Lazy Receiver Processing (LRP): A Network Subsystem Architecture for Server Systems

Wired & Wireless Network Security Lab.
Hanyang University
Park jin ho

Paper info

- Authors
  - Peter Druschel and Gaurav Banga
    Department of Computer Science, Rice University, Houston
- Appears in
  - Proceedings of the 2nd USENIX Symposium on Operating Systems Design and Implementation (ODSI), Seattle, WA, Oct 1996.

Contents

Background

- Background
  - Many operating systems use an interrupt-driven network subsystem architecture that gives strictly highest priority to the processing of incoming network packets.
  - This leads to scheduling anomalies, decreased throughput, and potential resource starvation of applications.
  - Furthermore, the system becomes unstable in the face of overload from the network.

Goals

- Goals
  - Proposes a new network subsystem architecture which provides stable overload behavior, fair resource allocation, and increased throughput under heavy load from the network.

UNIX Network Processing

- Overview

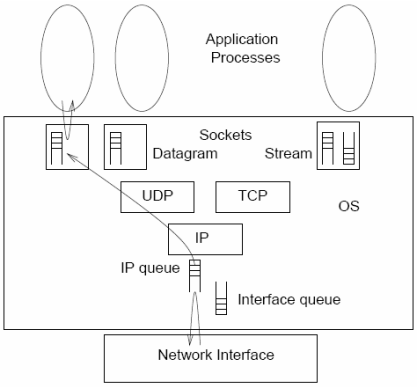

Figure 1: BSD Architecture

UNIX Network Processing

- Overview
  - On the receiving side
    - Interrupt handler
      - The arrival of a network packet is signaled by an interrupt.
      - The interrupt handler encapsulates the packet in an mbuf, queues the packet in the IP queue, and posts a software interrupt.
    - In the context of this software interrupt
      - The packet is processed by IP.
      - UDP's or TCP's input function is called, as appropriate.
      - The packet is queued on the socket queue of the socket that is bound to the packet's destination port.

UNIX Network Processing

- Overview
  - On the sending side
    - In the context of the user process
      - Data written to a socket by an application is copied into newly allocated mbufs.
      - UDP
        - mbufs are then handed to UDP and IP.
        - The resulting IP packets are then transmitted.
        - If the interface is busy, place the packet in the driver's interface queue.
      - TCP
        - mbufs are queued in the socket's outgoing socket queue.
        - Calls TCP's output function and IP's output function.
        - IP packets are transmitted or queued on the interface queue.
    - Interrupt handler
      - Data remove from the interface queue and transmit.

UNIX Network Processing

- Priority

- Whenever a user process is interrupted by a packet arrival, the protocol processing for that packet occurs before control returns to the user process.
- The reception of subsequent packets can interrupt the protocol processing of earlier packets.

UNIX Network Processing

- Problems
  - Eager receiver processing
    - Processing of received packets is strictly interrupt driven, with highest priority given to the capture and storage of packets.
    - Second highest priority is given to the protocol processing of packets.
    - lowest priority is given to the applications that consume the messages.
  - Lack of effective load shedding
  - Lack of traffic separation
    - Incoming traffic destined for one application (socket) can lead to delay and loss of packets destined for another application (socket).
  - Inappropriate resource accounting
    - CPU time spent in interrupt context during the reception of packets is charged to the application that happens to execute when a packet arrives.
    - Since CPU usage, as maintained by the system, influences a process's future scheduling priority, this is unfair.

UNIX Network Processing

- Sources of High Network Load
  - Simultaneous requests from a large number of clients
  - Misbehaved distributed applications
  - Incorrect client protocol implementations
  - Malicious denial-of-service attacks
  - Broadcast and multicast traffic

Design of the LRP Architecture

- Overview

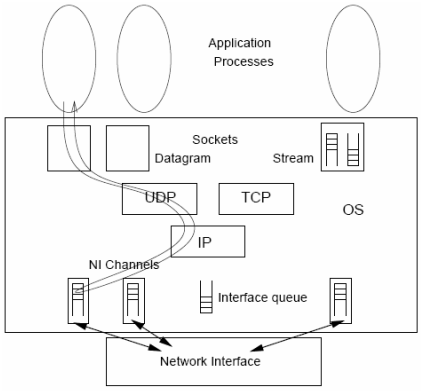

Figure 2: LRP Architecture

Design of the LRP Architecture

- Overview
  - The IP queue is replaced with a per-socket queue.
  - The network interface demultiplexes incoming packets according to their destination socket, and places the packet directly on the appropriate receive queue. If receiver queue is full, silently discarded (early packet discard).
  - Receiver protocol processing is performed at the priority of the receiving process.
  - Protocol processing is performed lazily, in the context of the user process performing a receive system call.

13

Design of the LRP Architecture

- Sockets and NI Channels
  - A network interface (NI) channel is a data structure that is shared between the network interface and the OS kernel.
  - It contains a receiver queue, a free buffer queue, and associated state variables.
  - Determines the destination socket of any received packets and queues them on the receive queue of the channel associated with that socket.
  - Effectively demultiplexes incoming traffic to their destination sockets.

14

Design of the LRP Architecture

- Packet Demultiplexing
  - Identify the destination socket of an incoming network packet.
  - So that the packet can be placed on the correct NI channel.
  - Demultiplexing performed by the NI itself (NI demux) or network driver's interrupt handler (soft demux).

Design of the LRP Architecture

- UDP protocol processing
  - The transmit side processing remains largely unchanged.
  - On the receiving side
    - Network interface determines the destination socket of incoming packets and places them on the corresponding channel queue.
    - When a user process calls a receive system call on a UDP socket, the system checks the associated channel's receive queue.
    - Performed in the context of the user process performing the system call.
  - reduce context switching and increase memory access locality.

Design of the LRP Architecture

- TCP protocol processing
  - Protocol processing is slightly more complex for a reliable, flow-controlled protocol such as TCP.
  - Receiver processing cannot be performed only in the context of a receive system call, due to the semantics of TCP.
  - Because TCP is flow controlled, transmission of data is paced by the receiver via acknowledgments.
  - The solution is to perform receiver processing for TCP sockets asynchronously when required.
  - Several ways of implementing asynchronous protocol processing (APP).

17

Design of the LRP Architecture

- **Other protocol processing**
  - In the TCP/IP suite, this includes processing of some ARP, RARP, ICMP packets, and IP packet forwarding.
  - In LRP, this processing is charged to daemon processes that act as proxies for a particular protocol.
  - These daemons have an associated NI channel, and packets for such protocols are demultiplexed directly onto the corresponding channel.

18

Experimental Evaluation

- Experimental Results

System	round-trip latency (μsecs)	UDP throughput (Mbps)	TCP throughput (Mbps)
SunOS, Fore driver	1006	64	63
4.4 BSD	855	82	69
LRP (NI Demux)	840	92	67
LRP (Soft Demux)	864	86	66

Table 1: Throughput and Latency

- Latency was measured by ping-ponging a 1-byte message between two workstations 10,000 times.
- UDP throughput was measured using a simple sliding window protocol.
- TCP throughputwas measured by transferring 24 Mbytes of data.

19

Experimental Evaluation

- Experimental Results

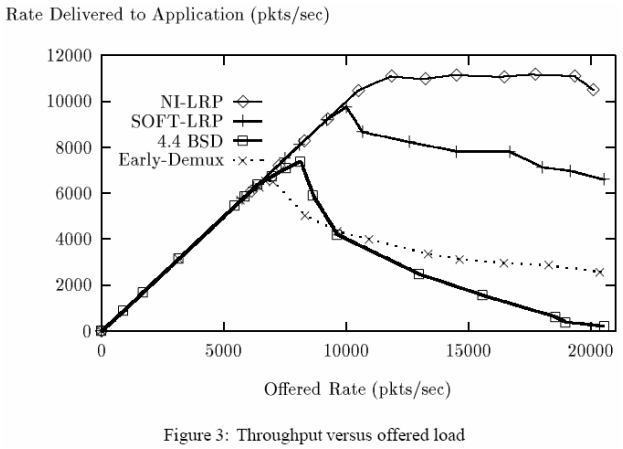

Figure 3: Throughput versus offered load

20

Experimental Evaluation

- Experimental Results

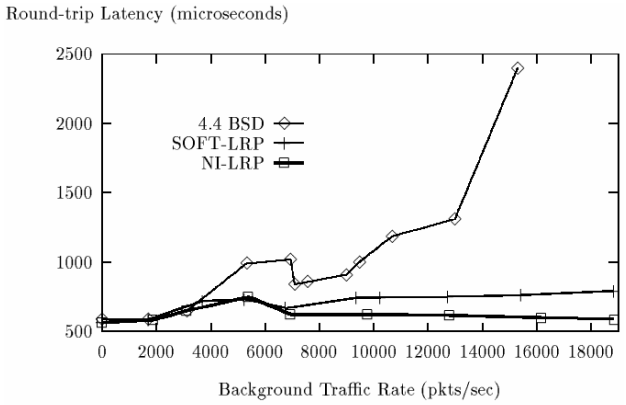

Figure 4: Latency with concurrent load

Experimental Evaluation

- Experimental Results

RPC	System	Worker elapsed time (secs)	Server (RPCs/sec)
Fast	4.4BSD	49.7	3120
	SO-LRP	38.7	3133
	NI-LRP	34.6	3410
Medium	4.4BSD	47.1	2712
	SO-LRP	37.9	2759
	NI-LRP	34.1	2783
Slow	4.4BSD	43.9	2045
	SO-LRP	38.5	2134
	NI-LRP	35.7	2208

Table 2: Synthetic RPC Server Workload

Experimental Evaluation

- Experimental Results

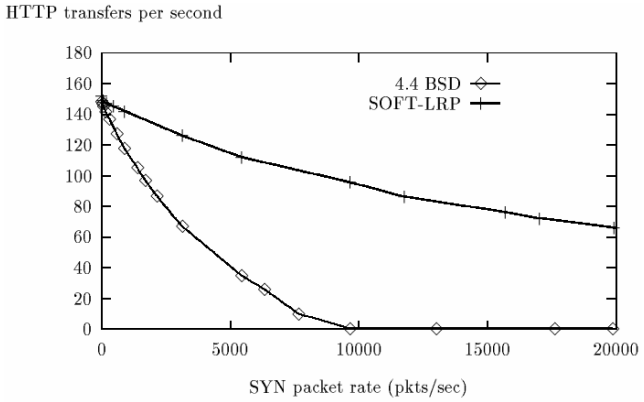

Figure 5: HTTP Server Throughput

Conclusion

- Summary
  - Under conditions of high load, LRP architecture offers increased throughput, stable overload behavior, and reduced interference among traffic destined for separate communication endpoints.
  - Delayed processing of received network packets reduces context switching and can result in increased server throughput under high load.
  - Combination of early packet demultiplexing, early packet discard, and the processing of incoming network packets at the receiver's priority provide improved traffic separation and stability under overload.