XIPC

Technical White Paper
Software Version 3.3.0

January, 2002

Table of Contents

Introduction 3

What is Envoy XIPC 4

Envoy XIPC Features 5

Who can use Envoy XIPC 6

Envoy XIPC Capabilities 8

Application-Level Configuration 9

Synchronous Functionality 10

Aynchronous Functionality 11

Event Driven GUI 12

Message Queuing 13

Synchronized Memory Read and Write Operations 16

Triggers 18

Real-Time Memory and Debugging 19

Introduction

The vast majority of computer applications under development today are comprised of multiple processes (or threads) of execution operating in concert with one another to perform the application's objectives. Such applications are referred to as multitasking applications when all the processes reside on a single platform, or distributed applications when some of the processes are separated by a network. The communication among these multiple processes is called Interprocess Communication (IPC).

IPC is at the heart of client/server, peer-to-peer and other forms of distributed applications. In fact, IPC is required for almost any application employing two or more cooperating processes. It is not surprising, then, that the selection of an IPC mechanism can greatly influence the measure of success in the development, deployment and execution performance of such applications.

The major operating systems provide IPC facilities for supporting process-to-process communication within a single platform. These facilities, while providing a relatively complete set of services (message queuing , semaphores, and memory sharing), are bound in range by the walls of the platform. Far more limited facilities are available for supporting process-to-process communication over a network. Developers must choose between these two facilities for supporting their application's IPC functionality - based on process location.

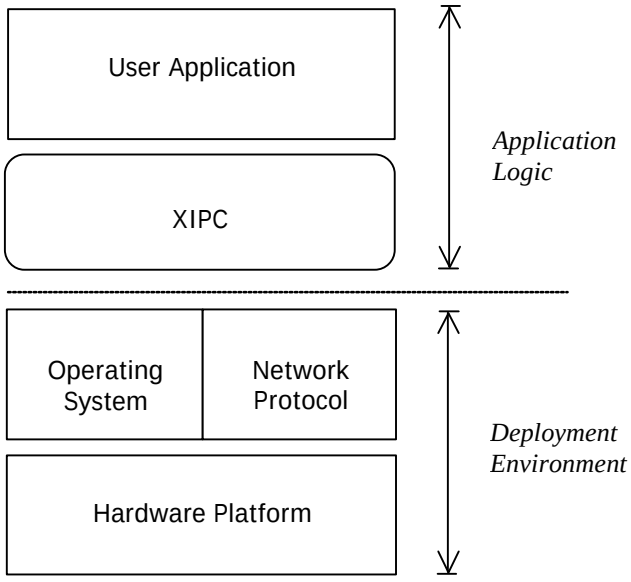
XIPC, by contrast, applies the standard programming model for multitasking IPC (message queues, semaphores, shared memory), and makes it completely operating system and network transparent. To the XIPC programmer, the network appears as a single multitasking operating system. With the addition of high-level functionality and extensive monitoring and debugging aids, XIPC enables the rapid development of sophisticated multitasking and network-transparent distributed applications.

What is Envoy XIPC

Envoy XIPC is a middleware tool that treats IPC programming as a single problem - regardless of the location of the involved processes, and regardless of the variety of operating systems, network protocols and hardware platforms being used. Envoy XIPC essentially permits the developer to conceptualize the programming environment as being a virtual multitasking environment wherein processes communicate and interact with each other using a *single* high-level programming model, independent of the physical positioning of the processes.

Envoy XIPC 's singular IPC approach effectively disengages the programming logic of an application from the issues of application topology and process deployment. The programming logic governing the interaction between processes within an application is unrelated to the positioning of the processes, and is unaffected by their occasional repositioning.

Envoy XIPC lies between an application and its underlying environment. Envoy XIPC defines an Application Program Interface (API) for performing interprocess communication operations in a manner that is independent of the underlying operating system, network protocol and hardware platform. Changes to the underlying deployment environment do not affect the application built on Envoy XIPC.



Envoy XIPC Middleware

Envoy XIPC Features

- ***Productivity:*** Envoy XIPC, eliminates the need for network programming skills. As a result, the cost of developing and maintaining distributed applications is significantly reduced.
- ***Connectivity***: Envoy XIPC's programming model allows processes of a distributed application to appear local to one another - client to server, peer to peer.
- ***Interoperability:*** The Envoy XIPC's programming model is preserved even when processes are spread over a network of dissimilar computer platforms.
- ***Portability***: Envoy XIPC, being a portable API, provides for immediate source code portability between dissimilar platforms. The IPC portion of an application need only be written once.
- ***Scalability***: Applications written using Envoy XIPC's are inherently scalable. Applications can be built on a single machine and then deployed to varying degrees over a network, without changing source code.

Who can use Envoy XIPC

The Gartner Group has noted, "Leading edge IS departments and ISVs with the need to develop sophisticated cooperative applications should consider Envoy XIPC for their program to program functions. Properly applied, Envoy XIPC has the potential to reduce the time, cost and technical risk of developing complex applications."

Envoy XIPC can be employed for constructing elegant solutions to a wide range of distributed computing applications. Examples of application that are well suited for Envoy XIPC include high-performance messaging or transaction oriented systems, multi-tier client/server applications and process control or real-time applications.

High-Performance Transaction and Messaging Applications
Envoy XIPC 's message queuing mechanisms provide the means for implementing high-performance messaging and transaction routing capabilities within stand-alone or distributed applications. The Envoy XIPC programming model, being network transparent, allows such functionality to operate uniformly regardless of the location of participating processes (local to one another or distributed over a network). This flexibility is of particular importance when deployment scenarios for the application are either unknown at the time of the application's development or are intended to vary.

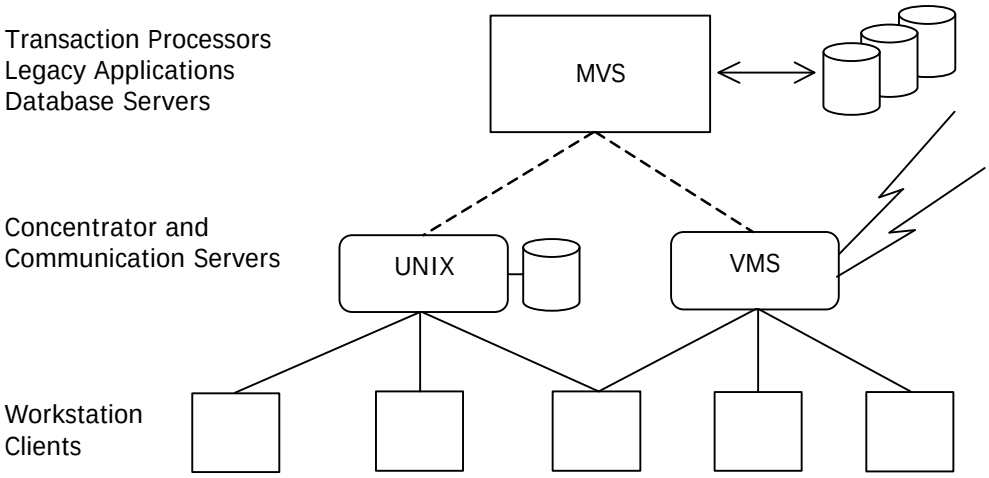
Envoy XIPC message queuing performance is extremely fast - making it ideal for implementing high-volume messaging applications. The high performance is complemented by features that support advanced flow-control functionality, including:

- ***Flexible message queue configuration***: Envoy XIPC queues are individually sized, can support extremely large capacity requirements and are optionally able to spool overflow messages to disk.

- ***Message burst capabilities***: Envoy XIPC provides the means for sending and receiving sustained bursts of application messages for supporting high-bandwidth throughput requirements.

- ***Traffic management***: Envoy XIPC messaging applications can be structured to react automatically to surges of message traffic by widening the message bandwidth (more queues) or by accelerating message throughput (more processing programs).

Multi-Tier Client/Server Applications
The Envoy XIPC programming model is well suited for implementing applications that are logically and/or physically layered as multiple tiers, with each tier supporting an autonomous level of functionality within the application - and where the tiers interact dynamically with each other. These applications generally exhibit the following architectural appearance:



Multi-Tier Application

Such architecture is often associated with legacy applications that are being downsized into client/server distributed environments. Component processes in such architecture require the means for communicating, synchronizing and sharing data with each other in a generally asynchronous manner. Envoy XIPC's rich sets of asynchronous capabilities simplify the logical expression and coding implementation of such requirements.

Real-Time Applications

Envoy XIPC 's wide range of IPC mechanisms (queues, semaphores and shared memory), and operation modes (synchronous and asynchronous) position it as an ideal tool for building complex process-control and real-time applications - where the interaction between processes is not easily expressed using a simple message passing paradigm.

Component processes within such applications generally require a more finely granular set of IPC tools for supporting the process-to-process synchronization requirements of the application. Envoy XIPC 's extensive IPC capabilities are ideal for building such systems. Such implementations have the added feature of being portable between different deployment environments.

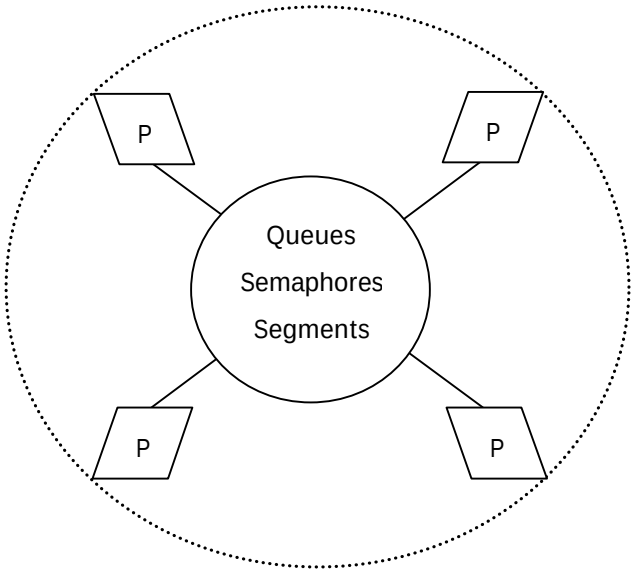
Envoy XIPC 's easy integration with native operating system I/O mechanisms additionally allow for the integration of Envoy XIPC level I/O data streams. It is thus possible to have a process wait simultaneously for operating system *and* Envoy XIPC activity, using the standard set of operating system semantics.

An example of such an application is one in which processes must interoperate with voice recognition equipment while simultaneously communicating with other application processes. It is possible to uniformly multiplex the voice and Envoy XIPC data streams using the native operating system I/O facilities.

Envoy XIPC Capabilities

The Envoy XIPC environment

An important Envoy XIPC concept is that of an *instance*. An Envoy XIPC instance is an environment for supporting Envoy XIPC objects - message queues, semaphores and memory segments. Envoy XIPC objects are created, manipulated and deleted over time, as needed. The processes of a Envoy XIPC application interact with each other by coordinated manipulation of Envoy XIPC objects. This relationship is depicted below.



A typical XIPC application

Envoy XIPC instances are themselves dynamic. They may be created as needed, used to support certain Envoy XIPC activity for a period of time, and then discarded when no longer needed. All of this occurs without disturbing the activity of the native operating system, or the activity of other Envoy XIPC instances.

Application-Level Configuration

One of the more unique aspects of Envoy XIPC instances is that they are application-level entities. A major benefit of this approach is that each application is able to configure its IPC environment according to its own specific needs. This is important where multiple applications, having different and perhaps divergent IPC requirements, are to run concurrently.

One application may be a high-volume message routing application where the messages are relatively small, a second may be an imaging application that supports the periodic transfer a large-sized messages. A third may be a new distributed application that is currently under development. Using Envoy XIPC, the three applications can run simultaneously, insulated at the IPC level from one another, all benefiting from customized, application-specific configuration settings.

There is no longer a need to arrive at a compromise set of IPC parameter settings for best addressing the conflicting IPC needs of *all* applications in the environment.
A second benefit of application level configuration is that the management of IPC resources is moved outside of the operating system environment. Configuration of a Envoy XIPC instance has no disruptive affect on kernel or other platform activities. It is accordingly easy to "play" with an application's Envoy XIPC environment for testing and tuning purposes.
Finally, Envoy XIPC instance configurations are portable. The resultant Envoy XIPC environment is identical regardless of the underlying operating system.

Application-Level Naming
Envoy XIPC supports natural character string names for identifying its entities. A two-tier naming structure is supported. At the higher level are Envoy XIPC instances - each with an identifying character string name. Instances, in turn, each support a unique name-space of Envoy XIPC objects that are identified, as well, using character string names.
The benefits of using natural character string names for identifying Envoy XIPC instances and objects are:

- ***Ease of programming****:* Working with meaningful IPC identifiers promotes the development of self-documenting and hence more easily maintained application code.

- ***Portability****:* Utilization of character string names for identifying Envoy XIPC entities guarantees the full portability of an application's Envoy XIPC environment when moved about among dissimilar platforms.

The benefits of Envoy XIPC's two-tier approach are:

- ***Name space insulation****:* The name space of Envoy XIPC objects within an Envoy XIPC instance is unique to that instance. This eliminates any chance of a "naming collisions" occurring between unrelated applications. It is thus possible, for example, to run multiple applications simultaneously, where each one utilizes its own "*InputQueue*".

- ***Application development****:* It is similarly possible to run multiple versions of a single application without naming conflicts. An example of such a situation would be to run an application's production version, while simultaneously working on a subsequent development version.

Synchronous Functionality

In the flurry of IPC activity occurring within an application, certain operations will potentially block due to the state of the IPC objects involved. Examples of such behavior include: a process issues a request to receive a particular message from a message queue on which no such message currently exists, or a process attempts to write to an area of shared memory that is currently locked by another process, or a process blocks while waiting for a group of events to occur. Envoy XIPC supports three forms of behavior for reacting to these situations in a synchronous manner:

- ***No Wait****:* The operation returns an error code indicating that the desired operation could not be completed at the current time. No blocking occurs.
- ***Wait****:* The operation blocks the calling process until the operation is able to complete.
- ***Time Out****:* The operation blocks the calling process for a user specified period of time, while waiting for the operation to complete.

Asynchronous Functionality

An alternative approach for such situations is to have the operation complete asynchronously, in the background, without blocking the mainline logic of the calling process. The benefits of such an approach are many - the most significant being that such a mechanism allows an application's distributed processes to execute *concurrently*.

With Envoy XIPC, it is natural to initiate concurrent operations involving multiple network platforms, and to have them run to completion in parallel. This has the direct affect of leveraging the inherent parallelism of a network environment.

The operation is initiated with a Envoy XIPC asynchronous option specified. The three supported asynchronous options are:

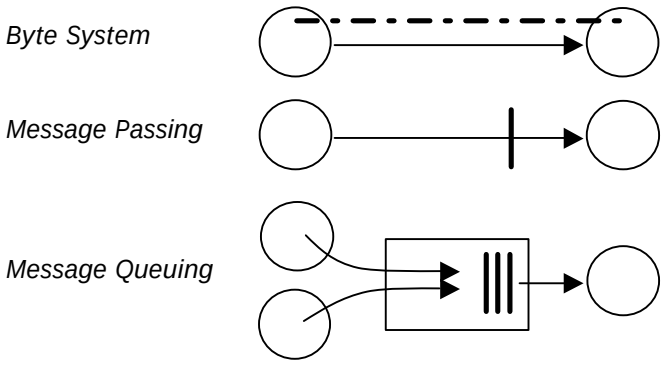
- ***Callback****:* Envoy XIPC notifies the process of the operation's completion by executing a user-specified function.
- ***Post****:* Envoy XIPC notifies the process of the operation's completion by setting a user-specified Envoy XIPC event semaphore.
- ***Ignore****:* Envoy XIPC completes the asynchronous operation silently.

Event-Driven GUI Programming

Envoy XIPC 's asynchronous capabilities are powerful tools for incorporating event-driven functionality within distributed applications. This is particularly important when part of the application is required to interface with an event-driven Graphical User Interface (GUI) such as MS-Windows or X-Windows. Envoy XIPC logic for managing an application's process-to-process communication can be naturally integrated within the event-driven programming environments of such GUIs.

Message Queuing

Distributed applications typically require the means for establishing some form of data-flow between underlying processes. This requirement has historically been met in a variety of ways:



Message queuing benefits from a number of advantages over byte stream and message passing mechanisms:

- ***"m-to-n" connectivity****:* Envoy XIPC message queuing inherently supports the ability for multiple processes to exchange messages over a single channel, (i.e., an *m-to-n* relationship). Byte stream and message passing mechanisms, being premised on a process-to-process communication orientation, are generally unable to support such functionality.

- ***Asychronousity****:* Message queuing is inherently asynchronous. Sending a message onto a queue can occur independent of whether the processing of previously sent messages have been completed.

- ***Staggered execution****:* Message queuing additionally provides a means for application processes to send and receive interprocess messages at staggered times - without requiring simultaneous execution of the communicating processes. This is generally not possible using conversational mechanisms.

- ***Flow control****:* Message queues inherently provide a buffering capability for supporting variable traffic rates between communicating processes. Using Envoy XIPC message queues it is possible to explicitly control the extent of such elasticity in terms of messages or bytes. This is managed as part of Envoy XIPC's individual queue sizing.

  Message prioritization*:* Envoy XIPC message queuing supports message prioritization and selection. It is possible to retrieve messages based on FIFO, LIFO and a wide range of priority specifications. Byte stream and message passing are, by comparison, primarily FIFO mechanisms.

Message queuing is accordingly recognized as the more advanced data-flow mechanism. Envoy XIPC message queuing is particularly advanced in that it supports a number of unique queuing features.

Message Queue Sizing
Envoy XIPC message queues are created dynamically with individual capacity limits. The limits may be specified in terms of maximum message capacity, maximum total byte capacity, both limits, or neither. Using this capability it is possible to throttle the flow of message traffic within an application's message queues on a queue-by-queue basis.

Individual queue sizing provides the architectural flexibility needed for tailoring each message queue to the traffic flow intended for it to support. Real-time communication applications, for example, often require large-capacity message queues that surpass the built-in limits of certain native operating system queuing services. Envoy XIPC queues are not subject to such limits.

Message Queue Overflow Spooling
Complementing the specific sizing capability is the ability to employ message queue overflow spooling on a queue-by-queue basis, as needed. When spooling for a queue is active, the message queue is given elasticity for accepting messages beyond its normal capacity by spooling overflow messages to disk until space on the queue becomes available. At that time, the queue automatically absorbs the spooled messages.

Overflow spooling can be triggered to react automatically to heavy traffic surges. This can be particularly useful for handling real-time message feeds that cannot be throttled without losing messages.

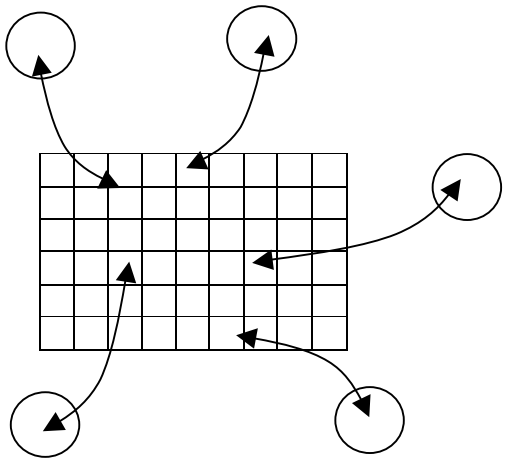
Message Selection
Envoy XIPC provides the developer with much flexibility in sending and receiving messages to and from message queues. Of particular note is the ability to operate atomically on multiple message queues. For example, one can retrieve the highest priority message, or the oldest message, from across multiple queues. Similarly, Envoy XIPC permits the dispatching of messages onto the shortest of a group of queues - yielding automatic queue balancing.

This feature permits the logical division of message traffic within an application, across multiple queues, while continuing to support atomic access to the queues as a single unit. This functionality is valuable for transaction processing applications where the relationship between server processes and transaction queues are to be adaptive in response to different traffic profiles.
A process can dynamically widen or shrink the bandwidth of queues being serviced by means of this feature.

Message Queue Browsing
Envoy XIPC message queue functionality includes the ability to momentarily freeze message queue traffic so as to traverse, examine, prune or even rearrange the messages currently held on a queue. Access to queue data while the data is still queued is a prerequisite for building high-performance message filtering programs. Similar functionality is not available using most native queuing services.

Memory Sharing

Envoy XIPC shared memory segments are used for supporting memory sharing between processes of an application. Using Envoy XIPC shared memory segments; processes are able to share memory-resident data with each other.



Network transparent shared memory

Besides supporting the ability to read and write segment data, Envoy XIPC also introduces a number of innovations in managing access to shared memory.

Synchronized Memory Read and Write Operations

Envoy XIPC shared memory read and write operations are guaranteed to execute atomically, regardless of the amount of data involved. Envoy XIPC, as such, enforces serialization of data access to its memory segments. It is thus not necessary to manually enforce serialized access to shared memory for preventing overlapping read and write operations - as is the case with native operating system shared memory services. This is done automatically by Envoy XIPC.

Byte-Level Access Controls
Envoy XIPC introduces the notion of shared memory access controls. These mechanisms permit the dynamic impositions of byte-level read or write access limitations over all or parts of Envoy XIPC memory segments. Particular areas of segments can be made read-only or entirely inaccessible for long or brief periods of time. This can be used for implementing in-memory data tables requiring such forms of data protection.

Shared Memory Locking
A specific form of access control is that of memory locking. A process can lock all or arbitrary parts of memory segments, on the fly, for performing updates to those areas. Memory locking can be specified with byte-level granularity. Read or write operations attempted against a locked area of memory are prevented until the memory is subsequently unlocked.

Distributed applications that require memory-resident data to be globally available to multiple distributed processes, (and for which using a DBMS would be an expensive overkill), can employ Envoy XIPC's memory sharing as an elegant high-performance solution. With it, it is possible to attain many of the data sharing benefits of a DBMS without the accompanying overhead.

Semaphores
Envoy XIPC semaphores are used for supporting the event synchronization and resource management occurring among processes of a distributed application. Semaphores are especially useful for expressing interprocess synchronization relationships in a concise manner. The setting of a single semaphore by one process can immediately influence the execution of many other distributed processes.

Envoy XIPC supports two classes of semaphores: event semaphores and resource semaphores.

Event Semaphores
Envoy XIPC event semaphores are general-purpose flags that can be *set* or *clear*ed in response to the occurrence of application events within a distributed application. Processes waiting for one or more such events to occur wait for the respective semaphores to become set. A process detecting the occurrence of an event notifies the other processes by setting the appropriate semaphore.

Event semaphores can be used for affecting simple and concise forms of one-to-many interprocess communication. By setting a single semaphore, a process can notify hundreds (or thousands) of other processes of an important piece of application-related information for them to react individually. An example of this is a stock price monitoring system that uses semaphores for notifying all interested parties when selected stocks rise or drop across certain threshold values. This form of leveraged communication between processes is unique to event semaphores.

Resource Semaphores
Resource semaphores are counting mechanisms that are used for controlling access to limited-supply resources within an application. Processes competing for access to such resources negotiate access to them by means of resource semaphores. Distributed applications that manage facilities or devices, over which there is more demand than supply, can be designed in a network transparent manner using resource semaphores.

As with other Envoy XIPC facilities, it is possible to arrange that the resource synchronization requirements of an application be performed in an asynchronous and network transparent manner. This is often a characteristic of distributed process control applications where access to physical devices and other equipment must be controlled with network transparency.

Operations Involving Multiple Semaphores
Envoy XIPC *provide*s the means for having a process wait synchronously or asynchronously on a list of events or resources. Such operations can be specified in three forms. When expressed within the context of event semaphores, they are:

- ***ANY*** - Wait for any of the listed events to occur.
- ***ALL*** - Wait for all of the listed events to occur cumulatively, over time.
- ***ATOMIC*** - Wait for all the events to occur simultaneously.

Similar specifications can be made when waiting for lists of Envoy XIPC resource semaphores to become available. Envoy XIPC's ability to group semaphores atomically within single operations provides the means for expressing elegant solutions to complex synchronization requirements.

There are significant advantages for using semaphores to support an application's event synchronization activity, as opposed to employing a messaging mechanism. Building such functionality using messaging tools requires that the application implement its own internal mapping system between messages and events, that additionally tracks "who is waiting for what" and finally, that remembers the appropriate notification logic for each waiting process. Besides being a non-trivial programming exercise, such an approach is indirect in concept and thus awkward to apply.

Envoy XIPC semaphores, by contrast, provide a concise and direct programming model for implementing an application's interprocess synchronization logic.

Triggers

Envoy XIPC introduces an additional form of asynchronous functionality - that of Envoy XIPC triggers. Envoy XIPC triggers are mechanisms that continuously monitor the state of activity within an application's IPC environment. Envoy XIPC triggers operate asynchronously to the application's logic - only coming to life when the monitored situation occurs.

Envoy XIPC triggers open up a wide range of interesting IPC programming techniques. It is possible, for example, to have an application set triggers that notify it when:

- A specific message queue rises above 75% (or some other threshold) of its message capacity. The application can react in a variety of ways, such as enabling queue overflow spooling (in anticipation of an overflow), or by starting more consumer processes to relieve the pressure on the queue.

- A specific area of shared memory is written to. Processes in the application, waiting for such an update, can react by making note of the new data.

- A specific process removes a specific message from a queue. The process that produced the message can be automatically notified of the message's arrival at the consuming process.

Envoy XIPC triggers, when combined with the general asynchronous command options, provide the means for building application logic that is entirely event-driven. As was mentioned earlier, this is significant when the application must interface with an event-driven Graphical User Interface.

Real-Time Monitoring and Debugging

Envoy XIPC includes full-screen interactive monitors that provide continuous real-time views of the activities occurring within an application's IPC environment. The monitoring facility does not require any special preparation of the application being monitored. It can be invoked to examine the IPC status of production applications in the field, without any extra provisions, and without incurring performance overhead in the application when monitoring is not in use.

When invoked, monitoring becomes an invaluable tool for identifying IPC problems occurring within an application. Users of Envoy XIPC have described its real-time monitors as being worth their entire investment in Envoy XIPC technology.

Envoy XIPC 's monitors provide critical support to distributed application developers throughout the application life cycle.

Application Development
The benefits of developing distributed applications with Envoy XIPC and its monitoring tools are best measured in terms of development time saved. The ability to watch a network application execute with all of its IPC activities exposed - and to have it do so in slow motion - is of immeasurable value when tracking down allusive coding bugs. The total development time of distributed applications can be cut significantly by means of Envoy XIPC 's monitors alone.

Application Testing
The extent to which an application's performance is tested and analyzed during its test period is usually an accurate predictor of how reliably the application will run in the field. Envoy XIPC 's monitoring tools provide detailed information regarding an application's IPC activity for studying the strong and weak links within an application's architecture. High water marks, low water marks, asynchronous operation backlog, text fragmentation and message throughput characteristics are examples of information available using the monitors.

Application Maintenance
Envoy XIPC monitors are support tools as well. Applications built using Envoy XIPC are inherently monitorable *after* they are shipped. No longer is it necessary to recreate problems on special "debug" versions of an application in order to study the problems. Envoy XIPC -based applications can be studied wherever they are - as soon as they begin to exhibit unexpected behavior.

Envoy XIPC monitors have a spreadsheet "look and feel". Information appears in a matrix-like display where processes and IPC objects form the axes of the matrix. Interactions between processes and IPC objects are displayed within the body of the matrix. The following diagram depicts the general layout of a Envoy XIPC monitor screen.

Monitor Modes
Envoy XIPC monitors update the display of ongoing Envoy XIPC activity in one of the following modes:

- ***Interval Snapshot Mode****:* Refreshes the monitor display at a user-specified millisecond interval.

- ***Trace Flow Mode****:* Refreshes the monitor after every Envoy XIPC operation, and pauses a user-specified number of milliseconds between each Envoy XIPC operation. This permits an application's IPC activity to be viewed in slow motion.

- ***Trace Step Mode****:* Similar to Trace Flow Mode except that the monitor completely stops the application after each Envoy XIPC operation. The user presses a key to single-step to the next operation. This mode is ideal for intensive IPC debugging exercises.

Zoom Windows
Envoy XIPC monitors provide zoom windows that permit developers to focus on the activity of specific aspects of an application's IPC environment, while continuing to monitor the general ebb and flow of IPC activity occurring within the application. Wide ranges of zoom windows are available, including:

- Zoom in on a particular message queue. It is possible to watch the actual message traffic as it moves through a queue.

- Zoom in on a semaphore. It is possible to track all the processes that are waiting for a particular event, and to observe the order that they blocked and will be woken up.

- Zoom in on a particular process. It is possible to watch every IPC operation being performed by a particular process within an application.

Browse Windows
A second, and more powerful, monitoring mechanism is the monitor's full screen browse windows. Browsing provides a complete and frozen view of all the message queues and memory segments within an application. It is possible to "walk" through, and about, all the messages on a queue, or to examine the contents of shared memory segments. It is additionally possible to search for character string or Hexadecimal patterns of data within message queues or shared memory.

Using the browse window, it is possible to confirm the format, length, contents, priority, offset and sequencing of messages on a queue, as well as analyzing the contents of shared memory segments.

Watch Windows
The most advanced form of Envoy XIPC monitoring is achieved using the monitor's watch windows. With them it is possible to observe the contents of memory segments changing in real-time, or at a user-specified update rate. It is additionally possible to observe the locking and unlocking of memory segment areas, as they occur in real-time.

Interactive Command Interpreter

Envoy XIPC provides an interactive command language for executing all of Envoy XIPC's operations interactively, or from within an operating system shell. This eliminates the need for writing C-Language programs in situations when ad-hoc IPC operations are necessary. This capability is particularly useful for testing an application's handling of various IPC scenarios that may be otherwise difficult to create through the normal execution of the application.

Using the interactive interpreter, it is additionally possible to initiate asynchronous Envoy XIPC operations that, upon their completion, kick off operating system commands. One such example would be to have an operating system disk backup utility triggered by the asynchronous arrival of an Envoy XIPC message, or by the setting of a Envoy XIPC semaphore.