A Signalling API for the ATM Warren or 'Look, no Microprocessors!'

Richard J. Bradbury* and David J. Greaves†
University of Cambridge Computer Laboratory
http://www.cl.cam.ac.uk/Research/SRG/HAN/Warren/

OPENSIG Spring '97 Workshop
17–18 April 1997
Møller Centre, Churchill College, Cambridge

Abstract

The ATM Warren is a low cost subnetwork of a general ATM network. Switches and end devices in the Warren have no control microprocessors, not do they possess unique identifiers. Instead, they implement all control and management functions entirely in hardware. A single computer, known as the Warren Controller, acts as a proxy for all of the devices within a Warren and provides the interfaces required to offer standardised, open control and new network services. As well as offering the normal ATM layer functions, these interfaces have support for handling anonymous devices and secure access control.

The Warren project at the University of Cambridge Computer Laboratory is investigating the viability of inexpensive ATM networks in the home area. ATM is often perceived as a high-cost networking technology, affordable only in the wide area, where high levels of traffic aggregation make high data rates economic. However, by limiting the Warren project to sub-100 Mbit/s speeds, and by reducing the complexity of network devices, we have been able to demonstrate that ATM equipment can be constructed very cheaply.

The multi-service approach of ATM is well suited to the home environment where low-speed, real-time control streams must co-exist with higher bandwidth, delay-sensitive multimedia streams. In the home, the mix of traffic is likely to include HiFi audio and compressed video together with data streams of varying bandwidth and low-rate control channels. As well as supporting these streams on a point-to-point basis around the home, the Warren's switched network design allows streams to be sourced or sinked externally via a wide area network connection.

Because the likely bandwidth requirements within the home can be quantified, it is possible to dimension the network parameters to suit. The 25.6 Mbit/s links with which we are currently experimenting are capable of delivering several MJPEG compressed video streams ($\sim$8 Mbit/s), HiFi-quality audio streams ($\sim$1.4 Mbit/s) and network control streams (very low bandwidth) simultaneously to any network outlet. Each inter-switch link has identical capacity. By running the same line interface components at double the speed, link bandwidths of 51.2 Mbit/s can be achieved at minimal extra cost. By replacing the UTP copper-based medium with Plastic Optical Fibre, line termination is actually cheaper, home installation is greatly simplified and electromagnetic interference is reduced.

Low cost is achieved in the Warren by trading hardware complexity in the switches and end stations for software functionality. Moreover, further economies are achieved by centralising this software in dedicated computational nodes ('Warren Controllers'), leaving very simple and cheap devices with a small and fixed set of capabilities. Warren end stations, for example, transmit and receive only on well-known VCIs; on/off control of these data streams is achieved by manipulation of the routing tables of immediately adjacent switches. The Warren switches themselves have no microprocessor, and control is achieved by means of a simple RPC mechanism ('Warren Control Protocol') which enables the routing tables to be manipulated remotely by central software.

All decisions on routing, call admission and security are taken centrally by the Warren Controller which has management responsibility for a particular Warren subnetwork. Optionally, the Warren Controller can participate in a wider, distributed control federation which enables communication between end devices in different Warrens via a wide area B-ISDN.

* *email:* Richard.Bradbury@cl.cam.ac.uk
† *email:* David.Greaves@cl.cam.ac.uk

Despite the simplifications described, the Warren switches support all in-band ATM semantics, allowing traditional UNI-compliant end devices to be attached to a Warren subnetwork. By nailing up suitable connections to a proxy signalling agent in the Warren Controller, all the standard signalling VCIs can be provided at the user–network interface. The agent interprets requests from UNI devices (and, optionally, signalling entities external to the Warren) and carries out the relevant actions.

The ATM Warren provides a solution to home area networking which combines both control and data into a single framework whilst remaining competitive with other HAN standards. The incremental cost of adding an ATM line interface, which supports a handful of hard-wired virtual circuits, to high volume consumer electronics products is also small enough to be attractive to manufacturers.

Interface summary

The `ConnectionFactory` interface manufactures a `Connection` based on the `DeviceID`s of two end points. The resulting object encapsulates the route between the two points, comprising a designated transit list (for the 'real' segment of the connection) and/or location(s) of end points inside the control architecture (for the 'virtual' segment) as appropriate. In the spirit of object-oriented design, the process of actually setting up and tearing down the connection is achieved by calls to the `Connection` object itself.

```
/* 'Soft MAC' */
struct DeviceID;

interface Connection {
  boolean connect();
  boolean disconnect();
};

interface ConnectionFactory {
  Connection manufacture(in DeviceID src, in DeviceID dest);
};
```

Each Warren switch is represented in the control architecture as a *proxy* object which implements the `Switch-Connections` interface (loosely based on *Ariel*). The `Connection` object makes calls to this interface of each switch along the designated transit list in order to set up an ATM virtual circuit for the 'real' segment of the connection. Additionally, an appropriate `ProtocolHandler` is instantiated dynamically to act as a gateway between the 'real' and 'virtual' segments where necessary.

```
typedef unsigned short Port;

struct PortVCC {
  Port port;
  unsigned short vpi;
  unsigned short vci;
};

/* Loosely based on DCAN Ariel */
interface SwitchConnections {
  boolean make(in PortVCC from, in PortVCC to);
  PortVCC make(in PortVCC from, in Port to);
  boolean break(in PortVCC from);
};
```