MSNL Connection Management

Simon Crosby

16th February 1993

1 Introduction

This document describes a management and control architecture which aims to permit flexible, platform independent communication management in MSNA. The management and control (M&C) functions are those required to set up and maintain a data path (a liaison) between two MSNL Service Access Points (SAPs) in an MSNL internetwork, and include the following tasks:

1. Signalling between intranetwork entities.
2. Signalling between network entities and internetwork routers.
3. Internetwork signalling between internetwork routers.
4. Topology determination and routeing of internetwork requests.
5. Allocation and management of MSNL SAPs.
6. Resource management and monitoring to ensure that QOS constraints are met.
7. Network maintenance functions.

Other functions may also be required for specific communication needs. The most basic requirements of a network entity which wishes to use MSNL include only:

- Signalling to intra- and internetwork entities requesting connection set up and teardown.
- Terminating the MS-Access, MSDL, MSNL, and MSSAR layers to permit data transfer.

Current implementations of MSNL combine the functions for data transfer and network M&C into a single functional unit, normally bound within the kernel of the host operating system in an operating system specific manner. Because it is difficult or impossible to gain access to the M&C functions from outside the kernel this has led to the *de facto* "definition" of an MSNL entity as the communication part of an operating system kernel. The consequences of this definition are that:

1. Every MSNL entity must incorporate a kernel (or sufficient intelligence) to permit it to carry out MSNL management and control functions.
2. An MSNL entity, and therefore its management plane, is bounded by an operating system kernel.

Consequence 1 means that dumb devices such as the ATM camera, which cannot support the M&C functions cannot use MSNL for data transfer. Consequence 2 makes the management of multiprocessor entities such as switches very difficult, because each processor (kernel) is responsible for its own management. It is impossible to manage shared resources, such as switch bandwidth, without managing the functionality of a switch as a single object with multiple interfaces. This proposal outlines an approach to MSNL communication management which will permit us to aviod these unneccessary restrictions.

The remainder of this document is structured as follows: Section 2 argues for a distinction in functionality between the different planes of the MSNL protocol stack, and identifies the functions of each plane. With reference to this model, Section 3 identifies the elements of the proposed management and control architecture, and Sections 4 and 5 describe the proposed structure of the control and managment planes, discussing metasignalling, signalling, connection management, routeing and other functions of management and control.

A Unix implementation of the architecture described in this document already exists. It is fully described elsewhere in this collection. Instructions for the use of the manager, and the environment in which it runs are also documented in this collection.

2 The MSNA Protocol Stack

It is important to distinguish between the different functions required for data transfer, network control and network management. The reference model for the BISDN protocol stack, pictured in Figure 1, identifies three protocol planes, the user, control and management planes. It is possible and indeed desirable to identify similar layer and plane boundaries in the MSNA protocol stack, and for the sake of simplicity we will adopt the BISDN plane and layer terminology in the discussion which follows. The layers in the MSNA protocol corresponding to the BISDN stack are are indicated to the left of the diagram.

The user plane contains the functions required to transfer data between two network entities. This involves terminating the MS-Access, MSDL, MSNL and MSSAR layers at every communication endpoint. The control plane is responsible for the following functions:

- Establishment, maintenance and teardown of connections.
- Support for point-to-point, multicast and broadcast communication.
- Negotiation of traffic parameters at the time of connection set up.
- Addition / deletion of connections to / from a muilticast call group.
- Resource control during a call. eg. congestion control and policing.
- Resource control at connection set up. eg. Connection Admission Control (CAC).
- Determining the network topology and the cost associated with each path through the network.
- Routeing of connection requests through the network and between networks.

Management plane functions include fault detection and reporting, configuration and topology management, financial management (charging & maintenance), performance monitoring and reporting, access and security management and design management [1]. Thus the management plane is not concerned with the establishment of communication *per se.* It

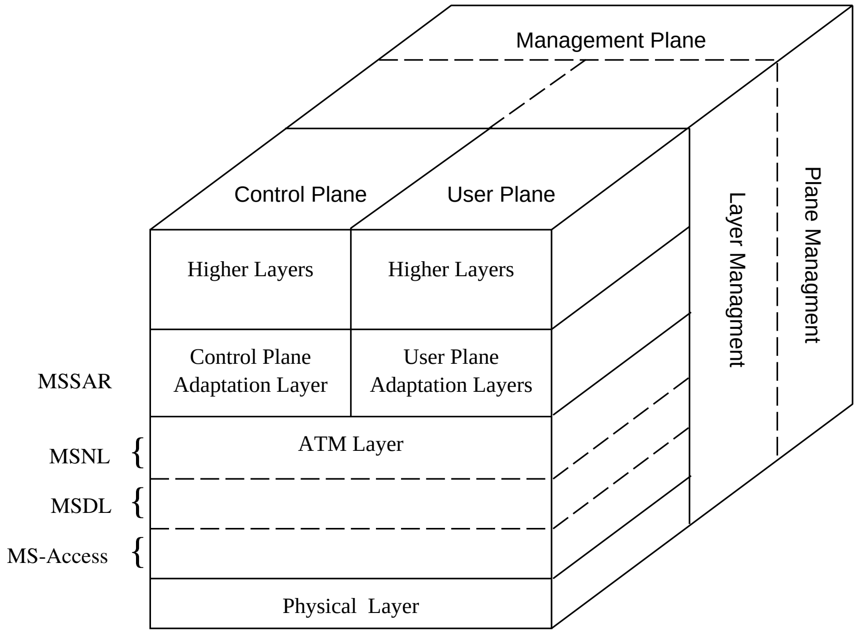

Figure 1: BISDN protocol reference model

provides access to information on the network state and permits the management of each layer of the protocol stack.

3 Management and Control in MSNL

This section identifies the components comprising a hierarchically structured MSNL internetwork, and associates with each layer of the hierarchy a set of functions which implement the management and control plane for MSNA at that layer.

3.1 Network Entities and Management Domains

We define a network entity to be an object which has one or more network interfaces. We wish to depart from the restrictive requirement that each network entity must implement the functions of the management and control planes as well as those of the user plane. We therefore define an MSNL Management Domain (MD) as a collection of one or more network entities grouped together for the purposes of management and control. Each MD implements those functions of the management and control planes which are required to ensure that the network entities in the MD can achieve their communication needs both within the domain and between management domains. Although the grouping of entities into a domain may be arbitrary, in practice a MD will probably reflect a logical group of entities, such as a switch. Within a MD the boundaries between entities are preserved – this is required because each entity is still capable of independent communication. At the MD level each entity is abstracted as a set of interfaces which provide the management and control functions access to the MSNL, MSDL and MS-Access instances in the entity. An entity exports via these interfaces information regarding its physical connectivity and capabilities, and also its status.

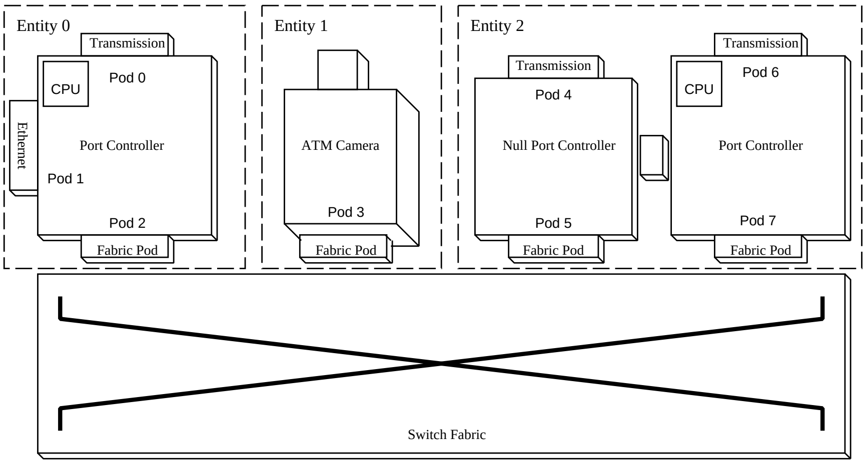

Figure 2: An Example MSNL Management Domain

Each network interface (also referred to as a pod) attached to an entity is assigned an interface address which is unique in the MSNL internetwork. An entity is responsible for implementing the user plane of the MSNL protocol stack for each of its pods. Thus an entity should be able to terminate the MS-Access, MSDL, MSNL and MSSAR protocol layers in the user plane for each pod, according to its needs. In addition an entity may provide support for the external implementation of the control and management plane functions for each layer of the protocol stack. For instance at the MS-Access layer an entity may export an interface which permits the management function to detect the error rate for the pod, or measure its performance. The control and management functions for each layer of the protocol are implemented on behalf of the entity by a set of co-operating management and control services which are instantiated per MD and which interact via well defined interfaces. The definition of the interfaces exported by each entity to permit the external implementation of the management and control planes is dealt with in the sections which follow. Figure 2 illustrates a MD composed of 3 entities. Entity 0 is a Fairisle Port Controller (FPC) with an Ethernet interface. Entity 1 is an ATM camera, and entity 2 is a Fairisle Port Controller and an attached Null Port Controller. This entity has 4 interfaces.

Some entities in a MD may be incapable of exporting the management and control interfaces mentioned above. An example is the ATM camera, which exports a control interface which is entity specific. Since the management and control services expect each entity to export the appropriate interfaces, however, they must be implemented for each entity within the MD, possibly by a proxy object which translates calls from the management and control services into the entity specific control protocol to achieve the management and control functions.

A MD is regarded as a single functional unit insofar as management and control are concerned. The configuration of an MD is fixed, and known when the MD is initialised. When an entity in an MD fails (crashes), then the MD is considered to have failed, and must be restarted. This restriction is motivated mostly for reasons of network consistency, but it may be possible in practice to relax it so that an MD is considered to have failed when any of its essential control and managment services fails.

3.2 MSNL Networks, Subnetworks and Internetworks

To permit the definition of the terms used to describe MSNL networks, we define the *type* of an MSDL instance to be the type of the physical layer on which it is implemented, for example CFR, CBN or YES/MAYBE. Different physical interfaces may exist for an instance of the same type, for example ethernet interfaces could be of kinds DEQNA or LANCE, but they interface to an instance of type ETHERNET. Note that these definitions are at odds with the current Wanda definitions.

The physical layer type frequently prescribes the *class* of MSDL which is implemented upon it. For instance the common cell size on the CBN and CFR can most efficiently support the MSDL class MDL, whilst the YES/MAYBE type efficiently supports the class FDL. Note, however that a network of type ETHERNET could easily support several MSDL classes. Within a class, different subclasses of an MSDL instance are permitted, however we do not distinguish between them here.

For the purposes of this document we define an MSNL subnetwork as follows: A single instance of MSDL, of any type, represents a (primitive) subnetwork. A set of MSDL instances (primitive subnetworks) of the same type which are directly interconnected also comprise a subnetwork. MSDL instances are interconnected by the MDs which are attached to them. According to this definition a set of Fairisle switches linked together by point-to-point links constitute an MSNL subnetwork; similarly a CFR is an MSNL subnetwork.

Each network interface within an MSNL subnetwork is given a unique MSNL interface address. Associated with each MSNL interface address is a mask which can be applied to the address to identify the subnetwork to which the interface is attached. A MD is assigned a primary address which is the address of one of its interfaces, and its other interface addresses are known as aliases.

The point of interconnection of two MSDL instances of different types will, in general, delimit two subnetworks. Two or more interconnected subnetworks form an internetwork. The terms network and subnetwork can be used interchangeably. Administrative or logical boundaries may be imposed on the network structure. Administrative boundaries are authoritative in the sense that they can impose subnetwork boundaries which are in contrast with the rules above. As an example two CBNs with a single host MD which is a router between them could be regarded as a single subnetwork, however it may be more convenient to view them as being separate subnetworks based on their physical connectivity or for other administrative reasons.

Entity 2 in Figure 2 has interfaces onto two MSNL subnetworks. Where an administrative boundary exists between two subnetworks, the management and control functions for the network entity bridging the two subnetworks may be split between two MDs. In this case each MD will implement management and control functions for the interface(s) and associated protocol layers which lie within its administrative boundary.

4 The Control Plane

The control plane functions are implemented by a set of co-operating services within each MD. Layer boundaries are preserved within each service, and access to the control plane is gained via control interfaces exported by each service. We describe each function in terms of the interface which it exports.

4.1 The User Network Interface

The MSNL protocol suite is supported by many operating platforms, each of which provides users with a different abstraction of (and interface to) the comunication facilities offered by MSNL. Any such system is by definition an entity in a MD. We define the the User Network Interface (UNI) to be the interface between a user wishing to use MSNL, and the MSNL control plane. Each operating platform must translate calls across its specific communication interface (the Platform Specific Interface, PSI) into calls across the UNI, on behalf of each of its MSNL users. The UNI defines the signalling protocol with which the user controls all aspects of the communication. This includes the allocation of MSNL SAPs and the set up and tear down of MSNL connections. On the user side of the UNI a state machine represents the user's view of the current state of a connection. On the network side (within the control plane) a similar state machine is used to control the connection. These control functions are implemented by a service known as the MSNL Master, and it is to this service that UNI signals are directed.

The MSNL Master maintains the state for every call within the MD. The UNI signalling protocol specifies the valid state transitions which can occur in response to user requests and network events. The UNI presents the user with an MSNL SAP, which is an 8 octet address which uniquely identifies the user service access point in the MSNL internetwork. A pair of SAPs uniquely identifies a connection, and is called a liaison. The user platform is responsible for some adiministrative tasks, such as ensuring that independent users (eg. programs) do not bind to the same MSNL SAP.

Consider the steps involved in connection set up between a user and a remote MSNL SAP. First the user must request a SAP for its end of the connection. It does so by signalling a request across the PSI. The platform supporting the user then signals the request across the UNI. The MSNL Master responds by returning a SAP bound to a call reference unique to the user and entity, which identifies the user access point to MSNL in the PSI. In Unix this call reference could be a socket identifier. Having been allocated a SAP the user can signal a request across the UNI (via the PSI) for a connection to the remote SAP. In response the MSNL Master takes steps to establish a communication path between the two SAPs. This may involve signalling between MDs distributed over the MSNL internetwork. It will also involve actions in the control plane to build each layer of the data path for the connection within the entity supporting the user. Each entity which supports UNI signalling, therefore, must export a control interface which the MSNL Master can use to direct the construction and destruction of MSNL data paths within the entity. This interface is called the MSNL Slave interface. When the path has been established the MSNL Master informs the user via the UNI that its request has been carried out and that the data path is available.

Note that the definition of an MD does not require the MSNL Master or any of its associated services to reside physically on an entity or entities within the MD. It would be perfectly feasible to locate many MSNL Masters for different MDs on different subnetworks on a single host, possibly within a single process, if required.

4.2 Metasignalling

Before an entity can signal across the UNI to request communication facilities, it requires an MSNL connection (a signalling connection) on which to transmit the UNI signals. In order to obtain a signalling connection it transmits a request on a metasignalling circuit using the metasignalling protocol. The metasignalling circuit is a permanent virtual circuit (PVC) established by an entity for each of its network interfaces when it boots. The virtual circuit identifier for the metasignalling connection is defined per MSDL instance, and has a well known value. The metasignalling protocol is a primitive protocol for which each message is contained within one MSDL PDU. The format of the metasignalling messages differs per MSDL class.

The metasignalling protocol permits an entity to request a connection to a control plane function (the MSNL Master) to which it can direct UNI signals. For dumb entities which are incapable of signalling using the UNI and which do not export an MSNL Slave interface, the metasignalling protocol can be used as a full blown signalling protocol. In this case all signalling from and to the entity is achieved on the metasignalling virtual circuit. Each such entity is regarded as an independent MD. The metasignalling protocol is specified in [2], and is currently used in MSNL as an out of band signalling protocol. Incoming metasignalling requests are associated with the pod on which they were generated. Metasignalling requests are forwarded to a service known as the Metasignalling Service, which is implemented per MD, possibly in a distributed fashion. Metasignalling requests are translated from their MSDL dependent form into a standard format and forwarded to the MSNL Master.

4.3 Signalling and the MSNL Master

Having obtained a signalling channel using the metasignalling protocol, an entity can signal its requests across the UNI. The signalling channel connects an entity to the MSNL connection control function for the MD, which is located in the MSNL Master. The MSNL Master is responsible for co-ordinating the management of communication resources within the MD. When a user signals a connection request across the UNI the MSNL Master must check whether sufficient resources are available within the entity, the MD and the destination entity to support the request. If not then the connection request will be refused. When a connection can be established, the MSNL Master controls the construction of the data paths within the entities involved in the connection, using the MSNL Slave interface. The UNI signalling protocol also permits resource management at the user level. The MSNL Master ultimately defers to the user in deciding whether or not to accept a connection.

The MSNL Master is also responsible for inter-MD signalling. When a user requests a connection to an MSNL SAP which is located on a remote MD, the MSNL Master of the user's MD must take steps to locate the remote SAP (routeing) and must engage in inter-MD signalling with the MSNL Master of the remote MD. If the remote MD is attached to a different network, or even several networks distant from the user's MD, then inter-MD signalling must take place for every "hop" which the connection will traverse. To accomplish inter-MD signalling, an MSNL Master uses a an inter-MD signalling circuit which connects it to the MD to which it wishes to signal. This circuit is obtained using the metasignalling protocol. The protocol used for inter-MD signalling is the Network–Network protocol, and it is specified in the Network-Network Interface (NNI). The definition of the NNI is independent of the definition of the UNI, however the NNI is expected to support the basic primitives required by the UNI. Dissociation of the NNI from the UNI permits each protocol to be independently implemented and modified.

The MSNL Master uses information from a number of control and managment functions, including:

- The resource allocation control function which manages resources in each entity and within the MD as a whole.
- The resource monitoring control function which ensures that the requested QOS pa-

rameters for each call are maintained.

- The topology determination managment function which determines the topology of the networks to which the MD is attached.
- The routeing function which determines which route to use for the data path connecting two MSNL SAPs.

Each of these functions is implemented as a separate service, exporting a well defined interface. Since many of these management and control services will be common to all MDs, we do not require an instance of each service for each MD, merely a mechanism by which elements in each MD can access the required services. As an example, there might be one implementation of the routeing service for a whole MSNL subnetwork. Each MSNL Master in the subnetwork can query the single instance of the routeing service to obtain routeing information.

The MSNL Master is responsible for the creation of an MSNL liaison between two SAPs. It must therefore instruct each entity to build the elements of the data path linking the network to the user SAPs. There are three types of liaisons: local, remote and forwarding liaisons. Local liaisons are those between two users on the same entity (platform). In this case the platform will in general provide an efficient and optimised mechanism for user–user communication. Remote liaisons are those which link a user (and its entity) to a different entity. The remote entity may be within or external to the MD. If it is within the MD then the MSNL Master for the MD is responsible for the setup of the entire data path and completion of the liaison. It is free to optimise this process. If the remote user is located on an entity external to the MD, then inter-MD signalling will be used to set up the elements of the data path which traverse other MDs and the destination MD. When A MD is traversed by a liaison which is not terminated in one of its entities then (for that MD) the liaison is a forwarding liaison. Such liaisons are implemented by simply building a data path which ensures that data transmitted over the liaison will be forwarded transparently through the entities in the MD. Such liaisons can frequently be very highly optimised.

To build the data path for a liaison of any type the MSNL Master exercises operations in the control plane of each layer of the MSNL protocol stack for the entity concerned. It does this via the MSNL Slave interface, which presents the MSNL Master with a mechanism for exerting control in the MSNL, MSDL and MS-Access layers within the entity. Every entity is required to support the MSNL Slave interface. Typical control actions include the creation and destruction of associations, assignment of associations to instances of MS-Access, and creation and destruction of liaison records. Figure 3 depicts a single entity and the control and data paths involved in constructing a data path. User and MSNL Master signal via the UNI, whilst the entity exports the MSNL Slave control interface to permit the MSNL Master to construct the data path within the entity. Examples of operations which are invoked across each interface are given. The MSNL Master makes use of other services, such as a router, to route requests. The diagram does not show all of the services and interfaces required to perform MSNL control and management, but illustrates the general principles.

4.4 Inter–MD Signalling

When a MD wishes to set up a connection to an MSNL SAP in a remote MD on behalf of a user supported by one of its entities, it needs to signal this to the remote MD. Each MD therefore supports the Network–Network signalling interface (NNI) which supports the required operations. The NNI is similar to, but far simpler than the UNI. The NNI is regarded

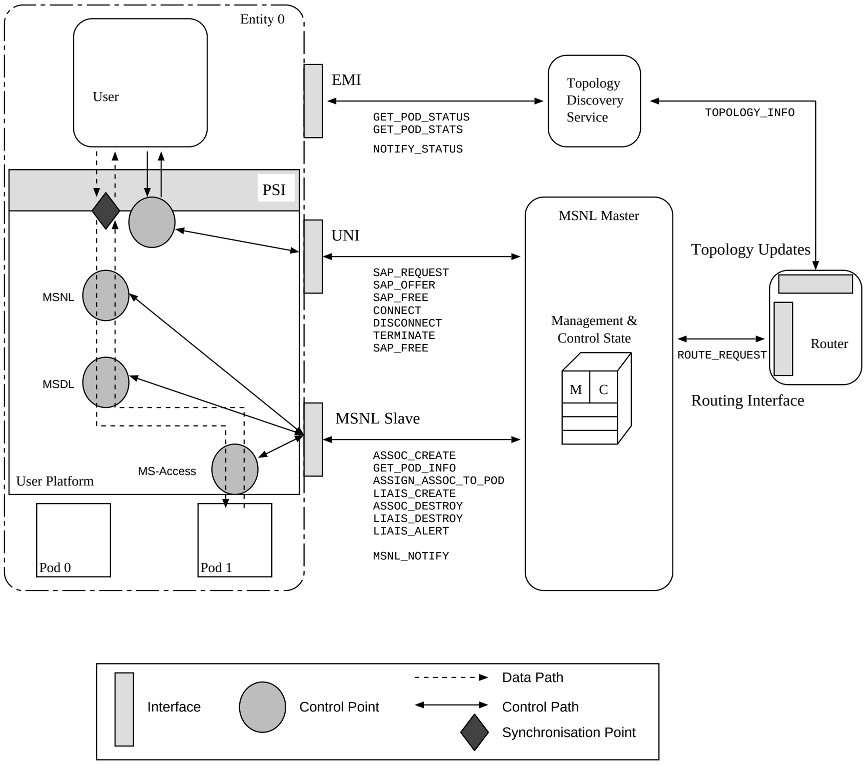

Figure 3: MSNL Master and Slave

as the MSNL network signalling protocol, and it replaces the current mechanism of signalling on the metasignalling channel between MDs.

4.5 Routeing

In order to route requests between MDs, a routeing function will be implemented. Each MSNL Master will have an interface to a routeing function which is capable of determining the MD within whose control an MSNL SAP lies. In addition the MSNL SAP will be used to determine the route for the data path linking two SAPs within a MD. The interface to the routeing function will be defined in such a way that arbitrarily sophisticated routeing algorithms may be implemented without requiring the interface to change.

5 The Management Plane

To support the management plane of the MSNL protocol stack each entity must provide information to the management functions about its state, and possibly permit them to modify its state. The management plane is provided this access via a management interface, the entity management interface (EMI) which each entity must support. Entities which are incapable of supporting the EMI will have the EMI implemented for them by proxy objects which are capable of determining the state of the entity and forwarding this information to the management functions.

5.1 Topology Discovery

The routeing function and the control functions within a MD need to know about the topology of the MSNL internetwork to which the MD is attached. Determining the topology of the internetwork is the task of the topology discovery service (TDS). In order to perform its task the TDS requires management information from each entity in the MD to permit it to determine the connectivity of the MD. It is assumed that each pod will implement some error detection capabilities, and will gather information on its state. This information will be forwarded to the TDS across the EMI.

5.2 Other Services

Additional services may be required in order to fully implement the managment plane. These might include [1]:

- Configuration management: specifying the management of physical resources and how they are included into the configured network.
- Financial management: dealing with charging and maintenance costs.
- Resource management: this is the function managing the physical construction of the network.
- Performance management: this function must ensure that the network is tuned to meet its performance requirements in terms of user perceived QOS.
- Planning and design management: this function should aid the planning of future network facilities and enhancements.
- Access and security managment: these functions control access to the network facilities and network managment entities.

These functions build upon the basic managment and control structures specified above. If required they could be implemented at a later date.

6 Other Issues

In order to implement the design described above, several issues need to be addressed, including the following:

- Error control and propagation.
- The performance of the control system.
- Bootstrapping of MDs and networks.
- Synchronisation during call set up and tear down.

This document has tried to address only the high level design issues for the MSNL management and control structure proposed here. Detailed design issues are the subject of current work and will be addressed in subsequent documents.

7 Protocol Definitions

The protocol described in this document has been fully specified, and is currently being verified to prove absence of deadlock and livelock. The specific will be provided on request to the author.

Thanks to Richard Black for several suggestions on the improvement of the proposals in this draft.

References

[1] T. J. King, T. Barker, A. Micallef, and K. Mackay. Asynchronous tranfer mode (ATM) – practical management issues. *British Telecom Technology Journal*, 9(2):57–70, Apr. 1991.

[2] U. of Cambridge Systems Research Group. ATM document collection. Technical note, University of Cambridge Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, UK, Mar. 1994.