Distributed Debugging
A systematic process and tool strategy for trouble shooting distributed real-time applications.

OMG Real-Time & Embedded Workshop
July 9-12, Arlington, VA

Gordon A. Hunt
Chief Applications Engineer, RTI
Agenda

- Distributed System Debugging Problem(s)
  - Soft and Hard Failures
- The Three Constraints…
  - Memory
  - CPU
  - Networks
- The Smoking Gun…
- Tools and possible tools…
History of “Bugs”

- This first bug…
  - 1945 in the Harvard Mark II System

- Bad bugs in history
  - Mariner Space Probe – transcription error
  - Therac-25 medical accelerator – software interlocks
  - Unix finger/bind daemon – buffer overflow
  - AT&T network outage – message crashes neighbors
  - Ariane 5 Flight 501 – integer overflow
  - Mars Climate Orbiter – units
  - National Cancer Institute, Panama City – use case
It Only Gets Harder…

- Unique Distributed System Challenges
  - Things are getting more distributed
  - Soft failures are much more common in these systems
    - Traditional debugging techniques don’t easily apply
    - Don’t readily know what everything is really doing
    - Systems live well beyond the scope of their original requirements
  
- You won’t ‘own’ the entire system
  - Current automated testing techniques can’t cover the scope and scale of the actual system
Categories of Debugging

- Integration Debugging
  - Logical malfunctions

- Stress Debugging – at a point it breaks
  - Scalability
  - Message Loss
  - Stability Problems

- Random Event Debugging – what’s going on??
  - Packet loss
  - Numerical/algorithmic glitches
Integration Debugging: Design for it...

- Subject of another whole talk...
- Use of tools for type specification
- Strong interfaces for software components
- Model sub-system and system interactions
  - For Example: Quality of Service specifications
  - Including temporal dependancies
- Verify existing applications
  - Check interfaces, QoS settings, state behaviors,...
Stress and Random Event Debugging

Requires a Systematic Debugging Process

- Series of “Questions” and Tests
  - Drill down into problem areas
- Determine likelihood of root cause
  - CPU, Memory, Logic, Network
- Provides a way of focusing your attention
  - Leverage appropriate tools to validate answers
  - Tries to minimize chasing invalid assumptions
- Everything is on the table
  - OS, Middleware, 3rd party products, application code
  - Hardware, network configuration, algorithms
A Process for Debugging…

- Getting the right data?
  - Integration Problem?
    - Network Problem?
      - Node Problem?
    - Data Interpretation Problem?
      - App Logic Problem?
- Doing the right thing?
  - Tools?
Questions Groups
– For performance and functional issues

- Application Dependencies
- Data Handling
- Programming Language
- Operating System(s) Specifics
- CPU Performance
- Application Logic
- Network Hardware
- Network Topology
- Message Protocol
- Message Handling
- Multicast
- Time Synchronization
- Dynamic Events
- Tools & Debugging Information

The Limiting Factors
- CPU
- Memory
- Network
Application Dependencies

- Tool chain
  - Using a middleware?
  - What does it use?
- Hardware
  - Drivers
  - Bus: PCIExpress, PCI, USB
- Compilers
- OS
  - Its patch level
- Services that OS, Application, Middleware use
  - ARP, DHCP, IGMP
Programming Language and Deployment

- **Java**
  - Considering garbage collection processing?
  - JDK compatibilities?

- **C/C++**
  - Recompiling all files after header/object changes?
  - STL usage? Implementation differences?

- **Dynamic libraries**
  - Are you loading the right one? Getting old version?
Operating System(s) Specifics

- What OS(s)?
  - What versions and configurations?
- Windows? Linux? Solaris?
  - How real-time is the application?
- Embedded OS? VxWorks? Lynx? QNX? Integrity?
  - What BSP and hardware?
- How many Ethernet nics?
  - Device drivers
  - BSD based IP stack?
- Does the application use file I/O?
  - On what device?
  - Other I/O on board?
- Socket configuration and buffer sizes?
- Relative priorities of running process and threads
- What tool support is there?
CPU Performance

- Average load?
  - How do you measure?
- Peak loads?
- Profiling all processes?
- Trace tools used?
  - What are the receive threads doing?
- Dynamic process?
  - Or once initialized is the application space static?
- Thread priorities?
- Kernel processes
Network Hardware & Topology

- What is the network topology?
- Are packets duplicated, how is this managed?
- What is the throughput of the switches?
- Multi NIC hosts
  - What do the routing tables look like?
- GigE?
  - Half or full duplex
  - Is everything configured the same?
  - Using Jumbo packets?
  - Are Ethernet drivers up to date?
- Other Ethernet concerns
  - Collision domains?
  - What are the MTUs on various links?
- Multicast configuration
  - How many multicast addresses in use?
  - Will networking hardware honor IGMP join/leave messages?
  - TTL limits and scoping boundaries for multicast addresses?
Data/Packet Handling

- Message characteristics
  - Multiple messages per packet?
  - How is endianness handled?
  - Word aligned?
  - Size?

- Reliability
  - Are all messages reliable?
  - Prioritized?

- Delivery models
  - Broadcast, multicast, unicast?
  - Multiple architectures?

- Rates of data
  - What are the sustained rates?
  - Bursts? How long, how much?
  - Periodic data?
  - Asynchronous data?
Message Handling

- Where is queuing done?
  - In the sending and receiving applications
  - How big are the queues?
  - Can multiple unacknowledged messages be outstanding?
    - How many?
  - Messages out of order, dropped, missing, etc.
  - What happens with an out of order message?
    - Dropped? Queued? Error notification?
    - Detected how? Replace original or dropped?
  - Does each message require acknowledgement?

- Sequence numbers?
  - How does the reliability mechanism work?
  - How many resends? What timeout?

- Messages sent using multiple NICs?

- Component failure
  - Is failure isolated to specific messages?
  - Specific components?
  - Repeatable?
Applications Logic - Sending

- What thread/process sends the data?
  - Thread priorities?
  - Which thread is responsible for resending data?
  - Is data sent immediately?

- How is the application notified of send timeout?
  - Blocking call? Timer? Callback?

- How is the resend timeout executed?
  - Within callback?
  - Which thread handles resend?

- Is data queued for later sending and resends?
  - What’s the size of the queue?
  - One queue per application or per message type?
  - Is dynamic memory being used in send?
  - Are other (non-network) resources being used in send?

- Is there any reporting of resource failures?
Application Logic - Receiving

- What thread(s) or process(es) handle incoming data?
  - Who owns the threads?
  - Single or multiple threads?
  - Single thread per socket? Per Port?
  - Dedicated threads for servicing data?
  - Threads blocked on ports waiting for data?
  - Which thread is responsible for sending acknowledgements?
  - Thread priorities?
  - If a receive thread is blocked, is any new data dropped?

- Acknowledgement indicate receipt of data or processing of data?

- Is data copied upon receipt?
  - Copied from network buffers into application buffers?
  - Copied from receive thread to processing thread?
  - zero-copy semantics?
  - Is data modified prior to copying? (deserialization, byte swapping, etc)
  - How is data validated?
Application Logic – Receiving (2)

- Dynamic memory being used in receive?
- Other (non-network) resources being used in receive?
  - System or blocking calls made while receiving or processing data?
- Message Queues?
  - OS? STL? select()? Mutex protected?
  - Fixed size?
  - What happens if space is not available?
  - Messages dropped by receive thread? Buffered?
- Reporting of resource failures?
- Multiple ports or single port?
  - How are send/receive ports determined?
  - What are the setsockopt() options used?
- How is corrupt or incorrectly addressed data processed?
- Are there deadline or timing requirements for processing the data?
  - Does data have an expiration time?
- Messages replace older messages that have not been processed?
Message Protocol

- TCP, UDP, both? Others?
- Are series of messages acknowledged?
- What happens if no acknowledgment is received?
- How is message fragmentation handled?
- Max message size?
- IPv4 or IPv6? Mixed?
Time Synchronization

- How is time handled in the distributed system?
  - Hardware based? Software based?
  - Frequency of updates?
- What is time synchronization failure?
  - How is it detected?
  - Handled?
- Time Source? (GPS, sys clock)
- Are packet/message loss decisions based on time or receipt of another message with an index number?
  - Both?
- What happens if time goes backwards?
- What happens if time jumps forward?
- What time synchronization resolution is required?
  - ms, us, ns?
  - Oscillator quality?
Dynamic Events and Initialization

- Initialization
  - What are the differences in steady-state operation versus turning the system on?
  - Does message loss occur during init or steady state?
  - How do you know initialization is complete?

- Dynamic events
  - Node failure and restart?
    - What happens? What does the network load look like?
  - New nodes join the systems?
  - Impact on existing nodes?
  - Memory? Event processing?
  - Does it cause a spike in CPU?
Tools & Debugging Information

- **Kernel**
  - Can a kernel visualizer or some other tool be attached?
  - Possible to instrument kernel?
- **Network**
  - Do switches allow captures of specific ports? All ports?
- **Do applications do any logging or data recording?**
  - How is it performed?
  - To what media?
- **Communication errors reproducible?**
  - Conditions/operating environment for at failure time documented?
  - All components reporting failure?
    - Or limited to specific subset?
- **Debugging capabilities of deployed systems**
  - Are there ways to determine what was happening from the application’s point of view (e.g. commands issued by the operator)?
  - Are there ways to determine what was happening on the network (packets on the wire)?
- **Use Timestamps!**
The Smoking Gun…

● There is certainly a trend in failure modes
  – Exceptions in receive threads
  – UDP receive socket buffer size
  – Group rate errors
  – Ethernet flow control
  – Treating all packet loss as bad

● However
  – UDP receive socket buffer size makes the most smoke
Packet Loss Misconceptions

- Caused by transmission errors, gamma rays, ...  
  - Actually most loss caused by buffer overflow
- No loss in properly configured/operating networks  
  - Totally normal
- Loss happens in the network  
  - Actually host network buffers are often to blame
- All packet loss is bad  
  - Signifies congestion, used for good
- Unicast and multicast loss are coupled  
  - TCP and multicast may follow different routing paths
Receive Socket Buffer Sizing

- Most OS have ‘small’ default values
  - UDP typically used for low volume query/response work (NTP, DNS, etc.)
- Easy to induce packet loss at the receiver with high data rates
- Need to know CPU scheduling latency when setting buffer sizes
  - Could have received multiple packets
- UDP buffer space not monitored like TCP
  - Packets simply discarded, and hopefully logged
Receive Socket Buffer Sizing

- **Too small**
  - Packet loss, increasing latency, bandwidth loss, and additional CPU usage, and memory to hold out of order packets

- **Too big**
  - Slower recovery

- **Know your default sizes**
  - Don’t assume setsockopt() actually changes it

- **Know how to get statistics from the stack**
  - E.g. netstat -s
Debugging Examples

- System that occasionally drops messages
  - Periodic glitches in the system
  - Recently added a logging capability
    - Due to paging of unbounded memory

- System that occasionally drops messages
  - Seems to happen during the nightly runs
  - Multipurpose operating system
    - Nightly tasks – heavy disk I/O

- The system crashes every 12ish days
  - Stack trace shows middleware freeing a null pointer
    - Setting a “large” timeout value…
Debugging Examples

- Dropping large chunks of messages occasionally
  - Using default OS stack parameters
  - Packet trace shows many bursts or ARP messages
    - ARP table being flushed

- Occasionally get garbled data
  - Occurs when changing packet size and rate
    - QNX IP stack building malformed packets

- Can’t scale the system past 50 nodes
  - New nodes added dynamically
  - Nodes share their config data with each other
    - But in this case, nodes resent all config data to everyone
Steps towards Distributed System Tools

1. Figure out the information needed
2. How to present it
3. How to collect it

- Introduce temporal events and randomness in automated testing
- Live visualization
  - Integrated across all the monitored parameters
- Interface abuse testing
- Design tools addressing complexity (QoS, etc.)
A Process for Debugging…

Getting the right data?
Integration Problem?
Network Problem?
Node Problem?
Data Interpretation Problem?
App Logic Problem?

Doing the right thing?

UML, documentation?
Instrumentation?