A Scalable Approach for the Secure and Authorized Tracking of the Availability of Entities in Distributed Systems

Shrideep Pallickara, Jaliya Ekanayake & Geoffrey Fox
Community Grids Lab
Indiana University
Motivation

- Proliferation of distributed systems
- Interactions predicated on knowledge of the availability of entities
  - Control messages, protocol handshakes, actions and data interchange
  - Track resources at all times
    - Load, usage patterns etc
- Remedial actions in response to failures
  - Failure suspicions, failures, shutdown
Definition of Terms

- **Entity**
  - Resource, service, instruments, clients etc

- **Tracing**
  - Process of probing, and becoming aware of, the availability of an entity

- **Traced Entity**
  - The entity whose availability is being probed

- **Trackers**
  - Entities initiating availability probes for a traced entity

- **Traces**
  - Messages encapsulating the state of a traced entity
Push or Pull

• Push
  ▪ Traced entities push traces to the trackers at regular intervals
  ▪ Receipt of such pushed traces provides info on availability

• Pull
  ▪ Trackers poll the availability at regular intervals
  ▪ Decisions based on the outcome of the poll
Issues in developing solution

• In simple scheme every entity issues a trace every second
  ▪ Does not scale well
  ▪ With increasing scale every entity is inundated with traces

• Entities operate in myriad domains
  ▪ May deploy different transports for communications
Desired Features

- Reduction of complexity
  - Reduce the number of entities that a given entity needs to communicate with
    - Push/pull requires some entity to communicate with a rather large set

- Transport independent

- Selectivity in traces
  - For e.g. register ONLY to receive change notification traces

- Authorization
  - Restrict who is allowed to be part of the tracing process

- Security
  - Make the trace itself confidential

- Cope with some classes of DDoS attacks
Publish/Subscribe Systems

Middleware

CONSUMER

PRODUCER
NaradaBrokering

• Provisions an enabling infrastructure for building distributed systems
  ▪ Has services, and makes these services accessible to entities through simple invocations.
    • Processing at the service, and the infrastructure may be complicated, but this is shielded from the clients
• Dissemination is based on the pub/sub paradigm
• Open-source project http://www.naradabrokering.org
• Deployed in a wide variety of domains
  ▪ GIS, Audio/Video conferencing, collaboration, Geoscience and Physics
• 1425 classes, 157 packages and 300,000 lines of code
The NaradaBrokering Substrate

Broker Cloud

Gateway to Services
Service multiple clients
Organize Subscriptions
Route Messages
Enforce constraints
Topic Discovery Scheme

- Create topics that are unique in space and time in a decentralized fashion
- Establish topic provenance
  - Deterministic cryptographic verification of ownership
- Restrict discovery of topics
  - Possess valid credentials
  - Specified ACL
- Establish topic life-cycle
- Manage topic collections & organization
Our scheme

- Leverage pub/sub for disseminating traces
- Facilitate **selectivity** in trace consumption by trackers
  - Different types of Traces are issued over different topics.
- Restrictions on **authorizations** related to topics over which traces are issued
  - Discovery of these topics
  - Actions associated with these topics
- Traces demonstrate authorization, tamper-evidence and source
  - Unambiguously verify this
- **Secure** the distribution of traces
Trace Topic

- Topic related to trace information related to a given entity
  - Derivative topics are constructed from this
- At creation time, the traced entity must specify who is authorized to trace it
  - ACL or based on credentials
- Register a descriptor that will facilitate discovery
  - Availability/Traces/Entity-ID
Constrained Topics

• Systems Topics
  • Derivative topics managed by the broker
    ▪ Multiple derived topics facilitates trace selectivity
  
• Constraints are based on
  ▪ Limits on performed actions
  ▪ Proof of authorization to perform action
  ▪ Security
  ▪ Dissemination range for the action

• Anatomy of a Constrained Topic
  ▪ /Constrained/{Event Type}/{Constrainer}/
    {Allowed Actions }/{Distribution} /{Other
    “/”separated Suffixes}
Registration of Trace Entity

- First create a Trace topic
- Register with broker
  - Over constrained topic that ONLY broker subscribes to
- Traced Entity needs to sign registration message
  - Verify credentials and tamper evidence
- Broker generates session identifier
  - Along with the trace topic, is used to derive a constrained topic
    - ONLY the Broker subscribes to
    - ONLY traced entity publishes to.
Broker Operations

• Responsible for failure detection
  ▪ Must report status of traced entity to trackers
• Issues pings at regular intervals to traced entity
  ▪ Every ping has monotonically increasing message number, and timestamp.
  ▪ Ping responses should include both of these
    • To correlate requests and responses
  ▪ Failure suspicions and confirmations based on lack of ping responses
• Traced entity can notify broker about
  ▪ LOAD, Network metrics, Graceful exits and state transitions
Registering to receive traces

- Need to discover trace topic associated with a given traced entity
- Construct appropriate derived constrained topics to initiate interactions
  - Selectivity allows registering to different types of traces
- A broker generates traces ONLY if there are entities interested in traces
  - GAUGE INTEREST message issued periodically
Authorization

• Every trace message initiated by the traced entity checked for authorization (source) and tamper evidence

• Traces published by broker
  ▪ Broker needs to demonstrate authorization to generate traces.
    • This is verified by every broker that receives it
    • Cryptographic token generated by traced entity is included
Security

- Broker indicates that traces will be secured in the GAUGE INTEREST message.
- Trackers respond with their credentials.
- The broker secures the secret-trace-key
  - Using the tracker’s credentials
Performance

• Cryptographic Profile
  ▪ 1024-bit RSA with 160-bit SHA-1 and PKCS#1Padding.
  ▪ Symmetric encryptions/decryptions use 192-bit AES keys

• 4 CPU Xeon, 2.4GHz, 2GB RAM

• 100 Mbps LAN

• JVM 1.4
Performance: Topology

Traced Entity

Broker

Tracker (2 hops) → Tracker (3 hops) → Tracker (4 hops) → Tracker (5 hops) → Tracker (6 hops)
Trace Routing Overhead vs Number of Hops

TCP: Authorization

Number of Hops

Trace Time (Milliseconds)
Trace Routing Overhead vs Number of Hops

TCP: Authorization

TCP: Authorization & Security

Number of Hops vs Trace Routing Overhead
Trace Routing Overhead vs Number of Hops

- TCP: Authorization
- TCP: Authorization & Security
- UDP: Authorization
Trace Routing Overhead vs Number of Hops

TCP: Authorization
TCP: Authorization & Security
UDP: Authorization
UDP: Authorization & Security
Benchmarks: Tracker Increase

![Graph showing the increase in Tracker time with the number of Trackers. The graph indicates that as the number of Trackers increases, the Trace Time (ms) remains relatively constant.]
Optimization: Authorization

![Graph showing the relationship between Trace Time (ms) and Number of Hops. The graph compares performance with and without Authorization. The triangle line represents 'With Authorization' and the square line represents 'With Authorization Enhancement.' The x-axis represents the number of hops ranging from 2 to 6, and the y-axis represents trace time in milliseconds, ranging from 0 to 100.]
Conclusions

• Pub/Sub provides loosely-coupled framework for the tracing scheme
• Topic provenance lays the groundwork for the trace authorization scheme
• Transport independence facilitates wider use
• The costs introduced by the secure, authorized tracing scheme are acceptable for several applications.
• System can enforce secure and authorization schemes equally well.
• Selectivity and Change notifications reduce number of messages