A Component-Based Active Network System for Satellite Platforms

S. Zachariadis[a], S. Bhatti[a], W. Fritsche[b], G. Gessler[b], P. Kirstein[b], K. Mayer[b] and L. Sacks[c]

[a]Department of Computer Science University College London, London, UK
[b]IABG mbH, Ottobrunn, Germany
[c]Department of Electronic & Electrical Engineering, University College London, London, UK

Abstract

Active networks are networks that can be reprogrammed by injecting customized programs into some of the nodes of the network, which change the computation that is performed on packets flowing through them. Active networks have been successfully used to offer adaptable routing mechanisms in disaster scenarios, to dynamically compress data before transmitting, etc. In this paper, we discuss the advantages that an active approach would offer for satellite network systems. We describe issues with the deployment of an active networking platform on a satellite. We detail the design and implementation of a component based active networking platform, which is able to dynamically deploy, instantiate and reprogram software components on a satellite system. We also show how the platform is used to build dynamic media transcoding services and test the implementation of our system.

1 Introduction

The Internet currently provides communication on a best-effort basis. The lack of support for enforcing QoS requirements makes the delivery of multimedia services difficult; as those services have strict latency and bandwidth requirements, a violation of those requirements can result in artifacts, delay or interruption in the multimedia stream. With the number of streams and users set to rapidly increase in the next years, the importance of this problem is likely to also increase.

In terrestrial networks, the operators try to reduce the probability of congestion and its accompanied problems by increasing the capacity of the routers, switches and optical lines, through either upgrading or replacing them, or by installing QoS enforcing software at key nodes in the network. Usually, however, these strategies, are not feasible for satellite networks. The frequency bandwidth per transponder is limited and increasing capacity translates to launching new satellites, which is very expensive with respect to production, launching and operation. Moreover, software systems deployed on satellites usually take a "black-box" approach to software engineering; they are monolithic and inflexible, developed, tested and deployed on the satellite before launching it into space, with little possibility of modification afterwards. This makes deploying new services on the satellite particularly difficult. As more and more multimedia streams such as television are transmitted over satellite, new strategies need to be introduced to tackle the issues introduced by congestion.

This paper presents worked developed as part of the PANAMAS project, funded by the European Space Agency. It utilises technologies projected to become commercially available within the next 10 years and illustrates the advantages of deploying active networking software into a satellite system. The paper continues by discussing how active networks could be used to solve a number of the issues described and what special requirements a satellite platform would introduce. Section 3 gives an introduction to SATIN, the platform that was used to implement a demonstration model of a satellite active network system and Section 4 details a dynamic and programmable transcoding proxy that was built and deployed using SATIN. Section 5 concludes the paper by discussing our approach and outlining ideas for future work.

2 Active Satellite Network Systems

2.1 Basics of Active Networking Systems

An active network is *active*, because it can respond intelligently to changes to the environment and to user requirements. It may also be *programmable* when it provides a well defined API to access and control various resources. One of the problems with most such systems is that the API provided, and hence the functionality offered, are fixed and cannot be changed. The programmable aspect allows the system to assert algorithms (*algorithmic programming*) or policies (*declarative programming*) on the control or content of data flows. An active network is often realised using mobile code. There are two approaches. One is the *capsule* approach, where each network packet carries code that is applied on its payload in the active nodes; the other is the *discrete* approach, where the code is transported and instantiated on the active nodes in a control stream, separately to the data stream. The capsule approach is usually used to introduce and realise new network protocols and exhibits problems with security, performance and scalability. The discrete approach is often used to introduce new services applied on specific streams (e.g. belonging to a specific application or user).

We have witnessed recently in the literature, the development of reprogrammable component based systems. A software component is an encapsulation of functionality, that is offered through well defined interfaces. Components offer a structured way to engineer a software system. Reconfigurable and adaptable component-based systems allow components to be loaded and invoked dynamically. In this project, we decided to take a component based approach because component-based systems allow for encapsulating active networking functionality into discreet components. Moreover they provide a systematic way to load and invoke components.

As such, the main requirements for a component-based active networking system are: (i) The ability to dynamically *deploy* and *instantiate* a component. Deployment is triggered as a result of a control event, which may map to a user or operator action, or even a feedback loop of the operation of the system. Deployment may involve discovering the component to deploy, requesting, transferring and instantiating it into the running system. Instantiation may involve resolving the component's dependencies, checking for potential security issues and allocating memory, usually in a sandboxed environment. (ii) The ability to *control* a component. This implies using the interfaces or API that the component offers to configure it, and pipe streams or individual packets through it.

2.2 Issues with Satellite Active Networks

A satellite active network allows the adaptation of the system to accommodate changes to its requirements, which are caused changes in network conditions or application requirements. The changes cannot always be anticipated at the initial deployment time. The system can adapt by introducing new services, or by adapting the existing ones. Given the longevity of a satellite platform, this benefit is particularly significant.

There are a number of issues with deploying a satellite-based active network system. By introducing this flexibility and essentially invalidating the end to end argument in network design, security must be guaranteed. In particular only trusted nodes (such as those of the satellite operator) should be able to reprogram an active node and components should be trusted and verified to act as advertised and not, for example, crash the node. Moreover, the flexibility of component-based code and programmable active networking, incurs a computational cost on the active node. Although this may not be a problem at the satellite edge nodes, deploying active networking software on the satellite itself is a challenging task because of the limited resources that the satellite offers. It is expected that this problem will be of lesser importance in the future, with advances in technology. We discuss how these issues are tackled by our chosen platform in the next section.

3 SATIN as an Active Networking

In the PANAMAS project, we used the SATIN [6] platform to implement our active network system. SATIN is a component metamodel, which can be used to build adaptable systems. It is instantiated as a middleware system, which can adapt itself or component based applications which run on top of it. Its use as an active networking platform was initially outlined as a case study in [5]. SATIN uses the principles of *reflection* and *logical mobility* to offer adaptational primitives, and offers them as first class citizens to applications. Reflection offers a *meta-interface* to the system, that allows SATIN applications discover which components are available locally, to be notified of changes in component availability, to load and discard components, to discover the interfaces that they offer and to read and write metadata attached to a component. Logical mobility allows SATIN systems to send and receive components, classes, objects and data.

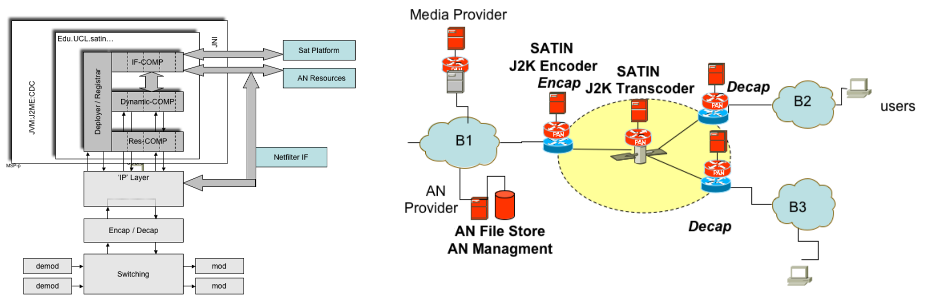
SATIN was designed for mobile devices and devices with limited resources, such as mobile phones and PDAs, and is implemented using Java 2 Micro Edition [4] (Connected Device Configuration / Personal

Profile). It occupies 150329 bytes of memory, and provides the following services which are relevant to an active networking system:

- **Reflection.** The use of reflection allows the system to reason about what it can currently do, dynamically invoke components, use an event service which notifies when there are changes in component availability, and to discover new APIs that are offered by components.
- **Logical Mobility.** The use of logical mobility allows components to be transfered dynamically from one SATIN node to another.
- **Dynamic Component Instantiation.** Components can be discovered, instantiated and invoked dynamically.
- **Advertising and Discovery.** SATIN provides an adaptable framework for component advertising and discovery.
- **Security.** SATIN nodes can employ digital signatures and trust mechanisms to verify incoming components. Moreover, SATIN uses the Java sandbox to instantiate components in.

In conclusion, SATIN is a small footprint middleware system that can be used to build adaptable systems. It provides a minimal runtime API that can be used to discover, request, deploy and invoke new components, and an interface to reason about what individual components or the system itself can do.

4 A Dynamically Deployable and Reconfigurable Transcoding Proxy



(a) SATIN Deployment on a Satellite (b) A Dynamic Transcoding Service

Figure 1: A satellite active networking system

Figure 1(a) shows a deployment of SATIN in the space segment. The IF-Components provide interface services to the satellite platform, to the active networking platform and to the switching layer. The Dynamic-Components encapsulate specific programmable and active services. The Res-Components provide resident services for common tasks of service engineering. The Registrar and the Deployer are components that allow SATIN applications to deploy and invoke new components. Deployment on the ingress node is similar, but the local SATIN instance is also connected with a repository of components, which can be advertised to, discovered by and dynamically deployed on the satellite itself.

Using this architecture, we implemented a transcoding system, that can recode streams dynamically to lower bitrates. An overview of the system is shown in Figure 1(b). In the scenario described in the figure, a media provider is streaming images through a satellite system. The images are encoded in JPEG2000 [1] format at the ingress point, encapsulated, streamed through the satellite and decapsulated at the egress. They are then served to two users at different spot beams. The active nodes are the ingress, space and egress segments. They are controlled by the active networking provider. Users connecting to the egress through B2 utilise a fast link, whereas those connecting through B3 are on a slow link. We simulated the links using the Linux traffic conditioning framework. To simulate satellite connectivity, we induced a 250ms delay on both links. We also limited the connectivity through B2 to 512Kpbs, and connectivity through B3 to 64Kbps. We simulated connectivity to the media server using two RES-Components: an http proxy implementation (10063 bytes) and a manager component (8361 bytes) that is responsible for controlling proxies. Both components can be deployed dynamically to the satellite emulator. The satellite and management platforms were simulated using 154809 and 155156 bytes respectively, including SATIN.

We used the reference implementation of JPEG2000 [2] in Java to encode the images to the appropriate format. We adapted and encapsulated the reference implementation as a SATIN JPEG2000 transcoding component that can dynamically change the resolution as well as the bits used per pixel of the image. The transcoder occupies 789391 bytes. It can be deployed dynamically to a proxy running on the satellite by the network manager, and offers a control interface that can be used to program it dynamically. We used a standard apache httpd server to simulate the media server and Mozilla running on Linux PCs to simulate one client on each link.

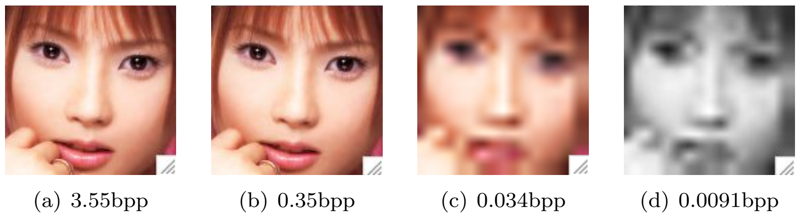


Figure 2: Recoding an image to a lower bitrate.

We conducted a number of tests with this architecture, serving still images and a slideshow. Without deploying the transcoder, the Mozilla instance connected through B2 took approximately 3 seconds to download a 174356 byte image, compared to 18 seconds for the one connected through B3. We deployed the transcoder dynamically, and got Mozilla to display the image through B3 at the same time as the one connected through B2, by either reducing the resolution of the image (which would be suitable for a device with a small screen, such as a PDA), or by lowering the number of bits per pixel of the image, which introduced artifacts (see Figure 2). Hence, users connected through B3 can be offered a fast service in case of congestion, but the quality of the stream suffers.

5 Conclusions

This paper discussed a dynamic architecture for active networks, and showed the potential for its deployment on a satellite system. We also discussed the development of a transcoding service and its implementation for JPEG2000 images. We showed the flexibility that the architecture can offer over traditional signaling protocols and illustrated the advantages that its deployment can have on accommodating changes to the systems requirements, by dynamically improving the QoS that users perceive.

There are a number of recommendations for future work. Currently, the system can handle operations on streams, but not on individual packets; this could be potentially offered by using iptables [3]. Moreover, a better management architecture needs to be developed, that allows the systematic and remote control of Dynamic Components.

Acknowledgments.

The authors would like to thank the European Space Agency for funding this work, under contract number 18376/04/NL/AD.

References

[1] M. W. Marcellin, A. Bilgin, M. J. Gormish, and M. P. Boliek. An overview of JPEG-2000. In *DCC '00: Proceedings of the Conference on Data Compression*. IEEE Computer Society, 2000.

[2] The Swiss Federal Institute of Technology-EPFL. JPEG2000 Reference Implementation (Java) Version. http://jpeg2000.epfl.ch/, 2002.

[3] The Netfilter Project. netfilter/IPTables. http://netfilter.org/, 1999.

[4] Sun Microsystems, Inc. Java 2 Platform, Micro Edition. http://java.sun.com/j2me/, 2000.

[5] S. Zachariadis, C. Mascolo, and W. Emmerich. Exploiting Logical Mobility in Mobile Computing Middleware. In *Proceedings of the IEEE International Workshop on Mobile Teamwork Support, Collocated with ICDCS'02*, pages 385–386, July 2002.

[6] S. Zachariadis, C. Mascolo, and W. Emmerich. SATIN: A Component Model for Mobile Self-Organisation. In *International Symposium on Distributed Objects and Applications (DOA)*, Agia Napa, Cyprus, October 2004. Springer.