Rorschach Plots and Network Performance Analysis

Jim Gilsinn
Kenexis Consulting Corporation
Rorschach?
“Rorschach” Plots

Histogram Count

Test Time (s)
“Rorschach” Plots
“Rorschach” Plots
ICS Environment
ICS Environment
ICS Systems
What’s This All About?

• I used to work at NIST
• I left about a year ago
• I worked on ICS network performance metrics, tests, and tools
• The test tools I developed have been dormant since leaving
• The vendors I worked with while at NIST want to tool
• My new employer won’t support open-source development

• I’m here to beg for help!
Performance Testing Methodology: 
Performance Metrics

- Publish/subscribe or peer-to-peer communications
- Main performance metric: Cyclic frequency variability/jitter
- Real-time EtherNet/IP uses publish/subscribe
  - Requested/Accepted Packet Interval (RPI/API)
  - Measured Packet Interval (MPI)
Performance Testing Methodology: Performance Metrics

- Command/response or master/slave communications
- Main performance metric: Latency
- Large numbers of protocols use this
  - Most (All?) PC-based server/client protocols – HTTP(S), (S)FTP, etc.
  - Most industrial protocols – Modbus/TCP, Profinet, Ethercat, etc.
IENetP Test Tool

• Industrial Ethernet Network Performance (IENetP)
• Current Version = 1.1.2, Released 2011-02-11
• Software Features
  • Analyze existing Wireshark captures
  • Allows user to override default EtherNet/IP filter
  • Isolates individual traffic streams
  • Determine cyclic jitter of those streams
  • Generates HTML report
  • Generates time-space & histogram graphs
  • Graphs allow zooming
NIST Performance Test Tool

- Industrial Ethernet Network Performance (IENetP) Test Tool
- Factory Equipment Network Testing (FENT) Framework
FENT Features

• All Analysis Features From IENetP
  • Analyze Wireshark capture files
  • Build graphs and reports of results

• Added Features
  • True multi-protocol support
  • Real-time testing capability
  • Extensible framework
FENT Personality Modules

- Wrapper for Driver Application
  - Implement a TCP-socket interface for UCA-API messaging
  - Build a simple XML-based PM Descriptor file
- Features
  - Describes Wireshark parameters
  - Allows any protocol to be used
  - Can be built/loaded at run-time

October 19-20, 2013
BSidesDC 2013
1. **Testing Module → Protocol PM** – Grab protocol-specific Wireshark parameters via UCA-API
2. **Testing Module → Wireshark** – Start capturing traffic
3. **Testing Module → Protocol PM** – Command driver app to communicate with DUT
4. **Testing Module → Wireshark** – Stop capturing traffic, process capture file using desired protocol and user parameters, generate PSML file
5. **Analysis Engine** – Read PSML file, analyze packets for desired metrics
6. **Reporting Engine** – Report data to user
FENT UCA-API Schema
FENT Framework

• Project Home:
  • [http://sourceforge.net/projects/fent/](http://sourceforge.net/projects/fent/)

• What’s Available:
  • SVN repository & schema
  • FENT software
    • Conduct real-time testing
    • Analyze results
    • Build graphs on-screen
  • NIST SensorSim PM, IEEE 1451 PM
  • EtherNet/IP PlugFest “Gold Standard” Background Traffic
FENT Framework

- Known Problems & Issues
  - Doesn’t work with Wireshark 1.9+
    - Tshark argument for getting fields changed
  - Logic problems with using multi-protocol Wireshark headers
  - Software doesn’t use true database
  - Testing automation not integrated
  - No installer
FENT Demo
“Gold Standard” Background Traffic
“Gold Standard” Background Traffic

• What Is It?
  • A set of Wireshark captures, Linux scripts, and analysis results
  • Based on EtherNet/IP PlugFest performance testing requirements
  • High precision and accuracy Wireshark captures of PlugFest performance background traffic
  • Linux scripts designed for use in BackTrack Linux (http://www.backtrack-linux.org/)
  • Analysis results show validation for use in PlugFest performance testing

• Where Can You Get It?
  • http://ienetp.sourceforge.net/EtherNet-IP_Testing.zip or
  • FENT SVN in Background_Traffic folder
<table>
<thead>
<tr>
<th>Traffic Type</th>
<th>Rate (pps)</th>
<th>Baseline</th>
<th>Steady-State Managed</th>
<th>Steady-State Unmanaged</th>
<th>Burst Managed</th>
<th>Burst Unmanaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARP Request Broadcasts</td>
<td>180</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Gratuitous ARP Broadcasts</td>
<td>180</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>DHCP Request Broadcasts</td>
<td>100</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ICMP (ping) Request Broadcasts</td>
<td>100</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>NTP Multicasts</td>
<td>10</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>EtherNet/IP ListIdentity Request</td>
<td>10</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>EtherNet/IP Class 1</td>
<td>1800</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>ARP Burst Requests</td>
<td>240 pkts @ 4k Hz</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
“Gold Standard” Captures

• Built From Individual Traffic Streams
  • Each traffic stream generated and captured using NIST Ixia system (a few microseconds jitter)
  • Assembled using editcap and mergecap scripts
  • Final captures are 60-seconds long
    • Can’t just loop continuously
    • Longer test captures require rebuilding (not hard)

• Analyzed Using IENetP
  • Analysis results are included in package
  • Well within spec for PlugFest performance testing needs (<25% of desired packet intervals)
Licensing?

- The project is Public Domain!!!
- There are NO LICENSING ISSUES!!!
What’s Next?

• Contact Me
  • Jim Gilsinn
  • 301-706-9985
  • jim.gilsinn@kenexis.com
  • Twitter – @JimGilsinn
  • LinkedIn – http://www.linkedin.com/in/jimgilsinn/

• Review the FENT SourceForge Project
  • http://sourceforge.net/projects/fent/

• Fork the Project