Architectural Support For Network Troubleshooting

Scott Shenker, Mark Allman, Vern Paxson, Christian Kreibich, and Nicholas Weaver
International Computer Science Institute & UC Berkeley

Architecting for Troubleshooting: Principles

Responsibility:
• Who has jurisdiction for solving a problem?
• Enable users to generate credible and actionable problem reports
  ⇒ sufficient evidence to unambiguously demonstrate problem
• Responsible party likely in the best position to perform detailed diagnostics

Beyond Modularity:
• Fundamental tension between modular design and troubleshooting
• Interface narrowness allows for separation of concerns and rapid innovation …
• … but interface narrowness tends to mask problems, as errors and exceptional conditions must propagate across layers of abstraction in some meaningful form

Tracking Causality:
• Network events entail lengthy sequences of activity dependent upon / affected by previous activity
• Determining chain of events that lead up to failures enables separating symptoms from root causes

Enriched Logging:
• Annotations associate meta-data with network activity
• Logging requires distillation into more abstract forms over time
• Logging requires dialog between components generating log entries and the logging infrastructure
  ⇒ callbacks support distillation and interactive debugging

Privacy:
• Information that facilitates debuggability can also facilitate detailed tracking of user activity
• We need mechanisms that, when possible, decouple logs of user activity from user identities
• Must recognize tussle between tracking activities for operational purposes versus masking it for reasons of privacy
• Problem even harder since often information needs to cross organizational boundaries
• Requests for information should include provenance attesting to the requester’s right-of-access:
  ⇒ E.g., demonstrate knowledge of related details or nonces known only to the traffic participants

Troubleshooting and Robustness:
• Troubleshooting and robustness are deeply intertwined
• Better troubleshooting can lead to automatic diagnosis and mitigation…
• … Which in turn can lead to masking problems
  ⇒ As can any robustness mechanism coupled with a narrow interface

Architecture for Troubleshooting: Preliminary Mechanisms

VAST:
Visibility Across Time and Space
Interactive repository of event level descriptions of network activity
• Implemented using “FastBit” database technology
Supports programmatic querying, aging, distillation, aggregation, and expiration
Designed to support cross-organizational data sharing
Queries for past activity can be mirrored into proactive monitoring for future activity

Network Radar
What are the elements along a network path?
How do they appear from different vantage points?
• Many elements not naturally exposed by standard operations
One technique: measure transformations to known content

X-Trace
• Pervasive Network Tracing Framework
• Architectural support for annotations

X-Trace

Reactive Measurement
Observations trigger measurements in response
Observations can come from:
User reports
Passive analysis
Proactive active probing
Changes measurement from an event to a process
Combine disparate measurement techniques by using the results of one measurement to drive additional assessments
For troubleshooting we can winnow possible root causes by using context-sensitive diagnostics

NSF Future Internet Design Meeting
November. 27-28th 2007
Washington, DC