, 896 in an overlay, and drops to 175 Mbps when IPsec is included. The CPU load for non-IPsec transfers was 0.05; for IPsec DES-encrypted transfers the load was 0.18. Although IPsec performance is much lower in software, hardware-assisted IPsec can easily support 10 Gbps.

The X-Bone discovery process is performed either on demand by the operator or as a part of other OM functions, and currently takes approximately 7 seconds, based on current timeouts. Overlay creation takes approximately 13 seconds using current timeouts. The smallest overlay topology of two hosts with a router between requires 61 packets for configuration, and 29 for feedback to the manager requesting the overlay. Tearing down the same overlay requires 27 packets.

Our approach uses existing IPsec and IP headers, together with two layers of IP-in-IP tunneling. For IPv6 packets, assuming L2 headers of approximately 18 bytes, TCP/IP headers of 60 bytes (lower for UDP traffic), the total minimum uncompressed original packet size is expected to be around 78 bytes, so that these 100-byte packets are really 178 bytes and application layer packets of 100 bytes will thus already have 78% additional overhead due to the L2/IP/TCP headers, resulting in only 56% overall efficiency. Our approach adds an IPsec header (52 bytes, which varies slightly depending on algorithm), and between 2 and 4 additional IP headers, which translates to an uncompressed additional 132-212 bytes, resulting in an overall packet size of 310-390 bytes, and 20-25% overall efficiency.

The required L2 data capacity for 100 messages/sec of 100 bytes payload each is thus approximately 250-312 Kbps -- to support a user payload capacity of 80 Kbps (around 25-30% efficiency). The overhead remains the same for larger packets, so the efficiency increases as payload size increases. Further, the overhead could decrease with minimal compression, *e.g.*, even using IPv4 headers for the overlay system would result in no additional CPU effort for compression/expansion and no storage for conversion tables but the total bandwidth reduces to 216-280 Kbps and the efficiency increases by around 5%. Further compression could achieve even higher efficiencies, compressing not only our headers but also the original TCP/IP headers. The number of intermediate nodes does not affect these numbers.

Simple IPsec would add around 52 bytes (depending on algorithm) to a base packet, and typically operate in tunnel mode (adding another 40 bytes for IPv6), so the overall additional overhead for a simple secure network would be 92 bytes on top of 178, for a total of 270. DynaSat adds only 1-3 additional IPv6 headers for a total of another 40-120 bytes, resulting in a relative efficiency of 70-85%.

V. Conclusion

DynaSat uses Internet overlay networks and networked devices to support an information architecture intended for the DARPA F6 distributed satellite program, but with potentially general use to any distributed system with fractionated resources. Its network uses a dual-mesh base network to control ground relay selection, and overlay networks to provide independent, per-mission dynamic routing control. Its architecture uses a COTS hypervisor together with COTS guest OSes to support MILS security, where only the configuration and resource control system and device sharing system operate in a single, trusted OS. Applications within a mission are provided a secure, private network by the deployed overlay, using COTS Internet protocols and programming interfaces. A multiplexer in the trusted OS virtualizes shared devices for scheduled use in separate mission overlays. The system also supports protected access to external networks through a NAT relay and published services. The resulting information architecture provides mission configuration in seconds, and supports fault tolerance both in the network and its supporting distributed services. To achieve these features, DynaSat typically consumes only 15% of the network capacity (over half of which for COTS encryption), and negligible CPU resources.

Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Grant No. NNA11AB05C. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the U.S. Government.

References

[1]Brown, O., P. Eremenko, "The Value Proposition for Fractionated Space Architectures", *AIAA Space Conference*, AIAA, 2006, pp. 21.

[2]Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," *Computer Networks*, Elsevier, July 2001, pp. 117-135.

[3]Finn, G., "An Integration of Network Communication with Workstation Architecture," *Computer Communication Review*, Vol. 5, Issue 2, ACM, Oct. 1991, pp. 18-29.

[4]Wang, Y., J. Touch, "Application Deployment in Virtual Networks Using the X-Bone," *DANCE: DARPA Active Networks Conference & Exposition*, IEEE, May 2002, pp. 484-491.

[5]Ardaiz Villanueva, O., J. Touch, "Web Service Deployment Using the X-Bone", *Spanish Symposium on Distributed Computing* (*SEID)*, 2000.

[6]Touch, J., Y. Wang, V. Pingali, L. Eggert, R. Zhou, G. Finn, "A Global X-Bone for Network Experiments," *IEEE Tridentcom*, IEEE Communications Society, 2005, pp. 194-203.

[7]Fujita, N., J. Touch, V. Pingali, Y. Wang, "A Dynamic Topology and Routing Management Strategy for Virtual IP Networks," *IEICE Trans. on Communications*, VE89-B, Num. 9, Sept. 2006, pp. 2375-2384.

[8]Gulbrandsen, A., P. Vixie, L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, Internet Society, 2000.

[9]Franchi, A., A. Howell, and J. Sengupta, "Broadband mobile *via* satellite Inmarsat BGAN", *Seminar on Broadband Satellite: The Critical Success Factors Technology, Services and Markets*, IEE, Oct 2000.

[10]Touch, J., T. Faber, "Dynamic Host Routing for Production Use of Developmental Networks," *ACM International Conference on Network Protocols (ICNP)*, ACM, Oct. 1997, pp. 285-292.

[11]Yang, Y., J. Touch, G. Finn, "The X-Bone API," USC/ISI Technical Report, ISI-TR-2005-611, University of Southern California/Information Sciences Institute, Dec. 2005.

[12]Zec, M., "Implementing a Clonable Network Stack in the FreeBSD Kernel", *Usenix*, 2003, pp. 137-150.

[13]Touch, T., L. Eggert, Y. Wang, "Use of IPsec Transport Mode for Dynamic Routing," RFC 3884, Internet Society, 2004.

[14]Alves-Foss, J., W. Harrison, P. Oman, C. Taylor, "The MILS Architecture for High-Assurance Embedded Systems," *International Journal of Embedded Systems*, Springer-Verlag, February 2005, pp. 179-190.

[15]Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization", *ACM Symposium on Operating systems principles (SOSP)*, ACM, 2003, pp. 164-177.

[16]Srisuresh, P., M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, Internet Society, Aug. 1999.

[17]Rushby, J., "The Design and Verification of Secure Systems", *ACM Symposium on Operating System Principles (SOSP)*, ACM, 1981, pp. 12-21.

[18]Rushby, J., B. Randell, "A Distributed Secure System", *IEEE Computer*, Vol. 16, No. 7, IEEE, 1983, pp. 55-67.