Real-Time Middleware

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…

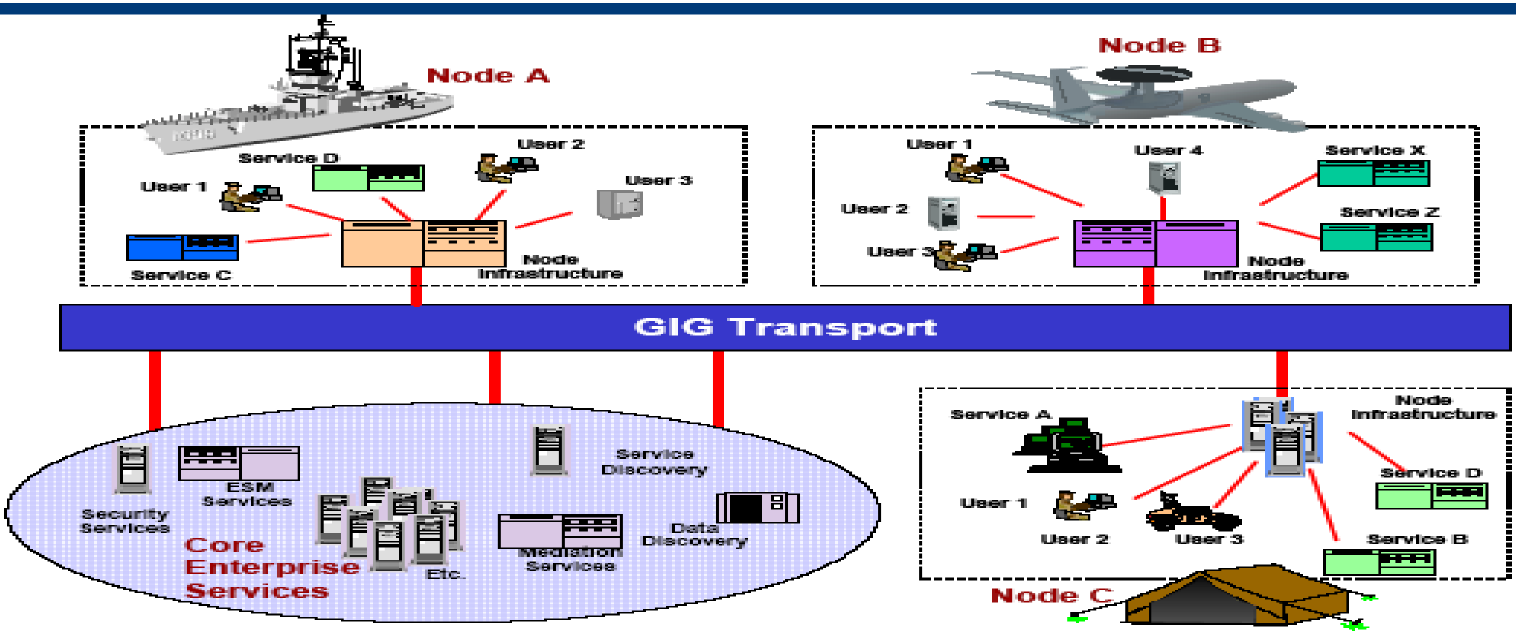
- Tools and possible tools…

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…



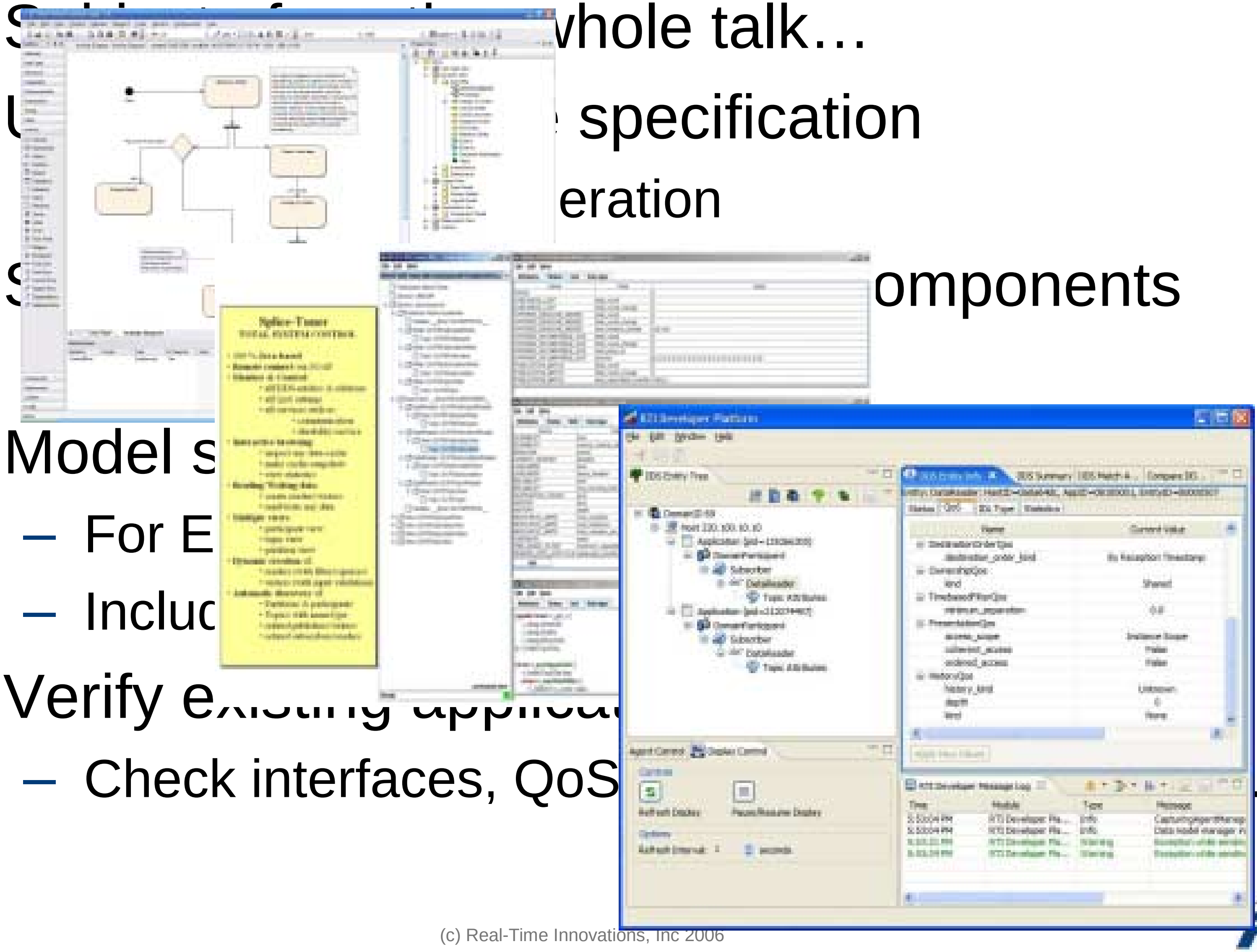
- Unique Distributed System Challenges
  - Things are getting more distributed
  - Soft failures are much more common in these systems
    - Traditional debugging techniques don't easily apply
  - Don't readily know what everything is really doing
  - Systems live well beyond the scope of their original 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…

- S whole talk…
- U specification
  eration
- S omponents
- Model s
  - For E
  - Incluc
- Verify e g applica
  - Check interfaces, QoS …

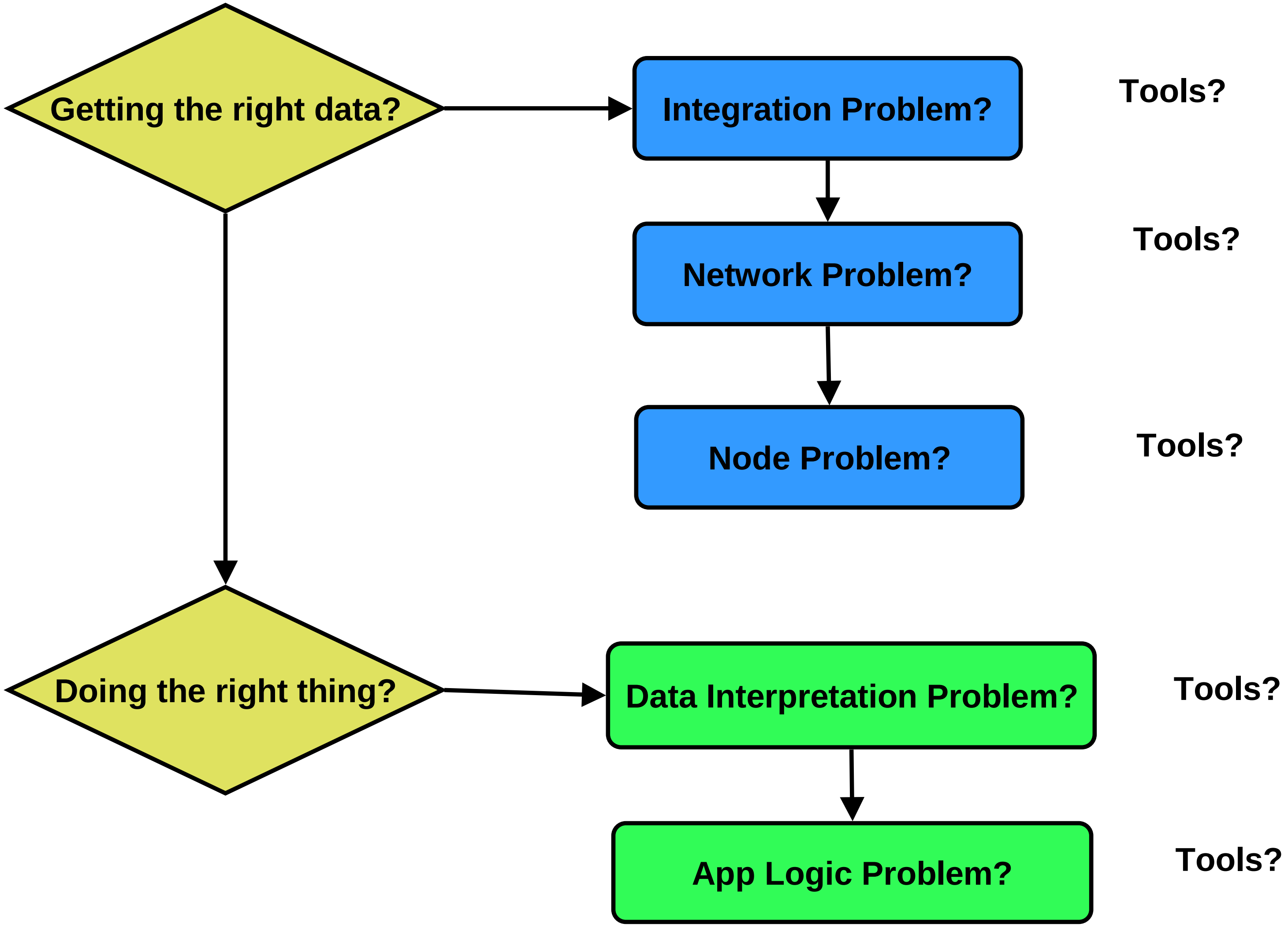


Stress and Random Event Debugging

Requires a Systematic Debugging Process

- Series of “Questions” and Tests
  - Drill drown 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…



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

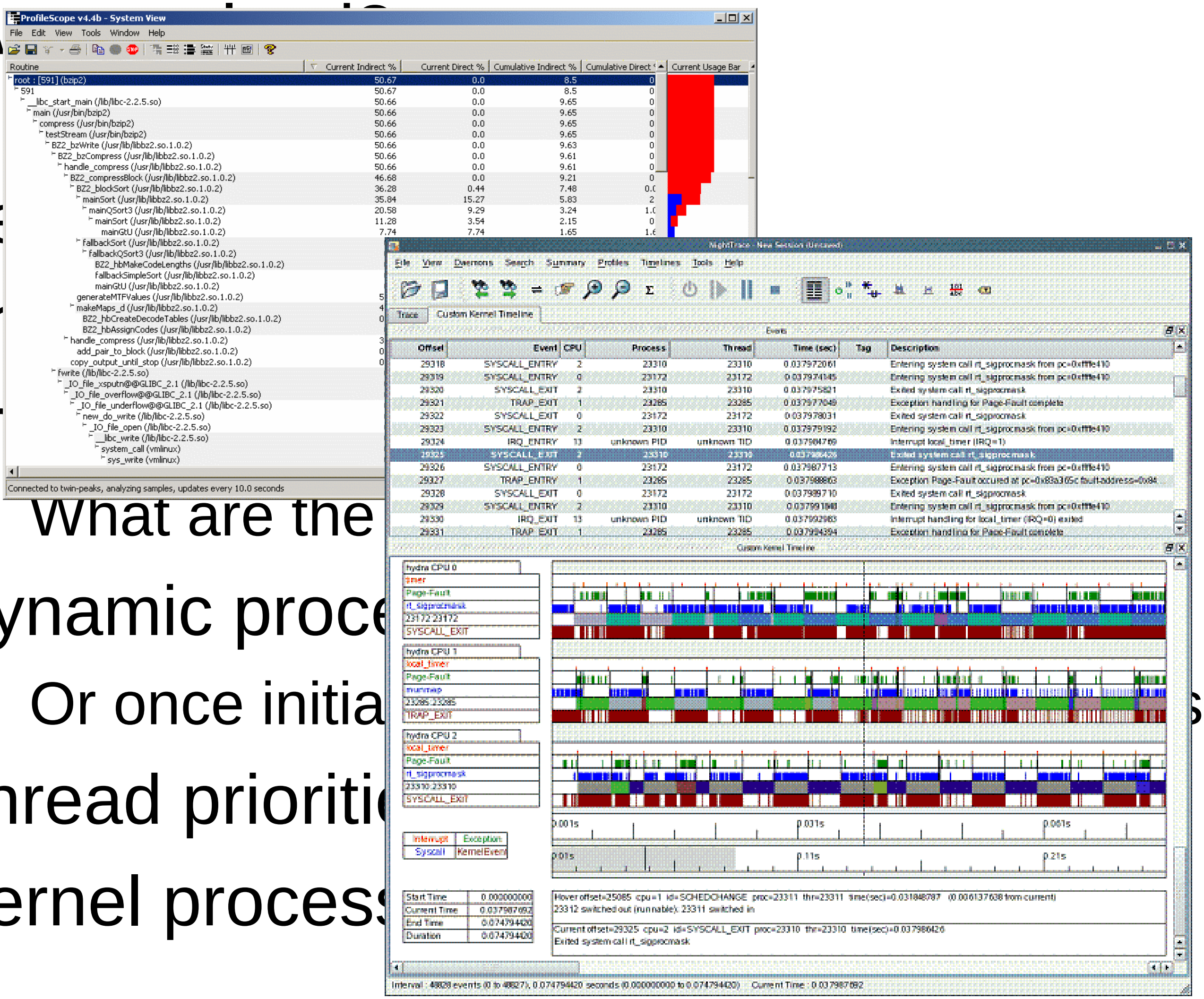
- Java
  - Considering garbage collection processing?
  - JDK compatibilities?
- C/C++
  - Recompiling all files after header/object changes?
  - STL usage? Implementation differences?
- Dynamic libraries
  - Are you loading the right one? Getting old version?

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

- Av
  –
- Pe
- Pr
- Tr
  – What are the
- Dynamic proce
  – Or once initia static?
- Thread prioriti
- Kernel process



Routine	Current Indirect %	Current Direct %	Cumulative Indirect %	Cumulative Direct
root : [591] (bzip2)	50.67	0.0	8.5	0
591	50.67	0.0	8.5	0
__libc_start_main (/lib/libc-2.2.5.so)	50.66	0.0	9.65	0
main (/usr/bin/bzip2)	50.66	0.0	9.65	0
compress (/usr/bin/bzip2)	50.66	0.0	9.65	0
testStream (/usr/bin/bzip2)	50.66	0.0	9.65	0
BZ2_bzWrite (/usr/lib/libbz2.so.1.0.2)	50.66	0.0	9.63	0
BZ2_bzCompress (/usr/lib/libbz2.so.1.0.2)	50.66	0.0	9.61	0
handle_compress (/usr/lib/libbz2.so.1.0.2)	50.66	0.0	9.61	0
BZ2_compressBlock (/usr/lib/libbz2.so.1.0.2)	46.68	0.0	9.21	0
BZ2_blockSort (/usr/lib/libbz2.so.1.0.2)	36.28	0.44	7.48	0.0
mainSort (/usr/lib/libbz2.so.1.0.2)	35.84	15.27	5.83	2
mainQSort3 (/usr/lib/libbz2.so.1.0.2)	20.58	9.29	3.24	1.0
mainSort (/usr/lib/libbz2.so.1.0.2)	11.28	3.54	2.15	0
mainGtU (/usr/lib/libbz2.so.1.0.2)	7.74	7.74	1.65	1.6
fallbackSort (/usr/lib/libbz2.so.1.0.2)				
fallbackQSort3 (/usr/lib/libbz2.so.1.0.2)				
BZ2_hbMakeCodeLengths (/usr/lib/libbz2.so.1.0.2)				
fallbackSimpleSort (/usr/lib/libbz2.so.1.0.2)				
mainGtU (/usr/lib/libbz2.so.1.0.2)				
generateMTFValues (/usr/lib/libbz2.so.1.0.2)	5			
makeMaps_d (/usr/lib/libbz2.so.1.0.2)	4			
BZ2_hbCreateDecodeTables (/usr/lib/libbz2.so.1.0.2)	0			
BZ2_hbAssignCodes (/usr/lib/libbz2.so.1.0.2)				
handle_compress (/usr/lib/libbz2.so.1.0.2)	3			
add_pair_to_block (/usr/lib/libbz2.so.1.0.2)	0			
copy_output_until_stop (/usr/lib/libbz2.so.1.0.2)	0			
fwrite (/lib/libc-2.2.5.so)				
_IO_file_xsputn@@GLIBC_2.1 (/lib/libc-2.2.5.so)				
_IO_file_overflow@@GLIBC_2.1 (/lib/libc-2.2.5.so)				
_IO_file_underflow@@GLIBC_2.1 (/lib/libc-2.2.5.so)				
new_do_write (/lib/libc-2.2.5.so)				
_IO_file_open (/lib/libc-2.2.5.so)				
__libc_write (/lib/libc-2.2.5.so)				
system_call (vmlinux)				
sys_write (vmlinux)				

Connected to twin-peaks, analyzing samples, updates every 10.0 seconds

Offset	Event	CPU	Process	Thread	Time (sec)	Tag	Description
29318	SYSCALL_ENTRY	2	23310	23310	0.037972061		Entering system call rt_sigprocmask from pc=0xffffe410
29319	SYSCALL_ENTRY	0	23172	23172	0.037974145		Entering system call rt_sigprocmask from pc=0xffffe410
29320	SYSCALL_EXIT	2	23310	23310	0.037975821		Exited system call rt_sigprocmask
29321	TRAP_EXIT	1	23285	23285	0.037977049		Exception handling for Page-Fault complete
29322	SYSCALL_EXIT	0	23172	23172	0.037978031		Exited system call rt_sigprocmask
29323	SYSCALL_ENTRY	2	23310	23310	0.037979192		Entering system call rt_sigprocmask from pc=0xffffe410
29324	IRQ_ENTRY	13	unknown PID	unknown TID	0.037984769		Interrupt local_timer (IRQ=1)
29325	SYSCALL_EXIT	2	23310	23310	0.037986426		Exited system call rt_sigprocmask
29326	SYSCALL_ENTRY	0	23172	23172	0.037987713		Entering system call rt_sigprocmask from pc=0xffffe410
29327	TRAP_ENTRY	1	23285	23285	0.037988863		Exception Page-Fault occured at pc=0x83a365c fault-address=0x84
29328	SYSCALL_EXIT	0	23172	23172	0.037989710		Exited system call rt_sigprocmask
29329	SYSCALL_ENTRY	2	23310	23310	0.037991840		Entering system call rt_sigprocmask from pc=0xffffe410
29330	IRQ_EXIT	13	unknown PID	unknown TID	0.037992983		Interrupt handling for local_timer (IRQ=0) exited
29331	TRAP_EXIT	1	23285	23285	0.037994394		Exception handling for Page-Fault complete

Start Time	0.000000000
Current Time	0.037987692
End Time	0.074794420
Duration	0.074794420

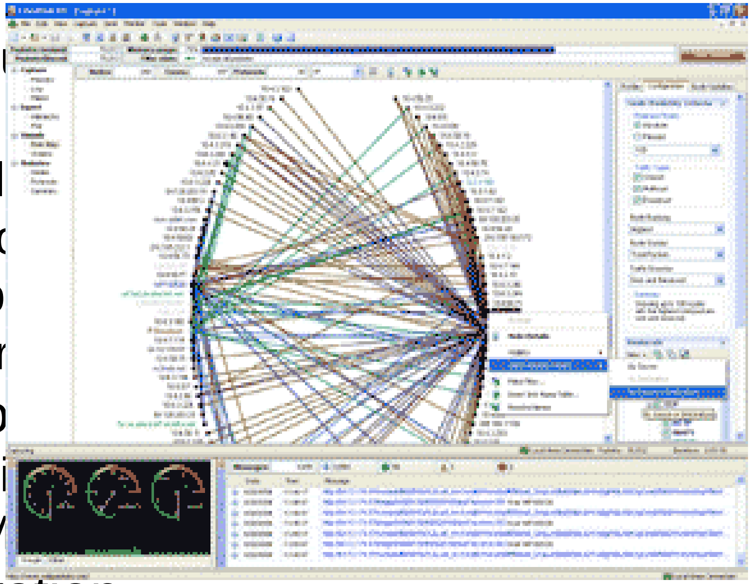
Hover offset=25085 cpu=1 id=SCHEDCHANGE proc=23311 thr=23311 time(sec)=0.031848787 (0.006137638 from current)
23312 switched out (runnable); 23311 switched in

Current offset=29325 cpu=2 id=SYSCALL_EXIT proc=23310 thr=23310 time(sec)=0.037986426
Exited system call rt_sigprocmask

Interval : 48828 events (0 to 48827), 0.074794420 seconds (0.000000000 to 0.074794420) Current Time : 0.037987692

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 ro
- GigE?
  - Half or full dupl
  - Is everything co
  - Using Jumbo p
  - Are Ethernet dr
- Other Ethernet co
  - Collision domai
  - What are the M
- 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

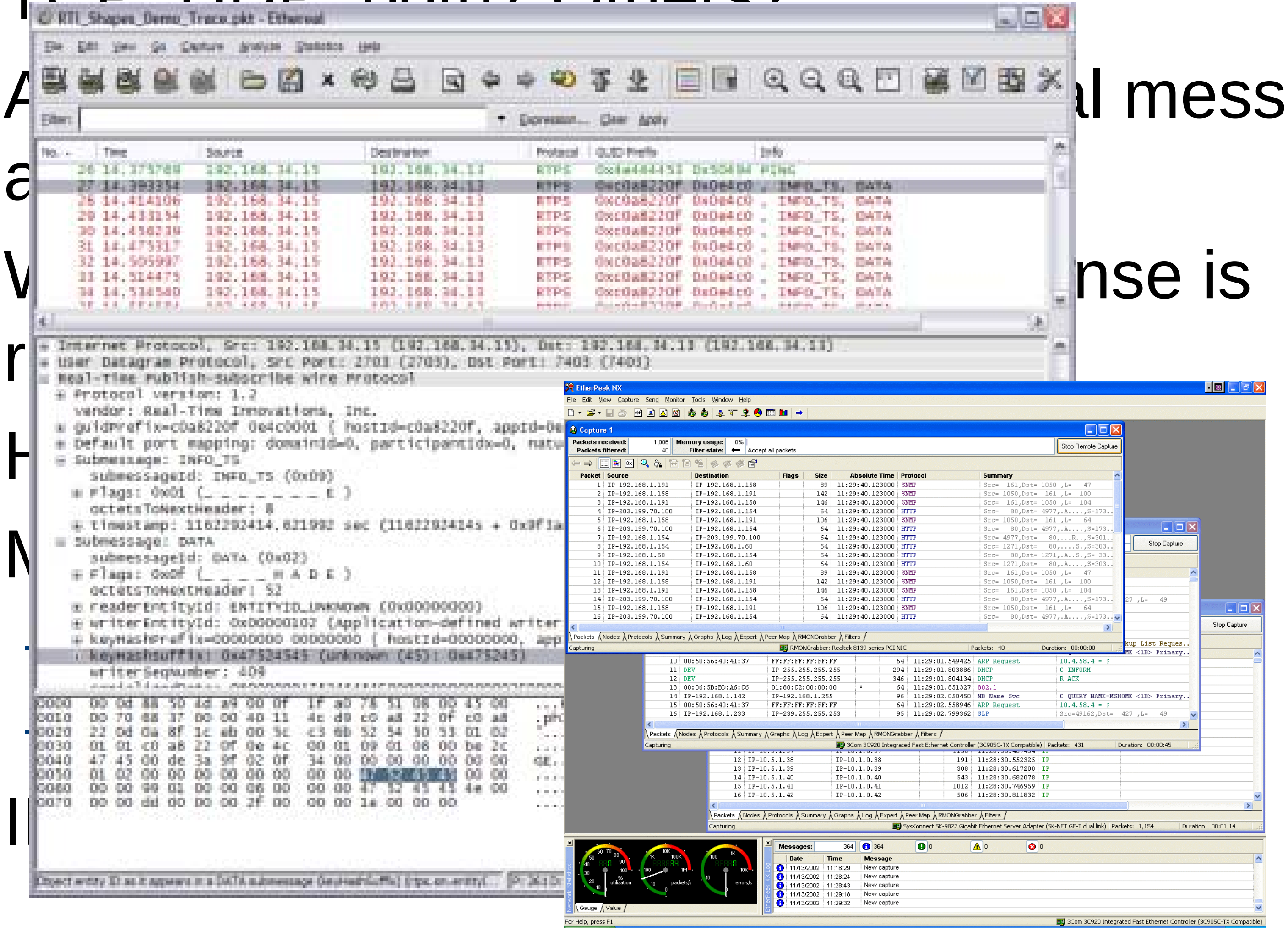
- What thread(s) or process(es) handle incoming data?
  - Who owns the threads?
  - Single or multiple threads?
  - Single thread per socket? Per Port?
  - Dedicated threads for servicing data?
  - Threads blocked on ports waiting for data?
  - Which thread is responsible for sending acknowledgements?
  - 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?
- A … al messages a…
- W … nse is r…
- H…
- M…
  - …
  - …
- I…


RTI_Shapes_Demo_Trace.pkt - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time       Source         Destination    Protocol GUID Prefix           Info
    26 14.375788  192.168.34.15  192.168.34.13  RTPS     0x4a444453 0x5048d PING
    27 14.393354  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    28 14.414106  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    29 14.433154  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    30 14.456238  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    31 14.475317  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    32 14.505997  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    33 14.514475  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
    34 14.534540  192.168.34.15  192.168.34.13  RTPS     0xc0a8220f 0x0e4c0 , INFO_TS, DATA
```

```
+ Internet Protocol, Src: 192.168.34.15 (192.168.34.15), Dst: 192.168.34.13 (192.168.34.13)
+ User Datagram Protocol, Src Port: 2703 (2703), Dst Port: 7403 (7403)
- Real-Time Publish-Subscribe Wire Protocol
  + Protocol version: 1.2
    vendor: Real-Time Innovations, Inc.
  + guidPrefix=c0a8220f 0e4c0001 { hostId=c0a8220f, appId=0e
  + Default port mapping: domainId=0, participantIdx=0, natu
  - Submessage: INFO_TS
      submessageId: INFO_TS (0x09)
    + Flags: 0x01 (_ _ _ _ _ _ _ E )
      octetsToNextHeader: 8
    + timestamp: 1162292414.621992 sec (1162292414s + 0x9f1a
  - Submessage: DATA
      submessageId: DATA (0x02)
    + Flags: 0x0f (_ _ _ _ H A D E )
      octetsToNextHeader: 52
    + readerEntityId: ENTITYID_UNKNOWN (0x00000000)
    + writerEntityId: 0x00000102 (Application-defined writer
    + keyHashPrefix=00000000 00000000 { hostId=00000000, app
    + keyHashSuffix: 0x47524545 (unknown (45): 0x475245)
      writerSeqNumber: 409
```

```
0000  00 0d 88 50 4d a4 00 0f  1f a0 78 51 08 00 45 00
0010  00 70 68 37 00 00 40 11  4c d0 c0 a8 22 0f c0 a8
0020  22 0d 0a 6f 1c eb 00 5c  c3 6b 52 54 50 53 01 02
0030  01 01 c0 a8 22 0f 0e 4c  00 01 09 01 08 00 be 2c
0040  47 45 00 de 3a 9f 02 0f  34 00 00 00 00 00 00 00
0050  01 02 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 01 00 00 06 00  00 00 47 52 45 45 4e 00
0070  00 00 dd 00 00 00 2f 00  00 00 1a 00 00 00
```

Object entity ID as it appears in a DATA submessage (keyHashSuffix) (rtps.sn.entity)

Packets received: 1,006 Memory usage: 0%
Packets filtered: 40 Filter state: ← Accept all packets

```
Packet Source             Destination         Flags Size Absolute Time   Protocol Summary
     1 IP-192.168.1.191   IP-192.168.1.158           89 11:29:40.123000 SNMP     Src=  161,Dst= 1050 ,L=   47
     2 IP-192.168.1.158   IP-192.168.1.191          142 11:29:40.123000 SNMP     Src= 1050,Dst=  161 ,L=  100
     3 IP-192.168.1.191   IP-192.168.1.158          146 11:29:40.123000 SNMP     Src=  161,Dst= 1050 ,L=  104
     4 IP-203.199.70.100  IP-192.168.1.154           64 11:29:40.123000 HTTP     Src=   80,Dst= 4977,.A....,S=173..
     5 IP-192.168.1.158   IP-192.168.1.191          106 11:29:40.123000 SNMP     Src= 1050,Dst=  161 ,L=   64
     6 IP-203.199.70.100  IP-192.168.1.154           64 11:29:40.123000 HTTP     Src=   80,Dst= 4977,.A....,S=173..
     7 IP-192.168.1.154   IP-203.199.70.100          64 11:29:40.123000 HTTP     Src= 4977,Dst=   80,...R..,S=301..
     8 IP-192.168.1.154   IP-192.168.1.60            64 11:29:40.123000 HTTP     Src= 1271,Dst=   80,....S.,S=303..
     9 IP-192.168.1.60    IP-192.168.1.154           64 11:29:40.123000 HTTP     Src=   80,Dst= 1271,.A..S.,S= 33..
    10 IP-192.168.1.154   IP-192.168.1.60            64 11:29:40.123000 HTTP     Src= 1271,Dst=   80,.A....,S=303..
    11 IP-192.168.1.191   IP-192.168.1.158           89 11:29:40.123000 SNMP     Src=  161,Dst= 1050 ,L=   47
    12 IP-192.168.1.158   IP-192.168.1.191          142 11:29:40.123000 SNMP     Src= 1050,Dst=  161 ,L=  100
    13 IP-192.168.1.191   IP-192.168.1.158          146 11:29:40.123000 SNMP     Src=  161,Dst= 1050 ,L=  104
    14 IP-203.199.70.100  IP-192.168.1.154           64 11:29:40.123000 HTTP     Src=   80,Dst= 4977,.A....,S=173..
    15 IP-192.168.1.158   IP-192.168.1.191          106 11:29:40.123000 SNMP     Src= 1050,Dst=  161 ,L=   64
    16 IP-203.199.70.100  IP-192.168.1.154           64 11:29:40.123000 HTTP     Src=   80,Dst= 4977,.A....,S=173..
```

Packets Nodes Protocols Summary Graphs Log Expert Peer Map RMONGrabber Filters

Capturing RMONGrabber: Realtek 8139-series PCI NIC Packets: 40 Duration: 00:00:00

```
    10 00:50:56:40:41:37  FF:FF:FF:FF:FF:FF          64 11:29:01.549425 ARP Request  10.4.58.4 = ?
    11 DEV                IP-255.255.255.255        294 11:29:01.803886 DHCP         C INFORM
    12 DEV                IP-255.255.255.255        346 11:29:01.804134 DHCP         R ACK
    13 00:06:5B:BD:A6:C6  01:80:C2:00:00:00   *      64 11:29:01.851327 802.1
    14 IP-192.168.1.142   IP-192.168.1.255           96 11:29:02.050450 NB Name Svc  C QUERY NAME=MSHOME <1B> Primary..
    15 00:50:56:40:41:37  FF:FF:FF:FF:FF:FF          64 11:29:02.558946 ARP Request  10.4.58.4 = ?
    16 IP-192.168.1.233   IP-239.255.255.253         95 11:29:02.799362 SLP          Src=49162,Dst=  427 ,L=   49
```

Capturing 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible) Packets: 431 Duration: 00:00:45

```
    12 IP-10.5.1.38       IP-10.1.0.38              191 11:28:30.552325 IP
    13 IP-10.5.1.39       IP-10.1.0.39              308 11:28:30.617200 IP
    14 IP-10.5.1.40       IP-10.1.0.40              543 11:28:30.682078 IP
    15 IP-10.5.1.41       IP-10.1.0.41             1012 11:28:30.746959 IP
    16 IP-10.5.1.42       IP-10.1.0.42              506 11:28:30.811832 IP
```

Capturing SysKonnect SK-9822 Gigabit Ethernet Server Adapter (SK-NET GE-T dual link) Packets: 1,154 Duration: 00:01:14

Messages: 364 ⓘ 364 ● 0 ⚠ 0 ⊗ 0

Date	Time	Message
11/13/2002	11:18:29	New capture
11/13/2002	11:28:24	New capture
11/13/2002	11:28:43	New capture
11/13/2002	11:29:18	New capture
11/13/2002	11:29:32	New capture

For Help, press F1 — 3Com 3C920 Integrated Fast Ethernet Controller (3C905C-TX Compatible)

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?
- NTP? DAYTIME? TIME? HTP? ICMP?
- 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 chucks of messages occasionally
  - Using default OS stack parameters
  - Packet trace shows many bursts or ARP messages
    - ARP table being flushed

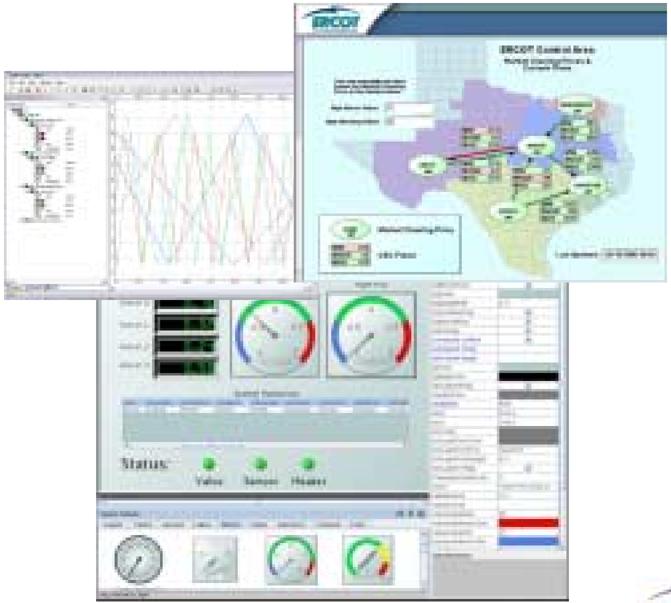
- Occasionally get garbled data
  - Occurs when changing packet size and rate
    - QNX IP stack building malformed packets

- 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

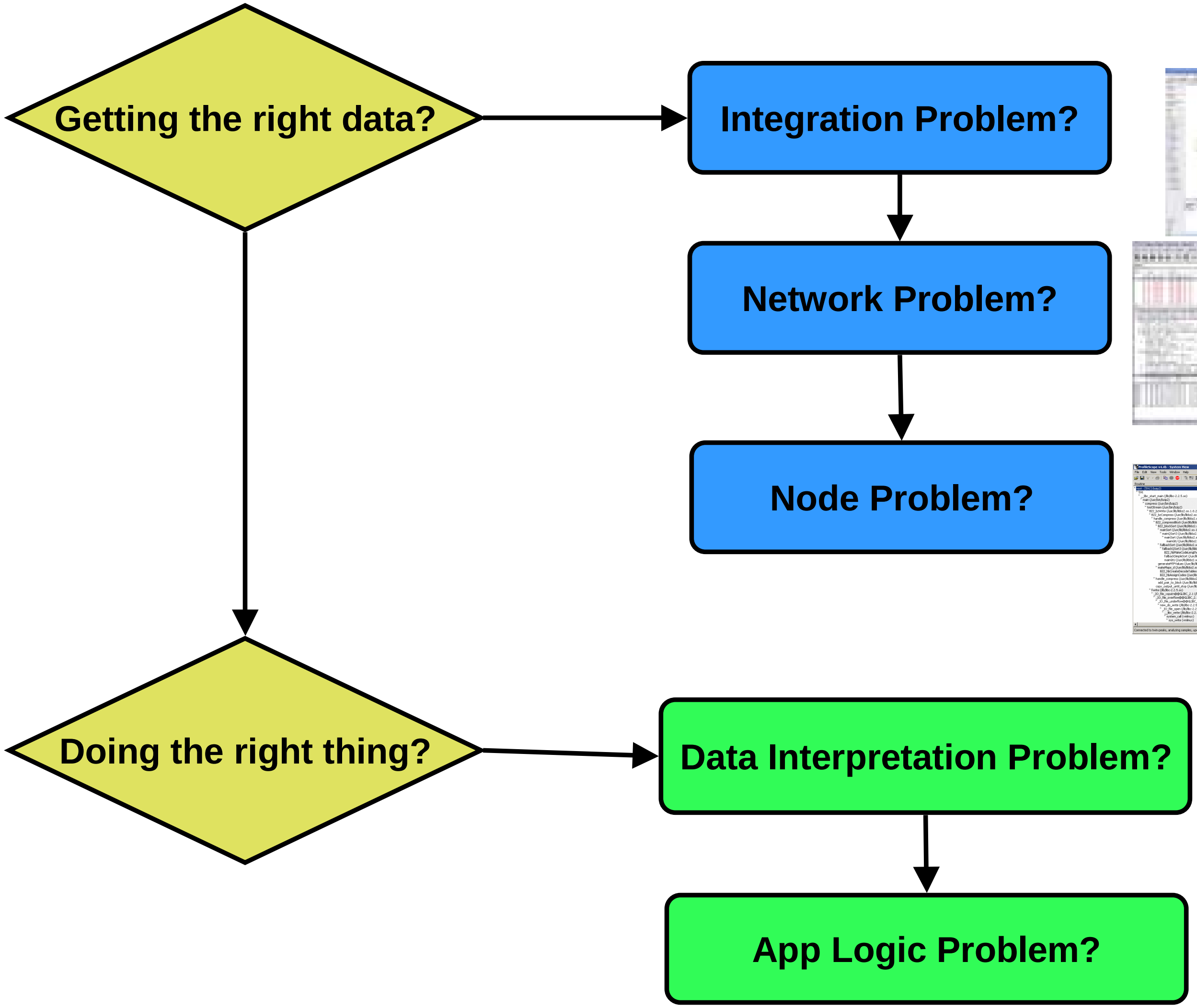
Steps towards Distributed System Tools

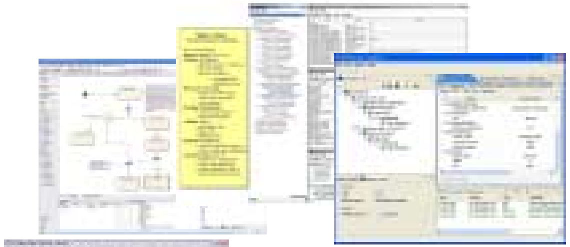
1. Figure out the information needed
2. How to present it
3. How to collect it

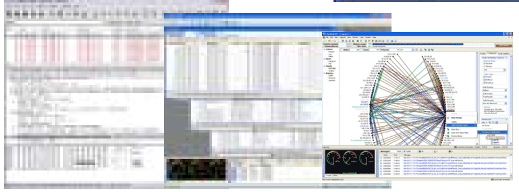
- Introduce temporal events and randomness in automated testing
- Live visualization
  - Integrated across all the monitored parameters
- Interface abuse testing
- Design tools addressing complexity (QoS, etc.)

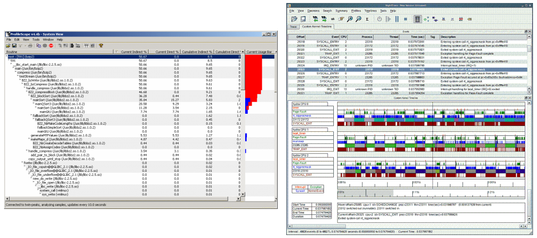


A Process for Debugging…









UML, documentation?

Instrumentation?