

## LINX

### Questions & Answers

#### What is LINX?

LINX is an interprocess communications (IPC) service used for both local communications within a CPU and remote communications in distributed systems. LINX provides transparent communications, meaning that processes and applications that use LINX for communications use the same API regardless of where the communicating peer is located. LINX is designed for scalability and performance, and can be used on any platform and operating system.

#### Why is LINX needed, what problem does it solve?

Many of today's systems are heterogeneous, utilizing a mixture of operating systems, CPUs, microcontrollers, DSPs and media interconnects, such as RapidIO, Gigabit Ethernet, or shared memory. This complicates communications between processes, applications and nodes, which often utilize different IPC mechanisms within the same system. For instance, processes located on the same CPU may communicate via the IPC provided by the operating system used on that particular CPU (for instance mailboxes or other methods). But those same processes, when communicating with processes located on another CPU, or node, might use a different method like TCP/UDP/sockets. LINX simplifies system design by providing a single IPC that spans all CPUs, operating systems and interconnects, regardless of where the communicating processes physically reside.

#### Isn't IPC typically something that comes with the operating system used?

Yes, typically it is. And in fact, LINX is based on the IPC method used in Enea's OSE operating system. However, LINX is operating system independent. It can, for example, be used on Linux. It can also be used between applications running on different operating systems, such as a combination of Linux and OSEck (OSE for DSPs). Equally important, LINX is well suited for *both* local (within a CPU) and remote/distributed communications (between CPUs / nodes), making it ideal as a system-wide IPC.

#### What IPC model is LINX based on?

LINX uses a message-based communications model that is *asynchronous* and *direct*. Messages are sent directly from sender to receiver, not indirectly via mailboxes. Messages are also sent asynchronously, meaning that the sender can continue execution directly after the message is sent, as opposed to having to wait for a synchronizing message from the receiver. This IPC model is superior for distributed systems with both high performance and high-availability requirements.

## **How can the LINX communication service be described technically?**

LINX can be seen as both an interconnect (i.e., Ethernet, RapidIO, shared memory) and a transport service (such as ATM, UDP, TCP, etc), spanning the Session, Transport and Link layers of the standard OSI model.

## **LINX is designed for scalability, what does that mean?**

LINX scales both upwards and downwards. It has a small footprint, which makes it ideal for CPUs and DSPs with little memory available. It also supports any network topology, no matter how complex. Even when networks grow very large (number of nodes), the size of LINX remains manageable. This is possible because LINX dynamically builds address maps for links and application connections at run time, but *only* maintains those connections and links that a given node needs. This greatly reduces the memory needed to store address maps.

## **LINX is designed for high performance, what does that mean?**

To be suitable for exclusive system-wide interprocess communications, the IPC method must deliver high performance. LINX accomplishes this by employing a lightweight connection-oriented protocol that delivers outstanding performance both for intra-node and inter-node communications, over both reliable and unreliable links. LINX performance figures are available on the Enea web site.

**Enea claims that LINX is significantly faster than TIPC by running benchmarks in a Linux environment. Everybody knows that there are many ways to implement a protocol in Linux, affecting speed, memory consumption, robustness, GPL contamination etc. Aren't the differences in performance depending on the Linux implementation choices rather than the protocols themselves?**

The difference in performance is both due to protocol design and due to implementation. Roughly the following aspects of a protocol affects its performance:

- Overhead
- Timeouts
- Retransmission strategy

Overhead comes in two forms, header size which affects every packet and packets not carrying any data. LINX tries to minimize both although header size only becomes significant when transmission time dominates processing time.

Timeouts affects how quickly nodes recover from packet loss or crashing peers. There are tradeoffs involved here, short timeouts allows peers to react faster but may cause unnecessary retransmission.

Retransmission strategies affects who detects lost packets and decides who initiates retransmission, and how quickly lost packets are retransmitted. More complex strategies do a better job than simpler ones at the expense of bigger footprint and perhaps more complex common case. For LINX we have assumed that packet losses happen rather infrequently and thus tried to optimize the common case (no lost packets).

And the following can be considered implementation:

- Optimizations
- Algorithms
- OS/driver features
- Locking strategies

We don't think LINX is more optimized than TIPC. LINX is in early stages of implementation and optimization has only begun, TIPC ought to have come much farther in this area given the time that it is has been available.

In a protocol of this kind what's most affected by algorithms is how packets are matched to processes. On Linux that is mostly given by the kernel environment and its socket implementation.

OS/Drivers, e.g. how a component make use of operating system resources, are again rather similar between LINX and TIPC.

Locking strategies becomes important when many processes send and receive messages in parallel. Here we believe that LINX is much better than TIPC - we are faster when many streams and many packets are in flight at the same time. In performance profiling measurements, LINX makes better use of CPU utilization to achieve significantly better throughput performance.

**What is the basis for Enea's claims of better performance on Linux than TIPC on Gigabit ethernet? It seems that latency figures are nearly identical for small packets of control information, which is the more common case for protocols such as LINX and TIPC.**

Latency figures for LINX and TIPC are nearly identical for small packet sizes up to about 1Kbyte. LINX starts doing better on packet sizes greater than 1.5K on Ethernet (MTU size). This is primarily due to the fact that LINX does a better job of in fragmentation and de-fragmentation of messages greater than MTU size. However, the real performance benefit of LINX is seen in throughput (or max throughput). LINX is superior by an average of 15-25% for all message sizes - this is primarily due to two things: 1) the threading model for LINX on Linux, which makes better use of available CPU bandwidth and 2) the fact that LINX has a traffic adaptation algorithm that can maximize throughput for hardware like Gigabit Ethernet that supports different modes of interrupt handling based on traffic rate. See question 1 above for some other reasons.

On other media, e.g. RapidIO, or shared memory, with higher bandwidth, the throughput difference between LINX and TIPC would be even bigger. In general the latency differences will be more pronounced for interconnects with smaller MTU (or max packet sizes), like RapidIO. For shared memory latency will be about equal for all message sizes.

## **Why does Enea make LINX for Linux open source?**

From a community/industry perspective, we believe that LINX is the best IPC technology for complex distributed systems. An important benefit of using a good IPC is that it simplifies the integration of applications from different communities and vendors. In order for such an IPC technology to become widely used, however, it must be open and accessible to all.

## **How does Enea use LINX in its own products?**

LINX provides the foundation for Enea's multi-platform DSO (device software optimization) solution. LINX enables Enea's applications, services and products to interact seamlessly with not only each other, but also with applications, services and products from multiple vendors and user communities. This is a win-win situation for both Enea and its customers.

## **How and when can I get access to LINX?**

LINX for Linux and OSEck are available for customer evaluation now. Contact any Enea office for more information. The LINX release is planned for June 2006. LINX for Linux will also be available as open source technology at that time.

### **Corporate Headquarters**

P.O. Box 1033  
Skalholtsgatan 9  
SE-164 21 Kista, Sweden  
Phone: +46 (0)8 507 140 00  
Email: [info@enea.se](mailto:info@enea.se)  
Web: [www.enea.com](http://www.enea.com)

### **US Headquarters**

2635 North First Street  
Suite 118  
San Jose, CA 95134  
Phone: 408-323-9480  
Toll-free: 866-844-7867  
Email: [info@enea.com](mailto:info@enea.com)  
Web: [www.enea.com](http://www.enea.com)

### **Asian Headquarters**

1-4-2 Kanda  
Ogawa-machi, Chiyoda-ku  
Tokyo, Japan  
Phone: +81 3 5207 6167  
E-mail: [osesales\\_jp@enea.se](mailto:osesales_jp@enea.se)  
Web: [www.enea.com](http://www.enea.com)



**Embedded for Leaders**