Network Interrupt Service Routine

Robert N. M. Watson

16 May 2007

FreeBSD Developer Summit
BSDCan 2007
netisr

• Historically
  – Could not access structures from interrupt handler
  – Asynchronous execution for network processing
  – Piggy-backed on interrupt handling facilities (ISR)

• Today
  – Used to avoid recursion, re-entrance, deadlock
  – Direct and deferred work dispatch handler
    • Direct dispatch executes in current thread
    • Deferred dispatch executes in worker thread
Registration and Dispatch

- Work managed at packet granularity
  - E.g., Link layer to protocol dispatch
  - E.g., Tunnel decapsulation dispatch
- Subsystem registers name and handler
  - netisr_register(NETISR_IP, ..., ip_input, ...)
- Packet source picks protocol + dispatch model
  - netisr_dispatch(NETISR_x, m)
  - netisr_queue(NETISR_x, m)
- Deferred processing executes in swi kthread

16 May 2007
## Use and Abuse

<table>
<thead>
<tr>
<th>Service name</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETISR_USB</td>
<td>Scheduling only</td>
</tr>
<tr>
<td>NETISR_POLL</td>
<td>Scheduling only</td>
</tr>
<tr>
<td>NETISR_POLLMORE</td>
<td>Scheduling only</td>
</tr>
<tr>
<td>NETISR_ROUTE</td>
<td>Routing socket input</td>
</tr>
<tr>
<td>NETISR_ATALK1</td>
<td>Appletalk level 1 input</td>
</tr>
<tr>
<td>NETISR_ATALK2</td>
<td>Appletalk level 2 input</td>
</tr>
<tr>
<td>NETISR_AARP</td>
<td>Appletalk address resolution input</td>
</tr>
<tr>
<td>NETISR_ATM</td>
<td>ATM input</td>
</tr>
<tr>
<td>NETISR_ARP</td>
<td>ARP input</td>
</tr>
<tr>
<td>NETISR_IP</td>
<td>I Pv4 input</td>
</tr>
<tr>
<td>NETISR_NETGRAPH</td>
<td>Scheduling only</td>
</tr>
<tr>
<td>NETISR_IPV6</td>
<td>IPv6 input</td>
</tr>
<tr>
<td>NETISR_IPX</td>
<td>IPX input</td>
</tr>
<tr>
<td>NETISR_NATM</td>
<td>NATM input</td>
</tr>
</tbody>
</table>
netisr future

- Direct dispatch now the default in 7-CURRENT
  - Avoid context switches and lowers latency
  - Enables parallelism between network layer input processing across different interfaces, but ...
  - ... disallows parallelism between interface ithread and network layer input processing.

- Netisr2 prototype in Perforce
  - Moves from swis to per-cpu kthreads
  - Will be used for loopback and tunneled traffic
  - Ordering is a key design concern
Key code paths to inspect

- netisr.h
- ip_input.c: ip_init(), ip_input()
- if_ethersubr.c: ether_demux()
- netisr.c:
  - netisr_register(), netisr_unregister()
  - netisr_dispatch(), netisr_queue()
  - start_netisr(), swi_net(), netisr_processqueue()