Warren

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

Richard J. Bradbury and David J. Greaves

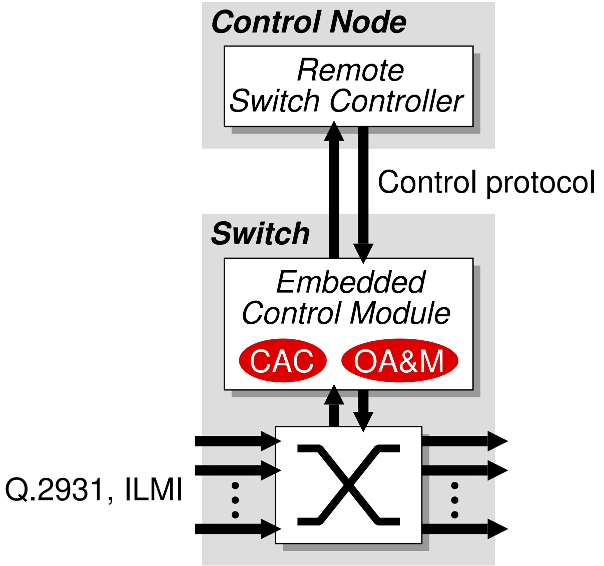
University of Cambridge Computer Laboratory

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

http://www.cl.cam.ac.uk/Research/SRG/HAN/Warren/

- This presentation introduces the Warren, an ATM-based Home Area Network, in its context as an application of Open Signalling.
- Because of the extreme simplicity of the ATM devices we have constructed for use in Warren networks, the project has been a proponent of Open Signalling from day one, not by choice, but by necessity.

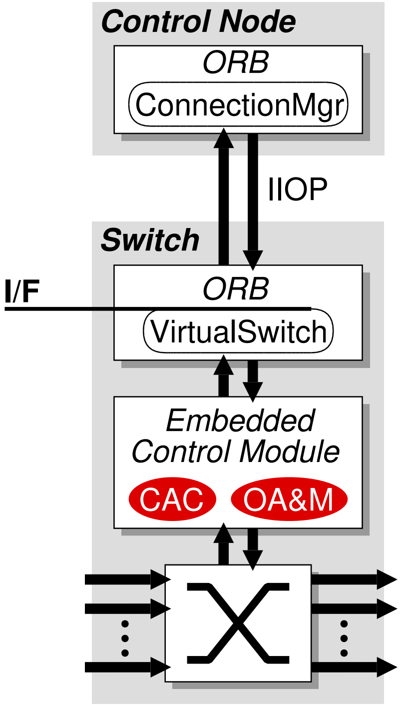
Traditional ATM Switch



- Controlled by traditional in-band signalling such as Q.2931…
- …or a proprietary lightweight signalling protocol such as *SPANS* or *MAST*…
- …or a general control protocol like *GSMP*.

- The rather naïve model of a traditional autonomous ATM switch shows that the embedded control module is responsible for interpreting signalling and OA&M streams. Decisions on CAC are made locally.
- Some vendors support a lightweight signalling protocol on their switches (notable examples are *MAST* and Fore's *SPANS*). These are intended to offer a more efficient form of signalling than the heavyweight standards-based signalling protocols. The obvious disadvantage is their proprietary nature.
- In addition to this, many switch vendors allow their hardware to be managed remotely by an external switch controller (typically a workstation dedicated to this task) using some proprietary control protocol.
- Some of these protocols (particularly *GSMP*) have gained more widespread acceptance, allowing the same control software to be used with switching equipment from a variety of vendors.
- (The figures presented illustrate the control protocol as a distinct interface to the switch. In reality, of course, these control paths may enter and exit the switch via its normal ATM ports, shown in the lower portion of the figure.)

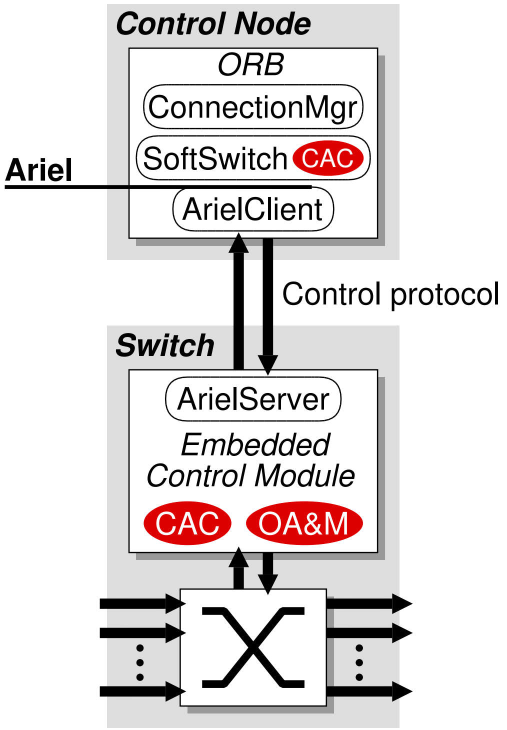
xbind Control Model



- Object Request Broker resident on the switch itself which hosts a *VirtualSwitch* object.
- Control by means of distributed object calls to the *VirtualSwitch*, using an inter-ORB protocol such as *IIOP*.

- One disadvantage of the *xbind* model is the need to support an ORB on the embedded switch controller. This is relatively straightforward for switches with control CPUs which run traditional UNIX-like operating systems. It could be more difficult for the new breed of switch products which employ embedded real-time operating systems.
- Interfacing the ORB to the existing switching software in this way also requires co-operation from the switch vendor. Many vendors are unwilling to reveal details of their software due to the high value they associate with the intellectual property.

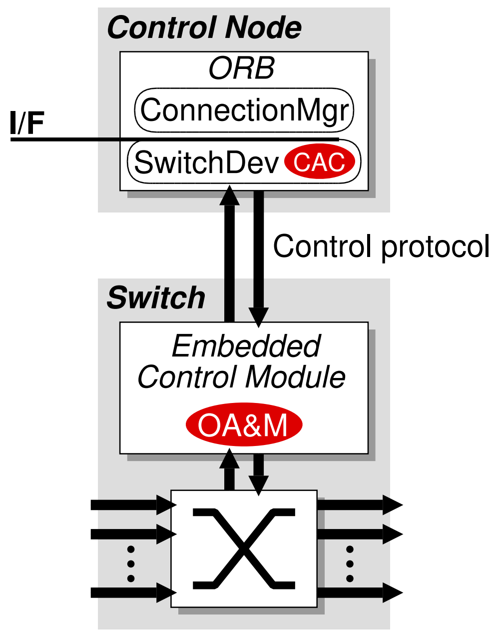
DCAN Control Model



- Controlled by a remote *SoftSwitch* which is a distributed object. The switch itself does not have to participate in the DPE.
- *Ariel* API offers a protocol-independent view of connection establishment.

- The switch is controlled by a remote *SoftSwitch* which is a distributed object in a DPE. The switch itself does not have to participate in the DPE.
- The *Ariel* API offers a protocol-independent view of connection establishment by encapsulating the particular control protocol supported by the switch. An example of an *ArielServer* is an SNMP daemon running on a particular vendor's switch. The corresponding *ArielClient* for such a switch would then dispatch SNMP messages to the hardware in response to calls on its interface.
- *Ariel* provides a common view of the basic 'nuts and bolts' of switching hardware, such as VCI mappings. The server itself is stateless.
- With this scheme, no particular protocol is favoured, and no modifications to the switch are necessary.
- The *SoftSwitch* is the representation of the switch in the control architecture. It manipulates the switch via the *ArielClient*.
- Other functionality, such as CAC decisions and certain aspects of OA&M can also be handled outside the switching hardware by the *SoftSwitch*.

Warren Control Model



- As with *DCAN*, a distributed *SwitchDevice* is in control, and any suitable protocol can be used to talk to the switch hardware.
- Because some of the switches under control are so simple, functionality such as CAC must be provided externally.

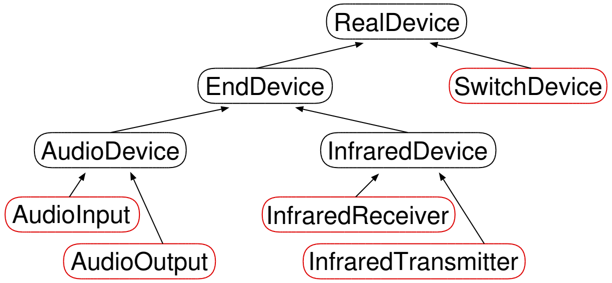
- The only components on the Warren switch are a 6-port physical layer chip, a Xilinx Field Programmable Gate Array and some SRAM. There is no microprocessor.
- The device is very good at forwarding cells, but apart from that is pretty dumb. It can't do CAC and it can't do standard OA&M (although the latter is feasible, we prefer to use a proprietary control protocol with equivalent functionality).
- We can't support GSMP on the switch, and we certainly can't follow the xbind model.
- Our only option, therefore, is the use of an external switch controller. We use the term *proxy* in the sense that it 'stands for' the switch in the Warren control architecture, and implements the control and management functions on its behalf.
- The higher layers of the control architecture, such as the *Connection* objects access the switch through this proxy object. In common with the other approaches, the interface to the switch proxy is derived from a generic base class, in order to support polymorphism between heterogeneous switches.

  The interface we use is suspiciously similar to *Ariel*.
- In the case of the Warren switch, the switch proxy translates connection requests into special 'magic cells' which poke values directly into the SRAM routing table in the switch.

  Clearly, the proxy for a different vendor's switch could use a different protocol, such as GSMP, or perhaps even traditional Q.2931 signalling requests.

Warren Device Proxies

- The idea of a proxy software representation for switches can be extended to end stations.
- The *xbind* project has already shown this concept with its *VirtualDevice*.
- In the present Warren Control Architecture, end station devices are classified according to their media type and, orthogonal to this, whether they are sources or sinks of data.

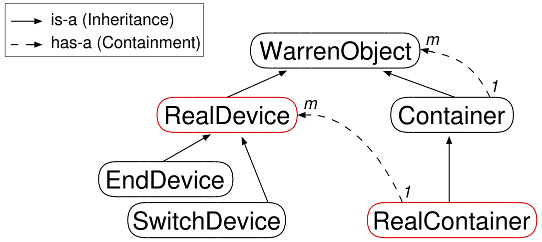


- Individual equipment is modelled by subclassing the appropriate base class.

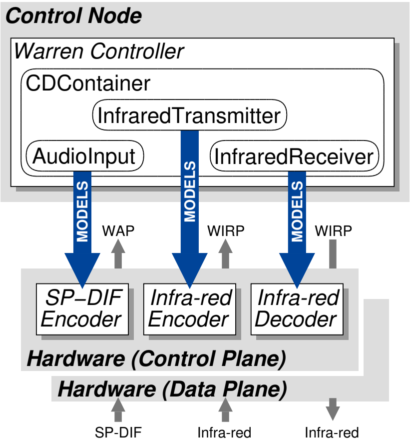
- Both the xbind and DCAN projects have extended the concept of proxy software object representation of switches to end stations as well.
- The AVA-200 (ATM Camera) is an example of a device which is not smart enough to operate autonomously. A proxy software object running on a general-purpose workstation handles all signalling requests on its behalf and tells the hardware, via a control connection, which VCI to transmit audio and video on. Hardware configuration options are also handled by proxy.
- In the Warren, all real devices are modelled by a software object which is derived from the *RealDevice* class. End stations are further classified by media type.

Composing Proxies

- Composite devices are synthesised by collecting several *logical* proxies together in a *Container*:

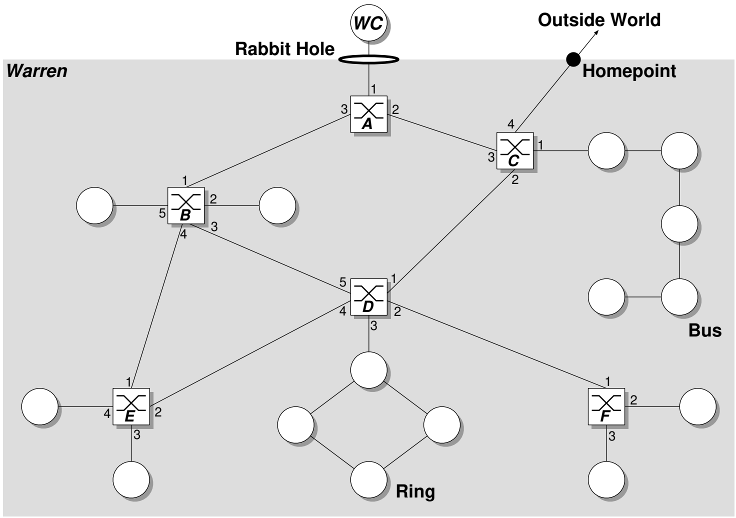


- The Warren CD Player is composed from three such proxies:



- Composite devices – those with multiple distinct functions – are synthesised by collecting several *logical* proxies together in a *Container*.
- For example, the Warren CD Player is composed from three such proxies. *Note that the figure only illustrates the control plane.*

Warren Network Architecture

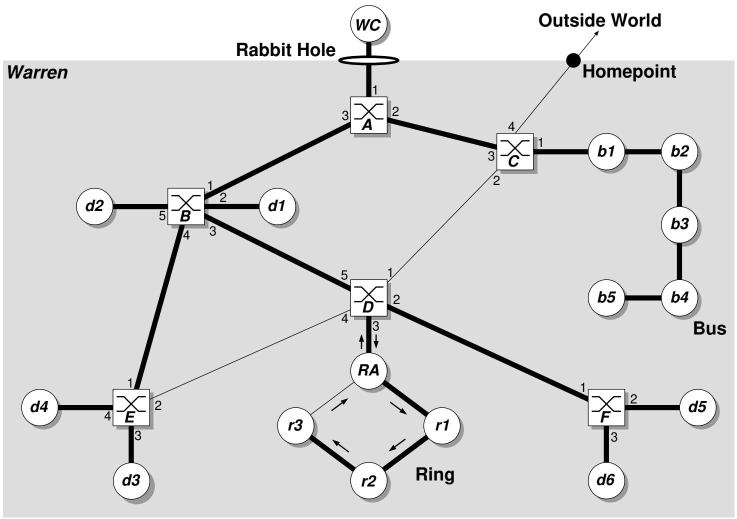


- Network is an arbitrary mesh of very simple and cheap ATM switches and end stations.
- At the top of the Warren is the *Rabbit Hole*, above which lies the *Warren Controller*.
- This node hosts the proxies for the real devices (both switches and end stations) below.

- Network is an arbitrary mesh of very simple and cheap ATM switches and end stations. The cost of the prototype switch was £300.
- At present, all links are 25.6 Mbit/s over UTP. We are currently evaluating a Plastic Optical Fibre alternative running at 52.1 Mbit/s.
- Other network topologies, such as buses and rings are also being considered in order to cut costs still further.
- At the top of the Warren is the *Rabbit Hole*, above which lies the *Warren Controller* computer. This hosts the proxies for the real devices in the Warren below.
- There may also be a connection to an external B-ISDN public network through which the Warren Controller communicates with other Warren Control Nodes in the federation. Clearly, connections with end points on networks external to the Warren can also be established across this link.
- In principle, device proxies can reside in any Warren Control Node, but for reasons of security policy, they are normally hosted by the corresponding Warren Controller.
- The term *Homepoint* has been coined to describe the active network termination forming the boundary between the public and customer premises networks.

  The Homepoint adapts the customer's chosen access network (PSTN, ISDN, ADSL, HFC, wireless, . . .) to a standard interface. This is presented to the customer as, for example, an ATM Forum UNI operating at 25.6 Mbit/s via an RJ-45 socket. All functions relating to line rate adaption, asymmetric bandwidth and virtual path remapping are hidden behind the Homepoint in the active NT unit.

Warren Device Addressing



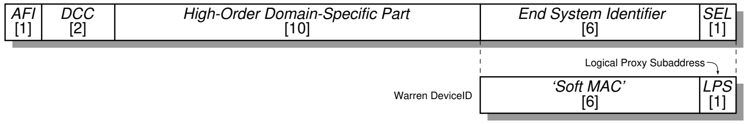
- Warren devices are designed to be as simple as possible and contain no 'MAC' address.
- Warren Control Protocol commands are sent to devices along the branches of a *spanning tree* by means of a *source routed* addressing scheme.
- Warren switches forward these *Warren Control Protocol* commands and support an *automatic topology discovery* scheme.

- Warren devices are designed to be as simple as possible and contain no built-in hardware ('MAC') address.
- Devices are controlled by sending them simple Warren Control Protocol messages. A typical command pokes a value into the routing table of a switch.
- A spanning tree which includes all the devices in the Warren is formed by an automatic topology discovery algorithm implemented in the Controller.
- Each device then has a unique source route from the Warren Controller, along the branches of the spanning tree. These physical addresses are then used to send commands to the Warren devices.
- The switches are specially designed to offer a superset of normal ATM cell switching functionality. This includes handling Warren Control Protocol commands and supporting the topology discovery scheme.
- Warren Control Protocol commands are sent along two reserved VCIs: one for 'down' commands (towards the leaves of the spanning tree), the other for 'up' commands (towards the Controller at the root). These VCIs are reserved on all Warren links.
- Apart from this, all normal ATM in-band semantics are supported by the Warren switches, making the Warren transparent to UNI-compliant end stations.

  Clearly, such devices do not understand the Warren Control Protocol; special arrangements are made to support conventional in-band signalling. When the device attempts to signal, the stream is automatically detected by the Warren switch and redirected into a signalling agent in the Warren Controller which handles requests for and on behalf of the device by proxy.

Warren Network Addressing

- ATM addresses commonly incorporate a hardware MAC address, i.e. a static mapping between physical and network addresses.
- The physical addresses of Warren devices are topology-dependent, and subject to change as devices are moved around. A static mapping is useless in this context.

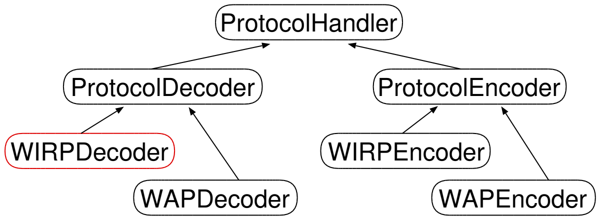


- Instead, every Warren object is assigned a *Soft MAC* by the Controller at creation time.
- The mapping of this identifier to the source route address is managed *intelligently* and *dynamically*.
- All connection management activities are carried out in terms of these Soft MAC identifiers.

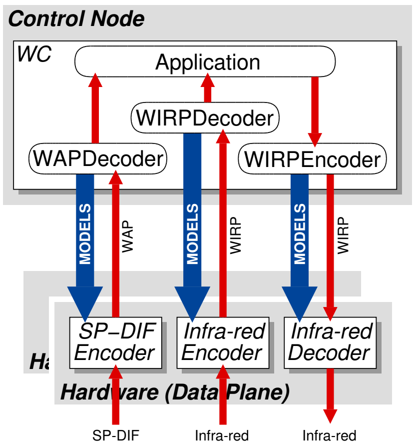
- Warren physical level addresses are based on *location*. In order to build useful and functionally rich applications, we need a location-independent network-level addressing scheme.
- This must also be able to interoperate with schemes used by conventional ATM devices and signalling protocols (e.g. Q.2931, PNNI).
- The 20-byte NSAP format addressing scheme incorporates the 6-octet MAC address of a device, i.e. a static mapping between network and physical address.
- Such a mapping is inappropriate for our needs because Warren devices have no unique MAC address. Moreover, in a home environment, there is a high likelihood that the location of a device will change over time as it is moved from room to room, so the source route address is of no use in this context either.
- A *dynamic* mapping is required between network identities and source route addresses. We call the former *Soft MAC addresses*.
- A Soft MAC identifier is assigned to each device at the topology discovery stage. The Warren Controller constantly monitors devices and applies intelligence when remapping identifiers.
- Individual logical proxies associated with a particular device are distinguished using a subaddress octet (mapped onto the *SEL* byte in NSAP format addresses).
- Not only real devices are assigned Soft MAC identifiers. *Virtual Proxies* – software emulations of imaginary devices – are also addressed in this manner. Even *Applications* are assigned Soft MAC identifiers.
- The interface to the Warren *ConnectionFactory* object enables a unidirectional connection to be setup (or torn down) between any pair of Warren objects. The objects may be real or virtual – any combination is possible.
- The dynamic address mapping scheme also comes in handy for accommodating mobile terminals in the architecture.

Warren Data Plane

- *ProtocolHandlers* are the gateways between Virtual Circuits and internal *EventStreams*. They enable objects within the Warren Controller to communicate in the *data plane* with real devices.



- The *ProtocolHandler* combines an ATM 'socket' with transcoding logic.



- Real devices are interconnected using conventional ATM Virtual Circuits; virtual devices are interconnected using an internal abstraction of a connection, the *EventStream*, which uses inter-object messaging.

- In order to connect a real device to a virtual one, a gateway called a *ProtocolHandler* must be interposed between the two conceptual connection types.

- The ProtocolHandler is an *adapter* between the network representation of a data stream (the ATM virtual circuit) and the control architecture's internal representation (an *EventStream*).

- It consists of a network end-point (an ATM 'socket') and some transcoding logic. For example, the *WIRPDecoder* accepts Warren Infra-red Protocol cells and decodes them into intermediate-level *Keysyms* such as 'play', 'pause', '4', 'select', etc. Higher level application are responsible for interpreting the semantics of these keystrokes.

- Returning to the earlier example of the CD player, the drive mechanism feeds SP-DIF and infra-red samples to the Warren line card for encoding and network transmission; in the opposite direction, infra-red control cells arriving from the network are decoded and injected into the control mechanism of the CD player.

Warren Application Support

- Applications need a *matchmaking* service in order to find devices to control.
- The control architecture enables searches based on *static* criteria (source/sink, media type, device class) as well as keeping track of *dynamic* information (location, current user identity).
- Much use is made of *infra-red* data. Transmitters and receivers on many end stations can be used to gather information about *physical* topology (proximity of devices and users)
- The *DeviceDatabase* offers an interface which applications can query to find out about devices or enquire about their status.
- This is work in progress.

- Before they can set up connections between devices, applications need to be able to locate them.
- The Warren Control Architecture incorporates *matchmaking* facilities which enable applications to find devices according to a number of static and dynamic criteria: source/sink, media type, device class, physical location.
- The ubiquity of infra-red transceivers on consumer electronics products can also be exploited. By sending out probe transmissions and gathering the received signals, information about the co-location of devices, and the proximity of users and devices can be deduced. This data is then useful in inferring and disambiguating the context of user requests.

Warren Application Level

- How is the behaviour of HAN applications specified?
- Is the user supplied with a repertoire of fixed-functionality applications, or is she allowed to customise the behaviour of the network?
- If the latter, what sort of 'programming' interface is she presented with? Manipulation of graphical flow diagrams, for example.
- Does the user interact with the system using a set-top box, or via a wireless handset with graphical capabilities?
- This is open research territory.

- Once the basic low-level application support is complete, we will need to build useful applications for the Warren.
- Command line interfaces are unlikely to appeal to users.
- It is likely that graphical user interfaces will be available via a set-top box. We can exploit the HTML and Java capabilities of such end devices. The Oracle Network Computer Reference Design (developed by Acorn Computers Ltd. of Cambridge), for example, already has an ATM25 NIC option.
- Alternatively, each user could be provided with a wireless control handset (using either an infra-red or radio data link). This device could have a small liquid crystal display with superimposed touch-screen. The software running on this *Rabbit* could either be fixed, or downloaded from the Warren Controller dynamically. A Smartcard slot would enable user authentication.
- Ideally, a range of application functionality should be available to the user:
  - Fixed functionality applications. A telephone application, for example.
  - Dynamic functionality applications. For example, with a baby monitor application, the user specifies the source device(s) to be monitored. A crying detection agent is instantiated automatically. The destination for alarms, however, is dynamic and depends on the room in which the parent is currently sitting.
  - Free-form applications. The system should provide facilities to enable the user to build her own applications. A graphical user interface could be used to connect devices (real and virtual) and agents together to form a composite application. The system should enable access to events such as key presses, temporal alarms and location information as conditional inputs to application components.
- All these issues have yet to be resolved.

Conclusion

- The Warren is an ATM-based Home Area Network.
- ATM does not necessarily imply high cost.
- By trading hardware complexity for software functionality, ATM networks can be built very cheaply.
- Open Signalling is one of the key technologies which enables us to achieve this goal.

- The Warren is a broadband Home Area Network based on ATM.
- ATM is wrongly perceived as an expensive networking technology.
- By trading hardware complexity in individual network devices for centralised software functionality, hardware costs are reduced to acceptable levels.
- Open Signalling enables us to exercise control over our network in a *consistent* and *device-independent* manner. As such, it is a key technology in supporting the Warren concept.