APM

POSEIDON HOUSE • CASTLE PARK • CAMBRIDGE CB3 0RD UNITED KINGDOM
+44 1223 515010 • Fax +44 1223 359779 • Email: apm@ansa.co.uk • URL: http://www.ansa.co.uk

ANSA Phase III

FlexiNet White Paper

Richard Hayton

1905.01.01 | **Draft** Technical Report | 10 December 1996

Distribution: ANSA Consortium
Supersedes:
Superseded by:

TABLE OF CONTENTS

1 .1 Introduction 1
1 .2 Background 1
1 .3 The Future Network Computing Environment 1
1 .4 Modular Advantages 2
1 .5 Automated Management 2
1 .6 Replaceable Modules 3
1 .7 Types of Modules 3
1 .8 Middleware Modules 3
1 .9 Operating System Modules 4
1 .10 Network Modules 4
1 .11 Focus 5
1 .12 Dynamic Adaptation 5
1 .13 Profiling 5
1 .14 Summary 5

1 .1 Introduction

Flexinet is a framework for open component based network computing. It will allow software configuration and update 'on the fly' by downloading application, protocol and networking modules from networked repositories. The framework supports many flavours of heterogeneity, including multiple object semantics, name spaces and network protocols. Flexible quality of service is provided to applications by allowing them to influence the module configuration, to build a network to suit their requirements.

1 .2 Background

Traditionally, applications and systems have been 'shrink wrapped', and distributed by CD or disk by a slow and costly process. Application designers put effort is put into creating stable and bug free releases relatively infrequently. This is because the cost per release is high, and because introduced bugs cannot be fixed until the next release. From the users point of view, installation of a new release is a potentially difficult and expensive operation. Upgrades cannot usually be done to running systems, and the risk of introducing fresh bugs discourages users from upgrading until the last possible moment.

Java and related technologies have allowed a shift of focus of application design. Applications may now be downloaded electronically, allowing designers to produce many, smaller, releases. The trend has been towards *component based computing*, whereby applications are split into small, reusable parts, which may evolve independently.

Flexinet acknowledges this trend towards component based computing, and attempts to apply the same methods to the management of a distributed computing environment. This has many advantages. Distributed object platforms and networking infrastructures are particularly large and complex, and by their very nature attempt to provide a panacea for use by distributed applications. This static view of platforms and networks is breaking down under the strain of multiple conflicting requirements (e.g., QoS, security, co-operative apps). A modular approach in the design of such a system will allow the independent evolution and coexistence of modules to cope with different classes of application requirement, and thus provide a more open and robust framework for distributed computing.

1 .3 The Future Network Computing Environment

We see the future environment of consisting of large numbers of Network Computers, Personal Computers and consumer devices (phones, PDAs, set top boxes). There will be widespread but heterogeneous connectivity between these devices. Many connections will be static, or essentially static, but there will also be an increasing degree of application, device and user mobility.

This changing environment will lead to the need for the network to become more service oriented. The network must include services to support issues such as mobility, network and application administration in addition to services to provide additional layers of abstraction. For example the network infrastructure should allow network computers to act as terminals, and be able to hide the heterogeneity of different physical interconnects. Applications themselves will tend to be more distributed, and will demand greater network support. For example co-operative applications are likely to become more important, and these have complex interconnection requirements that are ill suited to today's end-to-end reservation schemes. In general terms, the network will be seen less as a simple interconnect, and more as an operating system in its own right.

Operating system development has shown that a 'one size fits all' policy is inappropriate, and this is equally true for networks. In particular networks have complex administrative requirements, and must be able to evolve over time as needs change. This leads us to the conclusion that we need a 'plug and play' facility in all levels of network design; this is true for network elements, where new protocols and technologies lead to

changing models of behaviour, right the way up to high level views on management, connection establishment and charging.

The media that networks transport is also changing. There is an increased proportion of data and video, but more importantly there is a convergence in the way in which these media are used and should be handled. As applications wish to control resource usage, quality of service issues need to be applied to all media types. These issues include, but are not limited to, performance, reliability, security and pricing. New classes of application are emerging, in particular conferencing, surveillance and telepresence-presence applications. These are distributed, and have complex interconnection models. Current signalling models cannot exploit the shared data paths inherent in these applications and new application specific signalling models are required to deliver the full performance of the network. This again implies that a plug-and-play approach is required at all levels of network signalling.

1 .4 Modular Advantages

There are many advantages in a modular approach over and above the 'good software practice' of well define interfaces and code reuse. In Flexinet there will typically be a choice of modules available to provide a particular service, allowing applications to effectively define their own object model and choose QoS trade-offs. For example different modules may support the same message passing abstraction over different low level protocols, or with different failure semantics. The system can evolve by the provision of new modules to provide additional functionality. Existing applications will only use new modules that provide relevant services. For example a working application may not require failure tolerant message passing. If a new 'improved' networking module was produced that provided this facility, it would simply be ignored by that application. This is in contrast to traditional DOPs where all applications share common abstractions, and where existing applications may fail, or suffer performance degradation when the DOP is upgraded. The Flexinet project will have to address the degree of freedom and application has in expressing its requirements and expectations about system behaviour, so that the correct modules can be chosen at run time.

1 .5 Automated Management

The current 'official' view of system management, is that a small number of system administrators are responsible for maintaining the applications and services used by are relatively large number of users. However, it is more often the case that the systems being maintained evolve too rapidly for the administrators to keep pace. This leads to either a profusion of administrators -- who need a high degree of mutual understanding, or to the users being allowed to administer their own, individual, systems. This leads to inconsistency and the undermining of company policies or administrators authority. This situation, is, unfortunately getting worse. The trend is towards open, ubiquitous computing. In this environment, users can download applications from all over the world this increases the degree of user management required, and the probability of problems.

The approach in Flexinet is to assume automated management. Applications specify their requirements in generic ways, and management modules are used to configure other modules to meet these requirements. Where the system is unable to perform automated configuration and human intervention *is* required, then the decisions made may be embodied in a new management module. This may be used to arbitrate if similar circumstances arise in future. It is envisaged that new management modules will be created with each new protocol or system module and the goal is to allow applications to influence management decisions, but to make the process transparent to the user.

There are issues to be addressed here about module ownership and charging. For example a service provider may advertise a new module, that claims to do some task and charge applications or users wishing to use it. There are issues of trust and pricing benefit to be considered by an application that wishes to make use of the advertised module. This is particularly hard in an environment where the management is to be completely automated. The wishes of the application, user and organisation paying for the service must all be considered when making these decisions.

1 .6 Replaceable Modules

Flexinet is designed to be fully replaceable. That it to say that, to as great an extent as possible, Flexinet itself will be implemented as a set of replaceable modules, allowing it to be upgraded over the network in a modular way.

A FlexiNet client consists of a Java virtual machine, and a small core API allowing access to operating system resources and the Flexinet microcore. This is a Java Applet capable of bootstrapping Flexinet. All aspects of Flexinet will be implemented in modules, including object naming and location, security and the mechanisms for locating and installing modules. This will allow Flexinet to evolve ‘under the feet’ of a client application, without the necessity for that client's knowledge or co-operation. Of course, the evolution is itself managed by a replaceable module, so clients can dictate the degree of control they wish to exhibit.

1 .7 Types of Modules

It is useful to draw distinctions between the different kinds of modules that might be found in a system. Firstly, applications are becoming more modular, this is being driven by industry, and the details are not a primary concern of Flexinet. Applications exist in an environment provide by middleware modules. These are responsible for the location and binding of objects, transactions, persistence and other operating system independent issues.

The middleware itself will run over an operating system that may itself be modular. In particular the OS may have configurable scheduling or other QoS controls. Flexinet also considers the network that connects a set of machines in a distributed computing environment. Traditionally, this is seen as a ‘black box’ capable of supporting particular kinds of message transfer between the hosts. Flexinet has a more open view, and considers the network as a set of signalling modules providing different kinds of service on a per-application basis.

Although it is useful to classify modules to aid reasoning, this is a very *soft* interface. It is envisaged that application developers will also provide middleware or network modules in order to provide the API their particular application requires.

1 .8 Middleware Modules

Middleware modules comprise what is traditionally considered the Distributed Object Platform. The primary purpose of the DOP is to locate objects and bind clients to them. There are many decisions to be made as to the semantics and implementation of such a scheme, and Flexinet intends to expose these decisions to the application. The modular approach will allow *plug in transparencies* such as transactions or replication. When a client wishes to bind to an object, the process cannot be statically defined, but must be able to evolve independently of the objects themselves. It should be possible for existing applications to make use of new facilities provided by new middleware or network modules, without themselves being rewritten. This implies a model of late binding, and a trading mechanism whereby requests can be expressed in an abstract way. For example an application may wish to bind to ‘a local colour printer’ rather than name the printer explicitly. The decision as to how late to bind is a difficult one and will have consequences on the performance and flexibility of FlexiNet applications.

Middleware modules will also be responsible for the support of existing Legacy systems, and interworking with alternative approaches to distributed object computing. As the notion of heterogeneity is fundamental to Flexinet, this is a straightforward architectural goal, but implementation may be more complex. For example it must be possible to name and reference CORBA objects from a Flexinet environment. An important issue is whether these objects should be encapsulated so that they appear identical to other Flexinet objects, or whether applications should be made aware of the different origins and semantics of different objects. Architecturally, at least, it should be possible to give the application the choice.

1 .9 Operating System Modules

A continuing goal of operating system research, has been to move more functionality out of the fixed ‘kernel’ and into operating system modules that surround it. There are many reasons for this, including increasing the performance, reliability and security of the trusted kernel, as well as providing a more flexible application interface. Flexinet must address what facilities should be provided by minimal operating system. In addition we must determine what information should be made available by other modules in order to aid the operating systems in resource allocation. If it is to be a goal that FlexiNet is to be both operating system independent and provide applications with QoS guarantees about resource allocation, then a portable operating abstraction must be devised.

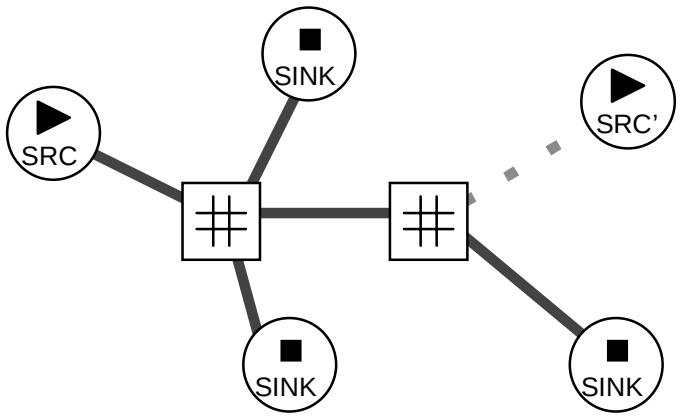
1 .10 Network Modules

New classes of application are changing the view of networks. Collaborative work, publish/subscribe and multimedia applications require application specific, typically multicast, connection patterns. From a resource conservation point of view, we need to split the connections as far downstream as possible. QoS policy may also dictate that resources are reserved throughout the network to minimise latency and jitter. For example, an application may be aware that it is about to create a new connection, and may wish to reserve resources in advance.

These requirements imply a ‘connections and switching’ model for the network. The application cannot view the network as ether but needs to expose the interconnections, so that it can exploit shared paths. Current signalling technologies are heavyweight and are oriented towards end-to-end connections. They explicitly hide the underlying interconnections that must be exposed for QoS and resource management. In addition these signalling technologies are engineered into the physical switching elements. This makes them difficult to change and inextensible.

The DCAN project has proposed a new model of network behaviour, and FlexiNet intends to embrace and extend this. Switches may be considered as simple network elements which buffer and transmit information from input to output ports. The intelligence in the network is responsible for the connection of these ports and the allocation of the buffers. This intelligence may be located in switching modules which need not reside in the same physical location as the switches themselves, and may be configured for particular application requirements.

In the example application below, **SRC** is multicasting data to a number of sinks. The application wishes to ensure that the shared portion of the multicast path is used to reduce the bandwidth requirements. Periodically, the application switches to **SRC’** when it does this, it wishes to retain the same data sinks, and wishes to reuse the existing point to point connections. This is the sort of requirement that is common in conferencing or surveillance applications, but that is difficult to provide using current signalling technology.



The issues to be addressed here is the degree of control that the different parties have in configuring the network. As the network is shared, and both configuration and cell transmission consume resources there must be a management policy to arbitrate between competing applications wishing to use the network. Flexinet applications will also wish to make use of existing, legacy networks, and the protocol modules and management systems must be designed to deal with this interoperability problem.

1 .11 Focus

The Flexinet project will focus on dynamic object location and binding. Flexinet and the systems built with it will be modular, and much of the effort will be focused towards the design and implementation of network repositories to store and manage these modules. The repositories must be federated, and will be responsible for the life cycle of modules stored in them. This will involve publicising new modules, aiding with their configuration, monitoring their behaviour and managing version changes. The issues of object location and binding are primarily concerned with the connection model and the expression and implementation of quality of service requirements. This must take place in an environment where the components available to service requests are constantly evolving themselves.

The approach to these problems is based on the paradigm of distributed objects. We wish to explore the abstraction of a *virtual active database* to express the state of distributed applications and network components. This will allow applications to query the state of the world, to aid QoS related decisions. It will also provide information to management applications, to help with a variety of issues such as configuration, adaptation and dynamic profiling.

1 .12 Dynamic Adaptation

There is a need for both short and long term adaptation of downloaded software. For short term adaptation, the module needs to be able to interrogate its environment and self-adapt. For example a mail posting module may detect that it is disconnected and change to a batch mode. In the long term there is a need for modules to be adapted based on experience of their use. Modules may therefore collect statistics about their use, and send this information to management applications that will check for unusual or unexpected conditions.

This information may be used to dynamically configure the system, for example through load balancing or changing a hierarchical caching strategy.

1 .13 Profiling

Profiling is an important software development process used to tune the performance of a system. In a highly modular system it is difficult to perform accurate profiling, as the programmer may be unaware how the modules will ultimately be used in combination. Dynamic self-monitoring allows the system to collect information about how modules are actually used in a real system and thus effectively allows profiling *in place*. This is an important software development tool in an open environment.

1 .14 Summary

Flexinet proposes a highly modular and configurable solution to open distributed processing. Its key advantages are that is supports multiple abstractions and is designed to evolve over time. The main challenges of this approach are the problems associated with automatic configuration. We believe that these problems must be addresses by future ODP platforms, and that the Flexinet model of replaceable management modules will ensure that future solutions to these issues can be integrated into running systems.