



Inetics™

White Paper

ImageStream Internet Solutions, Inc.

7900 East 8th Road
Plymouth, Indiana 46563

<http://www.imagestream.com>
info@imagestream.com

Phone: 574.935.8484

Sales: 800.813.5123

Fax: 574.935.8488

INTRODUCTION

ImageStream began developing WAN drivers for Linux in 1996. At that time, WAN card manufacturers were primarily focused on hardware, and to most, software was little more than a necessary evil. WAN card manufacturers commonly focused on OEM customers that wrote their own driver software, so there was little software vision beyond the goal of developing portable libraries that maximized the number of supported computing platforms.

When ImageStream entered the WAN card market, the most manufacturers offered generic driver designs that were inefficient, hard to scale, and difficult to maintain. These drivers were often the extensions of minimalist code that was developed to support the company's first revision hardware, and the monolithic designs would not scale properly in complex networking applications.

In response to this situation, ImageStream engineers went to work on a solution. The development team concluded that a new high-performance driver component architecture was needed. The driver component architecture would need to be integrated with a custom Linux kernel, and new software would be needed to manage the entire system. This development strategy was expected to shorten the company's development cycle, and provide an efficient platform that would be easier to port, scale, and maintain.

ImageStream's engineering team analyzed the limitations of existing driver designs which led to the development of the design objectives that are listed below.

1. Shorten development cycles
2. Enhance performance
3. Improve scalability
4. Increase memory efficiency
5. Provide a flexible OEM platform
6. Support third-party binary-only components
7. Upgrade standard and custom software seamlessly

The *Inetics*[™] platform was developed to address these requirements, and to provide a flexible development system to streamline the integration of new hardware and software over time.

Today, ImageStream's *Inetics* platform is a commercially successful networking software system that consists of the *Inetics* Driver Component Architecture (IDCA), an integrated custom Linux kernel, and management software that makes it all work together. ImageStream routers and WAN cards rely on *Inetics* technology. Telecommunications manufacturers like AT&T, Ericsson, and Lucent Technologies rely on *Inetics*. OEMs like Northrop-Grumman, Lockheed-Martin, and Westinghouse rely on *Inetics*. And *Inetics* is used in networking, telecommunications, radar, and satellite applications, by enterprises, governments, space agencies, military organizations, and military contractors.

OVERVIEW

An important part of the *Inetics* platform is the driver component architecture called IDCA, which specifies a standard software interface for binary driver components. *Inetics* also standardizes the processing stages that are used to manipulate network data streams. These standards provide a solid software architecture that meets all of ImageStream's application objectives.

Inetics uses a "process pool" of generic kernel threads that are designed to execute "chained" driver software components. A "chain" is an executable binary that is loaded into memory, linked into an execution list, and executed by a kernel thread from the process pool. As simple as it sounds, this modular design represents a significant advance in driver component technology that effectively abstracts the execution thread from the functional code that it executes.

Inetics can construct and modify unique lines of execution in real time for each I/O interface. Once *Inetics* has constructed the transmit and receive chains for an interface, different threads are used to execute those chains. Outbound data is queued, processed through the transmit chain, and then transmitted onto the physical network by the interface card. Incoming data is processed by an interrupt service routine (ISR) or polling thread that processes the inbound data through the receive chain.

The *Inetics* driver architecture defines standard processing stages so plug-in components can be used to manipulate inbound and outbound data streams. In essence, each processing stage is an execution list that can be modified by plugging in and removing different software components that manipulate the data streams at different stages of encapsulation. For example, a firewall manufacturer may need to manipulate IP encapsulation (Layer 3), while a WAN protocol stack developer may be more interested in writing software that can manipulate link encapsulation (Layer 2).

The stream processing stages defined for IDCA provide a standard programming interface for Layer 2 and Layer 3 developers. In addition, IDCA specifies chains for hardware I/O and interface setup. The following list provides the names of the *Inetics* driver components and a description of each.

Physical Interface Control (PIC):

Configures line interface hardware such as CSU/DSUs and ethernet transceivers.

Hardware Interface Control (HIC):

Hardware-specific driver component for network data input and output.

Encoded Data Processor (EDP):

Provides developer access to data link encapsulation.

Data Link Protocol (DLP):

Supports plug-in LAN and WAN protocol stacks.

Decoded Data Processor (DDP):

Provides developer access to network encapsulation.

The Physical Interface Control (PIC) is unique because it does not process network streams like the other *Inetix* driver components. The PIC chain is first executed when the *Inetix* software is loaded, and then again when a physical interface is reconfigured. The remaining *Inetix* processing stages are all dedicated to manipulating network streams at different levels of encapsulation.

Here is a data flow diagram of the four stream processing stages that *Inetix* executes for each network interface as network data is moved between the network hardware and the operating system.

HARDWARE \rightleftarrows HIC \rightleftarrows EDP \rightleftarrows DLP \rightleftarrows DDP \rightleftarrows OS

NETWORK HARDWARE

Description: Network interface hardware.

Function: Provides a hardware interface between the physical network and the host system.

Examples: Ethernet and token ring cards; other network cards with standard bus interface such as PCI.

HARDWARE INTERFACE CONTROL (HIC)

Description: Hardware-specific driver with interrupt service routine (ISR) and raw data I/O routines.

Function: Moves streaming network data between host memory and the network interface card.

Examples: PCI 201-ADSL HIC, PCI 530 series HIC, and PCI 1000 series HIC.

ENCODED DATA PROCESSOR (EDP)

Description: Processing stage for custom driver software components that manipulate Layer 2.

Function: Provides a chain for installing custom software that processes data link encapsulation.

Examples: Customized frame relay processing; test software that injects data link encapsulation errors.

DATA LINK PROTOCOL (DLP)

Description: Driver component for processing standards-based WAN protocols.

Function: Strips and encapsulates standard data link protocols.

Examples: Cisco HDLC, bisync HDLC, frame relay, PPP, multilink PPP, and ATM.

DECODED DATA PROCESSOR (DDP)

Description: Processing stage for custom driver software components that manipulate Layer 3.

Function: Provides a chain for installing custom software that processes network encapsulation.

Examples: Firewalls, tunnels, and VPNs; quality of service (QoS) and multicasting applications; test software to inject network encapsulation errors; and custom TCP/IP extensions.

The *Inetix* Driver Component Architecture (IDCA) separates the functional code (i.e. the plug-in driver components) from the execution units (i.e. the transmit and receive threads) so the system can construct unique lines of execution in real time for each network interface. This capability makes the *Inetix* platform one of the most powerful driver component systems available.

Inetix makes it possible to equally support commercial and Open Source developers as they focus on different aspects of network application development. In this context, the *Inetix* stream processing engine delivers extensive support for both standards-based and custom networking applications.

For all of its networking-specific capabilities, there are only two elements that make *Inetix* specifically adapted to networking applications: (1) the device-specific support provided by the Hardware Interface Controls, and (2) processing stages that are specific to networking applications. By creating custom processing stages, the self-modifying component technology provided by the *Inetix* platform can be deployed as a stream processing engine for a wide range of non-networking applications including manufacturing and multimedia control, telemetry and guidance systems, and artificial intelligence.

FEATURES & BENEFITS

When you consider ImageStream's engineering objectives for *Inetics*, it may not be obvious how the platform achieves its stated goals. This section describes how *Inetics* accomplishes its mission.

1. Shorter Development Cycles

ImageStream's primary objective for *Inetics* was to shorten development times for both standard and custom networking applications. *Inetics* can save developers enough time that it can be difficult to compare monolithic development. Even so, *Inetics* produces considerable time-to-market savings that can be correlated to the features of the architecture.

For example, *Inetics* supports commercial and Open Source plug-in components. The time-saving benefit of a plug-in interface for networking components is obvious: it takes less time for developers to plug in components than it takes to write the first line of code.

Inetics can also speed development by providing modular support for shared Data Link Protocol (DLP) components. Under *Inetics*, a shared protocol component may be linked to any hardware interface in a system. This can save development time by eliminating the need to maintain redundant protocol stacks in multiple monolithic drivers.

Inetics may also reduce time-to-market by streamlining the overall development process. *Inetics* is a complete component-based stream processing platform, and it predefines thousands of software relationships and methodologies. Using *Inetics*, OEM developers can ignore most of the design issues that affect software installation, configuration, execution, and upgrades. In addition, *Inetics* provides optimized support for many scalable hardware features, and it handles all of the design decisions required to implement a scalable stream processing engine.

With *Inetics*, Systems Engineers can save a lot of development time by simply avoiding the need make significant architectural decisions. A highly complex, scalable networking application may take years to design and develop. In this context, using *Inetics* can represent savings on a scale of thousands of programming hours and millions of dollars.

Moving deeper into the platform, *Inetics* reduces development time by organizing complex code into easy-to-understand processing stages. These predefined stages eliminate many opportunities for introducing bugs into the system, and they make it easier to isolate bugs once they have occurred.

Finally, *Inetics* is designed to save time when developing custom network applications. *Inetics* makes it possible for OEMs to use standard components and to modify their applications in isolated processing stages. *Inetics* defines where developers should install custom code for processing Layer 2 and Layer 3 encapsulation, so little time is spent deciding where to put the code. *Inetics* also saves time by isolating system functions and allowing the OEM developer to focus on the minimum breadth of the application.

2. Increased Performance

Compared to monolithic drivers, component drivers require support code that introduces measurable performance overhead. Some developers adopt monolithic designs because of this issue alone. When ImageStream set out to implement *Inetics*, it was clear that the platform would need to provide performance enhancements that would offset most of the architectural overhead.

Inetics improves performance by reducing the number of branch decisions that are usually present in a monolithic driver. For example, most monolithic drivers use numerous variables to keep track of the protocol configuration for each network interface. When each packet is processed, the driver must test a protocol selection variable to determine which protocol should be used, and then branch to the associated code based on that variable. *Inetics* eliminates the need for protocol selection variables by linking the correct protocol component into the execution chain for each interface.

Inetics also improves driver performance by allowing developers to focus on more narrowly defined efficiencies. As designers narrow their focus down to the efficiency of only a few lines of code, they can easily realize incremental performance enhancements in testing and optimization.

Finally, *Inetics* can also improve system performance by providing support for packet co-processing and embedded downloadable code. These capacities support the quick deployment of DSP hardware to support CPU-intensive applications, and provide architectural support for distributed processing.

3. Improved Scalability

Scalable networking systems can be expanded to support an application's growth over time. To OEM developers, "scalability" also means that a system can be modified to support different standards and custom applications. In this context, *Inetics* makes it easy for manufacturers, OEM developers, and end users to scale networking applications efficiently.

The capacities of a networking system are always constrained by hardware. Common scalable hardware features include the speed and number of processors, RAM and long-term storage capacities, and the number of network interfaces per expansion card. Most scalable hardware features also rely on software support to make them work. For example, a router that has a capacity of 18 WAN cards must also provide software that supports different hardware configurations with as many as 18 cards.

The ability to scale software can be affected by everything from the selection of a variable to the most important architectural feature. As a result, *Inetics* had to be created from the ground up, to ensure that the system would scale properly from application design through testing, and from product deployment through upgrades.

The *Inetics* platform provides support for a number of scalable hardware and software features. For example, the ability to integrate support for new hardware and new WAN protocols is an important scalability feature. The *Inetics* plug-in component interface makes it very easy to add incremental support for new hardware. The *Inetics* component interface can also provide enhanced scalability by supporting off-the-shelf driver components that may be used to scale a system. And *Inetics* supports the ability to scale network hardware by providing a stream processing engine that can be configured to support any number of network interfaces as they are added to the system.

Inetics was also designed to be highly scalable in multi-processing systems. For example, *Inetics* provides thread-based event processing which scales well in a multiprocessor environment. In addition, the separation of processing stages makes it possible to plug in software components that off-load stream processing to dedicated processing units such as DSPs, NPUs, and additional CPUs.

Finally, *Inetics* is designed to improve scalability for different kinds of developers and end users. For example, *Inetics* can bridge the development gap between the WAN card manufacturer and the OEM developer by providing separate stages for each to install stream processing code. As the manufacturer works to deploy standards-based networking applications for the general market, OEMs can work in parallel to deploy and maintain separate interdependent code for custom applications. *Inetics* provides these scalable features for developers which directly translate into application scalability for the end user.

5. Shared Common Code for Enhanced Memory Efficiency

Inetics can reduce RAM consumption when compared to monolithic device drivers. When a developer adds support for a new protocol stack to a monolithic driver, the software grows to consume more memory. When one protocol stack is copied into the monolithic drivers for several different network interface cards, the code is duplicated in memory each time a different driver is loaded.

Inetics implements a shared protocol interface that overcomes this problem. The *Inetics* driver component architecture supports plug-in Data Link Protocol (DLP) components that are shared among the different network cards in a system. With shared protocol components, all existing and future network interface cards are supported without wasting memory on redundant code.

By definition, a monolithic driver includes all of its code in a single binary file. If a monolithic driver supports the PPP, HDLC, frame relay, and ATM data link protocols, then those protocols must be embedded in the monolithic code. When a monolithic driver is loaded, all of the protocol support code is loaded even if it is not used.

Inetics' plug-in support for different data link protocols makes it possible to load only those protocol components that are being used. This ability represents a useful memory efficiency when compared to monolithic drivers that cannot unload unneeded code.

6. Flexible OEM Support

Inetics supports rapid application development for low-level OEM networking applications. Sections 1, 3 and 7 of this white paper describe some of the architectural capacities of the *Inetics* platform including support for off-the-shelf components, reduced development cycles, and enhanced application scalability for developers. These capabilities make *Inetics* a flexible OEM application platform, which was an important design objective for the system.

7. Seamless Upgrades

Most OEM developers have low expectations regarding the drivers provided by hardware manufacturers. Most developers are pleasantly surprised when the source code compiles and works as advertised. As a developer makes progressive changes to a monolithic driver, it becomes more difficult to leverage future software updates that may be supplied by the manufacturer. In some cases, an OEM developer can spend more to produce an upgrade than it cost to develop the original application. Because of these

challenges, some OEMs choose not to integrate driver updates from the manufacturer. This can seriously impair the manufacturer's ability to deploy new technology with its OEM partners, and it can leave customers exposed to serious security risks.

Inetics was designed to streamline the ability for both the manufacturer and OEM developers to quickly and efficiently deploy new software in the same end user application. To accomplish this, *Inetics* provides a plug-in architecture where standard and custom components may be defined for Layer 2 and Layer 3 operations.

When an OEM deploys a specialized *Inetics* application, the developer's custom components are normally inserted into the Encoded Data Processor (EDP) and Decoded Data Processor (DDP) chains. This approach makes it possible for OEMs to update custom components without affecting the standard components provided by the manufacturer. Similarly, OEM developers can deliver software updates from the manufacturer to end users without affecting the OEM's custom software.

In practice, a manufacturer might supply a T1 frame relay card with three *Inetics* driver components including a hardware-specific Physical Interface Control and Hardware Interface Control, and a frame relay Data Link Protocol component. An OEM could deploy this hardware and software in a standard routing application as the manufacturer had intended. But the OEM might also develop custom components that work with standard components to deliver support for a custom application such as voice over frame relay (VoFR). The OEM developer might choose to implement call control and other network-based voice services through a custom Decoded Data Processor, and then install custom quality of service (QoS) extensions for frame relay as an Encoded Data Processor.

Developers at any level can use the available processing stages in the *Inetics* driver component architecture for their own purposes. For example, an OEM developer might also want to design a standard Data Link Protocol module that is not available off-the-shelf. A card manufacturer might design an EDP module that works as a plug-in compatibility interface to support non-standard implementations of a standard protocol. Like most object systems, *Inetics* driver components work the best when standard software components are designed for the most common applications, and then custom components are written to handle the exceptions.

With all of these different standard components, the *Inetics* platform had to make upgrades easy for developers and end users. In most cases, a complete *Inetics* software upgrade requires little more than copying new files into place and restarting the *Inetics* system. Operations like this can be automated easily under Linux.

APPLICATIONS

Inetics's real-world capabilities may not be obvious to the uninitiated. In most of the applications described in this section, the *Inetics* platform involves driver components, a tightly integrated custom Linux kernel, and management software to support different routing and streaming data applications.

1. Wireline Routers - *ImageStream Internet Solutions, Inc.*

Router products make an ideal OEM application for *Inetics*. ImageStream developed *Inetics* with routing as a reference application, and the *Inetics* platform is used to operate every ImageStream router. The packet forwarding engine is actually provided by the Linux kernel, itself. Companies that want to develop applications on top of a proven router platform can save a lot of time and money using the *Inetics* platform.

2. Wireless Routers & Hot-spot Solutions - *ANTlabs*

Wireless access points and "hot-spots" provide wireless connectivity to end users. These wireless systems must inevitably interface with wireline systems that carry data to and from the WAN or Internet. These demanding broadband systems often require DS3/E3, OC3/STM1 or faster connectivity. ANTLabs supplies Singapore Telecom (SingTel) with hot-spot technology using ATM connections and the ImageStream router platform running *Inetics*.

3. Internet Access Devices (IADs), All-in-One Systems & Point-of-Sale Products - *Various Systems Integrators and OEMs*

Internet access devices (IADs), integrated all-in-one router-firewall-server systems, and point-of-sale solutions often require high-speed WAN or Internet connectivity to accomplish their design goals. Systems integrators and OEMs use *Inetics* to provide a wide range of WAN and Internet connectivity solutions to the automotive industry.

4. Network Security - *Intrusion.com*

Inetics is an ideal platform to deploy in a wide range of network security applications including firewalls, tunnels, virtual private networks, and decoy networks. With Linux, many of these security applications are available under Open Source licensing, and do their work on the operating system's networking layer which is part of a complete *Inetics* system.

ImageStream routers provide a Linux-based IP translation firewall for internal network security. This firewall uses IP tables to provides a capable chained component interface and scalable rules engine that make it unnecessary to implement those features from within the *Inetics* platform. *Inetics* can also be used to implement custom firewall applications and security analysis engines. Custom security applications that use the *Inetics* platform may also be developed and deployed in tandem with Open Source security applications that work outside the *Inetics* system.

5. Real-Time Network Testing & Surveillance - *Network Instruments*

Inetics supports a wide range of network test and surveillance applications. These applications may be

similar to the general security applications described in the preceding section entitled “Network Security,” or they may focus on network performance and protocol analysis. The main differences between these applications is the software, not the hardware and driver component architecture. *Inetics* supports ImageStream complete line of network taps that are used to monitor network traffic in test and surveillance applications. Network Instruments uses ImageStream building block in their commercial network analysis solutions.

6. Telephony Systems - Ericsson

Inetics can be used to support a wide range of telephony applications. For example, *Inetics* can support conventional multiplexing applications and stream processing for the H.110 telephony bus. For many years, Ericsson has used *Inetics* to multiplex cellular telephone calls onto T1 and E1 lines in cell site installations around the world.

7. Satellite Equipment - NORSAT & Surrey Satellite

With the appropriate network interface card, *Inetics* can be used to interconnect different kinds of synchronous satellite uplinks, downlinks, and land lines. Many satellite systems transmit data without any link layer encapsulation, and ImageStream provides off-the-shelf DLP modules for raw IP and ASCII streaming data to support this application. NORSAT and Surrey Satellite use *Inetics* in satellite interface applications because it is a powerful platform for processing digital satellite transmissions.

8. Lab Test Equipment - AT&T Labs, Spirent

ImageStream excels in test equipment applications. The *Inetics* software works with ImageStream network hardware to provide a wide range of test solutions for developers. With ImageStream building blocks, developers of test equipment can focus on their software, because *Inetics* makes it easy to inject and analyze network data streams at different network layers.

9. Laboratory Research - Duke University, Johns Hopkins University, DePaul University, Iteso University (Mexico), Advanced Science & Technology Institute (Phillipines)

Inetics excels in laboratory applications where an incredible range of analysis software must be interfaced at multiple levels with device driver software. *Inetics* makes it easy to develop laboratory applications, with the modular capacity to inject raw data streams at layer 2 and 3, and independently probe data streams at each layer, as well.

10. Aeronautics & Telemetry - Naval Undersea Warfare Center (NUWC),

Northrop-Grumman, Lockheed-Martin, Raytheon, Westinghouse & Matsushita Avionics

Inetics can be used to process streaming data in many navigation and telemetry applications. In some cases, *Inetics* may be used to interface with radar systems that require the manipulation of a synchronous bitstream. In others cases, *Inetics* is used to encode long haul lines from monitoring sites. *Inetics* is truly a network chameleon that can quickly connect unique data streams from radar and telemetry stations to standard network interfaces and legacy logging systems.

THE FUTURE

Inetcs for Linux may already be the most powerful networking platform on the planet. With its extensive support for scalable networking applications, *Inetcs* has already created new possibilities for the development of competitive networking products.

In many ways, the future of *Inetcs* can be extrapolated from the past. For example, CPU horsepower increases every year to allow off-the-shelf components to be deployed in a wider range of networking applications. At the same time, Linux and *Inetcs* continue to become faster and more powerful. Each year, new *Inetcs* components are developed to support new hardware interfaces and off-the-shelf protocols. Custom *Inetcs* modules are also developed each year to support plug-in processing for custom applications. These advances have made it possible for *Inetcs* to be deployed in some of the most demanding applications in the telecommunications industry, and these historical trends are expected to continue.

A part of *Inetcs*'s future can be expressed as a function of ImageStream's development plans. For example, ImageStream plans to develop *Inetcs* support for USB-based devices and high-end switch fabrics with distributed interface processing. But some areas of ImageStream's *Inetcs* development are not known in advance. This is because ImageStream is a market-driven company, and customers often request unanticipated features that may be added to future software releases.

In the final analysis, *Inetcs* is already a highly capable networking platform. The future of *Inetcs* will be driven by new hardware, new Linux performance enhancements, new Open Source development projects for networking and telephony, and ImageStream's vision of what a router can be. But over time, the needs of ImageStream customers will drive *Inetcs* development more than any other factor.