Sherlock – Diagnosing Problems in the Enterprise

Srikanth Kandula
Victor Bahl, Ranveer Chandra, Albert Greenberg, David Maltz, Ming Zhang

Enterprise Management: Between a Rock and a Hard Place

**Manageability**
- Stick with tried software, never change infrastructure
- Cheap
- Upgrades are hard, forget about innovation!

**Usability**
- Keep pace with technology
- Expensive
  - IT staff in 1000s
  - 72% of MS IT budget is staff
- Reliability Issues
  - Cost of down-time
Well-Managed Enterprises Still Unreliable

Response time of a Web server (ms)

Fraction Of Requests

85% Normal

10% Troubled

0.7% Down

10% responses take up to 10x longer than normal

How do we manage evolving enterprise networks?

Well-Managed Enterprises Still Unreliable

Response time of a Web server (ms)

Fraction Of Requests

85% Normal

10% Troubled

0.7% Down

How do we manage evolving enterprise networks?
Current Tools Miss the Forest for the Trees

- Monitor Individual Boxes or Protocols
- Flood admin with alerts
- Don’t convey the end-to-end picture

But, the primary goal of enterprise management is to diagnose user-perceived problems!

Sherlock

Instead of looking at the nitty-gritty of individual components, use an end-to-end approach that focuses on user problems
Challenges for the End-to-End Approach

• Don’t know what user’s performance depends on
  – Dependencies are distributed
  – Dependencies are non-deterministic

• Don’t know which dependency is causing the problem
  – Server CPU 70%, link dropped 10 packets, but which affected user?
**Sherlock’s Contributions**

- Passively infers dependencies from logs
- Builds a unified dependency graph incorporating network, server and application dependencies
- Diagnoses user problems in the enterprise
- Deployed in a part of the Microsoft Enterprise

**Sherlock’s Architecture**
Sherlock’s Architecture

Network Dependency Graph

User Observations

Inference Engine

Sherlock works for various client-server applications

30ms 1000ms Timeout

User Observations

List of Suspicious Components

Network Dependency Graph

Inference Engine

Sherlock works for various client-server applications
How do you automatically **learn such distributed dependencies**?

**Strawman**: Instrument all applications and libraries  
→ **Not Practical**

**Sherlock** exploits timing info

If talks to B, whenever talks to C ‡  Dependent Connections
Strawman: Instrument all applications and libraries → Not Practical

Sherlock exploits timing info

If talks to B, whenever talks to C → Dependent Connections

\[ \Delta t \]

False Dependence

**Inter-access time**

Dependent iff \( \Delta t \ll \text{Inter-access time} \)

If talks to B, whenever talks to C → Dependent Connections

As long as this occurs with probability higher than chance
Sherlock’s **Algorithm** to Infer Dependencies

- Infer dependent connections from **timing**
- Infer topology from **Traceroutes & configurations**

- **Works with legacy applications**
- **Adapts to changing conditions**
But hard dependencies are not enough…

‰ Need Probabilities

Sherlock uses the frequency with which a dependence occurs in logs as its edge probability

If Bill caches server’s IP ‰ DNS down but Bill gets video

p1=10% p2=100%
How do we use the dependency graph to diagnose user problems?

Diagnosing User Problems

Which components caused the problem?
Need to disambiguate!!
Diagnosing User Problems

- Disambiguate by correlating
  - Across logs from same client
  - Across clients
- Prefer simpler explanations

Will Correlation Scale?
Microsoft Internal Network
- O(100,000) client desktops
- O(10,000) servers
- O(10,000) apps/services
- O(10,000) network devices

Dependency Graph is Huge

Will Correlation Scale?

Can we evaluate all combinations of component failures?

The number of fault combinations is exponential!

Impossible to compute!
Scalable Algorithm to Correlate

Only a few faults happen concurrently
But how many is few?

Evaluate enough to cover 99.9% of faults

For MS network, at most 2 concurrent faults ‡ 99.9% accurate

Exponential ‡ Polynomial

Only a few faults happen concurrently
But how many is few?

Evaluate enough to cover 99.9% of faults

For MS network, at most 2 concurrent faults ‡ 99.9% accurate

Exponential ‡ Polynomial
Scalable Algorithm to Correlate

Only a few faults happen concurrently
But how many is few?
Evaluate enough to cover 99.9% of faults
For MS network, at most 2 concurrent faults ‡ 99.9% accurate

Exponential ‡ Polynomial

Only few nodes change state
Re-evaluate only if an ancestor changes state

Reduces the cost of evaluating a case by 30x-70x

Results
Experimental Setup

• Evaluated on the Microsoft enterprise network
• Monitored 23 clients, 40 production servers for 3 weeks
  – Clients are at MSR Redmond
  – Extra host on server’s Ethernet logs packets
• Busy, operational network
  – Main Intranet Web site and software distribution file server
  – Load-balancing front-ends
  – Many paths to the data-center

What Do Web Dependencies in the MS Enterprise Look Like?
What Do Web Dependencies in the MS Enterprise Look Like?

Client Accesses Portal

What Do Web Dependencies in the MS Enterprise Look Like?

Client Accesses Portal
What Do Web Dependencies in the MS Enterprise Look Like?

Sherlock discovers complex dependencies of real apps.

What Do File-Server Dependencies Look Like?

Sherlock works for many client-server applications.
Sherlock Identifies Causes of Poor Performance

Dependency Graph: 2565 nodes; 358 components that can fail

87% of problems localized to 16 components

Sherlock Identifies Causes of Poor Performance

Inference Graph: 2565 nodes; 358 components that can fail

Corroborated the three significant faults
• SNMP-reported utilization on a link flagged by Sherlock
• Problems coincide with spikes

Sherlock identifies the troubled link but SNMP cannot!

Comparing with Alternatives

• Dataset of known (fault, observations) pairs
• Accuracy = 1 – (Prob. False Positives + Prob. False Negatives)
Comparing with Alternatives

- Dataset of known (fault, observations) pairs
- Accuracy = 1 – (Prob. False Positives + Prob. False Negatives)

<table>
<thead>
<tr>
<th>Method</th>
<th>Probability Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shrink (probabilistic)</td>
<td>59%</td>
<td></td>
</tr>
<tr>
<td>SCORE (non-probabilistic)</td>
<td>53%</td>
<td></td>
</tr>
</tbody>
</table>

Sherlock outperforms existing tools!
Time to Localize Faults

- From Inference Graph, create scaled versions

Time to Localize faults is roughly linear!

Experiments

Sensitivity to Errors in the Inference Graph

- Perturb Inference Graph in Several ways
- Robust up to 10% error in dependencies
  ⇒ Can run off auto-generated graphs
Conclusions

• Sherlock **passively infers** network-wide **dependencies** from logs and traceroutes

• It **diagnoses faults by correlating** user observations

• It works at **scale!**

• Experiments in Microsoft’s Network show
  – **Finds faults missed by existing tools** like SNMP
  – **Is more accurate** than prior techniques

• Steps towards a Microsoft product
Ongoing Work

- Helping MS product groups that maintain business apps like payroll
- Integrating automated dependency inference into MS product MOM

Sherlock System
Challenge 1: At what level of abstraction should dependencies be modeled so as to be feasibly automated

Root causes

User Experience
The Inference Graph

Root causes

Root Causes –
§ Computer (IP address)
§ Service (IP address, port pair)
§ Routers, Links

Meta-nodes

User Experience

Challenge 2: Non-deterministic dependencies

Model allows significant re-use of nodes
The Inference Graph

**Root causes**
- Computer (IP address)
- Service (IP address, port pair)
- Routers, Links

**Meta-nodes**

**User Experience**

**Challenge 4:** How to find dependent connections without deep packet inspection?

- Client request depends on DNS
- Server response depends on SQL backend

**Idea**
If host A talks to host B, whenever it talks to host C, then the two connections are dependent

**Algorithm**
Given a pair A—C, find all other servers that are accessed by A
Experimental Setup

- Monitored 40 production servers for 3 weeks
- Intranet Portal and Software Dist. File-server
- Load-balancing and ECMP