An End-to-End Security Auditing Approach for Service Oriented Architectures

Mehdi Azarmi*, Bharat Bhargava, Pelin Angin, Rohit Ranchal*

Computer Science Department
Purdue University

Norman Ahmed, Asher Sinclair, Mark Linderman

Air Force Research Laboratory
Rome, NY

Lotfi Ben Othmane

Eindhoven University of Technology, Department of Mathematics and Computer Science

SRDS 2012

Outline

- Background
- Problem Statement
- Proposed Solutions
- Evaluation (And transition to Cloud)
- Future Directions
- Conclusion

What is SOA?

- Service Oriented Architectures (SOA) facilitate the interoperable and seamless interactions among services. The need to communicate with other service partners demands a seamless integration of services across organizational boundaries.
- Definition proposed by the Open Group, the Object Management Group (OMG), and OASIS:
  - Service-Oriented Architecture (SOA) is an *architectural style* in which a system is *composed* from a series of loosely coupled *services* that interact with each other by sending *messages*.
  - In order to be independent from each other, each service publishes its *description*, which defines its interface and expresses constraints and policies that must be respected in order to interact with it. A service is thus a building block for SOA applications.

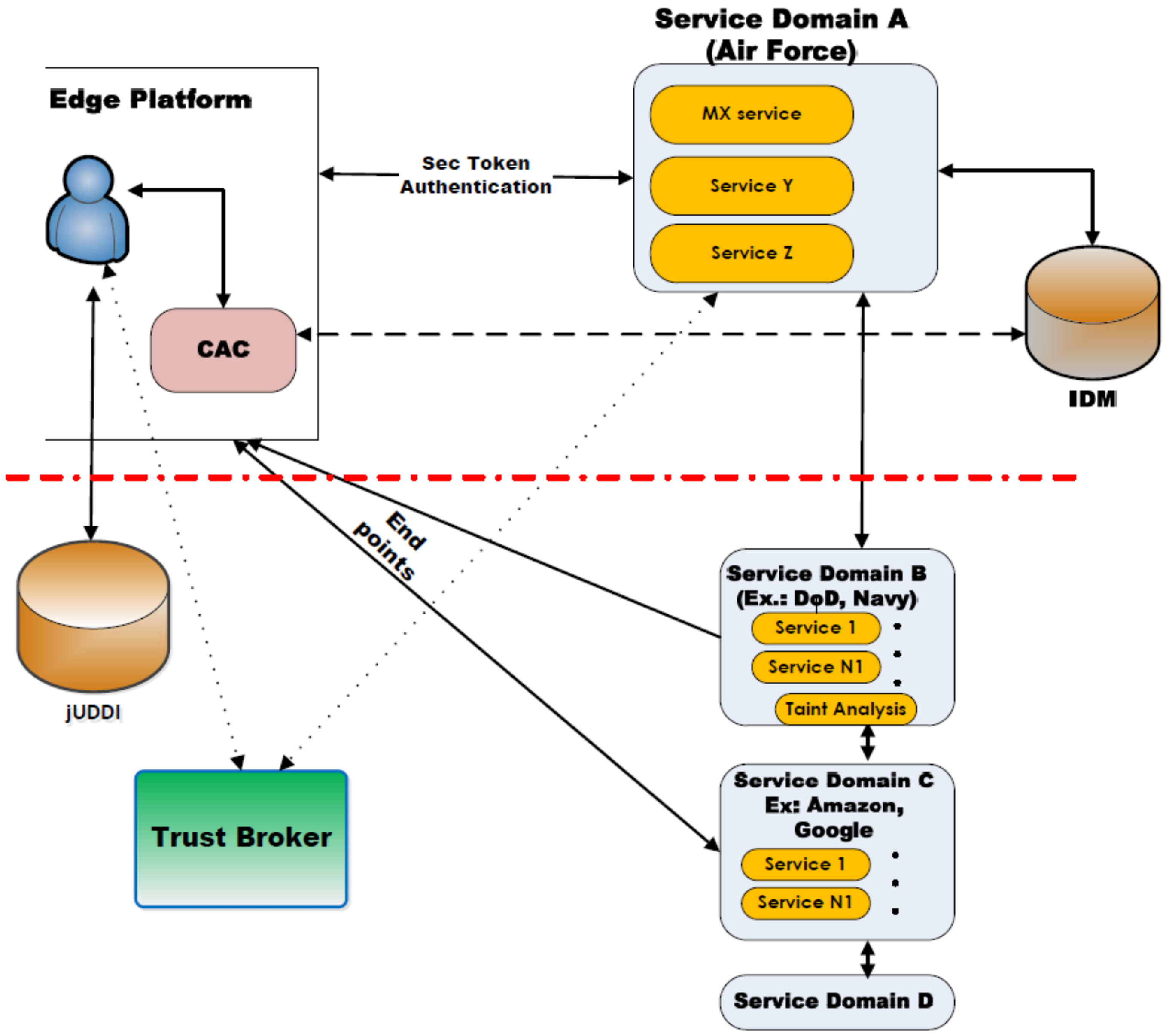
Security Challenges in SOA

- A new threat landscape (Large attack surface)
  - Diverse security administration domains
    - Security across organizational boundaries
  - User/services may get compromised
- Unauthorized external service invocation
  - User has no control on external service invocation within an orchestration or through a service in another service domain
- Businesses place a lot of trust in their partners (trust is not transitive!)
- Data leakage
  - Intermediate steps of service execution might expose messages to hostile threats (data leakage)
- Violations and malicious activities in a trusted service domain remain undetected
- Once one of the services is compromised, the whole system should not fall apart! (APTs)

Problem Statement

- The channels of communication between the participating entities in a SOA application are much more vulnerable:
  - Compared to operating systems or within the boundaries of an organization's computer network
  - To alleviate the security vulnerabilities that were introduced in the complex context of SOA applications, numerous and often overlapping security standards by the industry actors exist.
- **How to provide End-to-End auditing in SOA?**
- **How to define and build trust across domains?**
  - The trust issue is more complex in inter-organizational context than it is in traditional fields of computing

SOA End to End Security Architecture



Attack Model

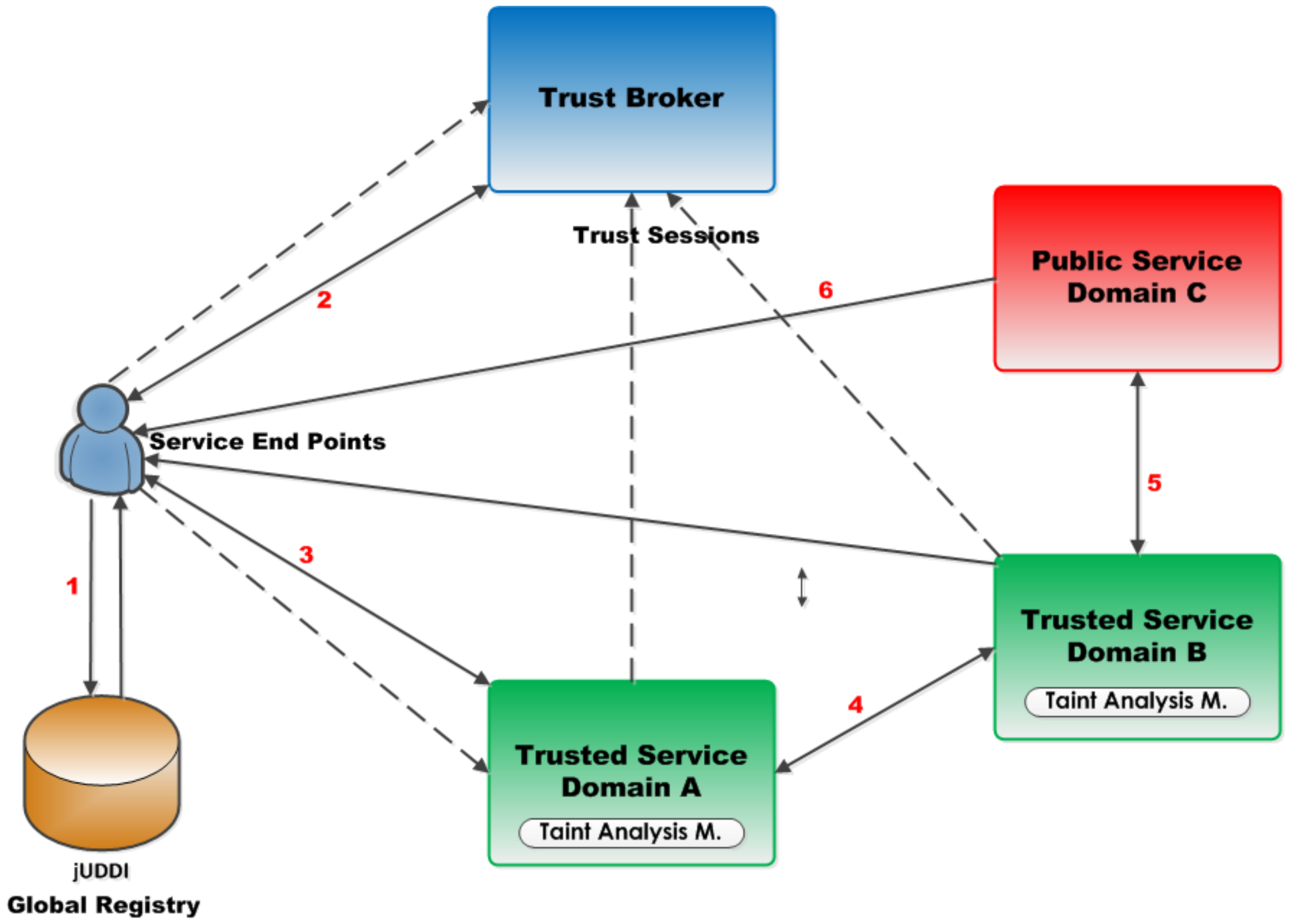
- Attackers may have full access to the in-transit SOAP messages (MITM attacks)
- Attackers may gain full control of certain number of services in a domain.
- Some domains may have inside attackers
- Trust broker (TTP) is secure.
- Cloud providers support Trusted Computing facilities (which is realistic with vTPM)

Proposed Approach

To address these challenges, we designed and implemented:

- A comprehensive security architecture for SOA.
- A novel service invocation control mechanism for SOA using dynamic taint analysis (TA)
- A trust broker (TB) system that maintains trust and classifies services. TB is used for dynamic validation and verification of services and keeps track of history of service invocations.
- functionality for using widely adopted web service WS-* standards (WS-Security, WS-Trust)
- A prototype implementation of proposed approaches based on open source technologies

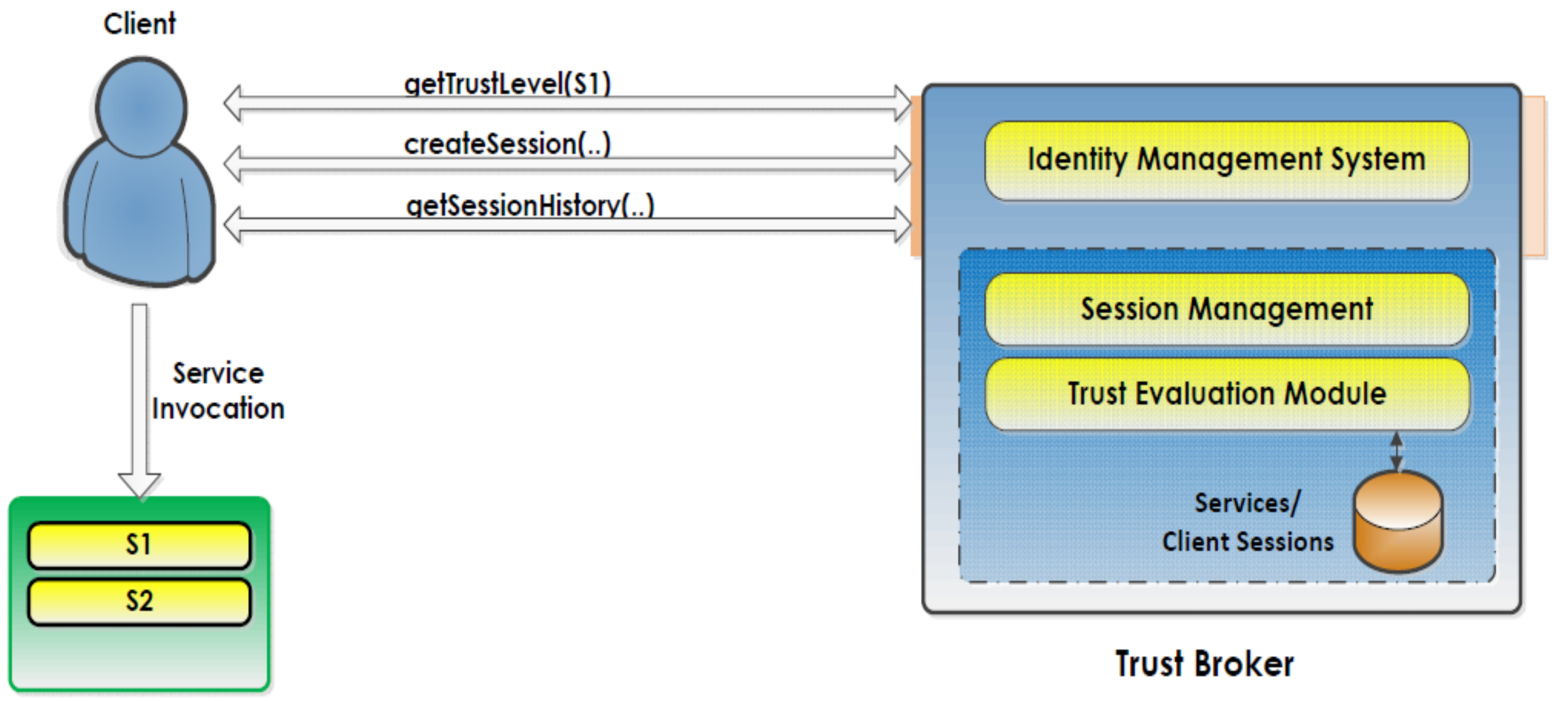
System Architecture and SOA Baseline Scenario



SOA End to End Architecture

1. UDDI Registry request
2. Forwarding the service list to Trust Broker and receive a categorized list
3. Invoking a selected service
4. Second invocation by service in domain A
5. Invoking a service in public service domain
6. End points (Reply to user)

Trust Broker



1. Calculates trust for a given set of services:
   - Given a set of services identified with UDDI service keys, Trust Broker returns trust categories for all of those services as determined by the Trust Evaluation Subsystem.
2. Manages end to end user/service-invocation session.

Trust Broker

Trust Evaluation Subsystem

– Classification of services into Trust categories

- Certified (supports WS-* specifications and has Taint Analysis Module)
- Trusted (having Trust value above threshold)
- Untrusted (having Trust value below threshold)

– Trust calculation is based on parameters such as:

- WS-* support specified in Service Level Agreement (SLA)
- Trustworthiness of services in Orchestration specified in SLA
- History of previous service runs (Using sessions)
- Taint analysis feedback
- User experience feedback

Trust Broker

Trust Evaluation Subsystem

- Calculating Trust
  - Using weighted moving average model
  - Recent feedbacks for a service are weighted more heavily than feedback further in the past
  - The trust value T_s for a service ***S***, with SLA ***L,*** getting feedback ***F*** at time ***t*** is updated using the equation*:*

$$T_s(t) = \beta \times [\alpha \times T_s(t-1) + (1-\alpha) \times F] + (1-\beta) \times L$$

where $\alpha < 0.5$ and β is evaluated based on the appropriate WS-* supported

Trust Broker

Session Management Subsystem

- Extending the Trust boundary
  - Manages end to end service invocation session
    - User creates a session with Trust Broker and selected service
- Maintaining end to end Trust sessions across different domains
- Auditing service behavior including violation and malicious activities
  - Taint Analysis and user feedback to Trust Broker for updating Trust
  - Trust Maintenance

Taint Analysis

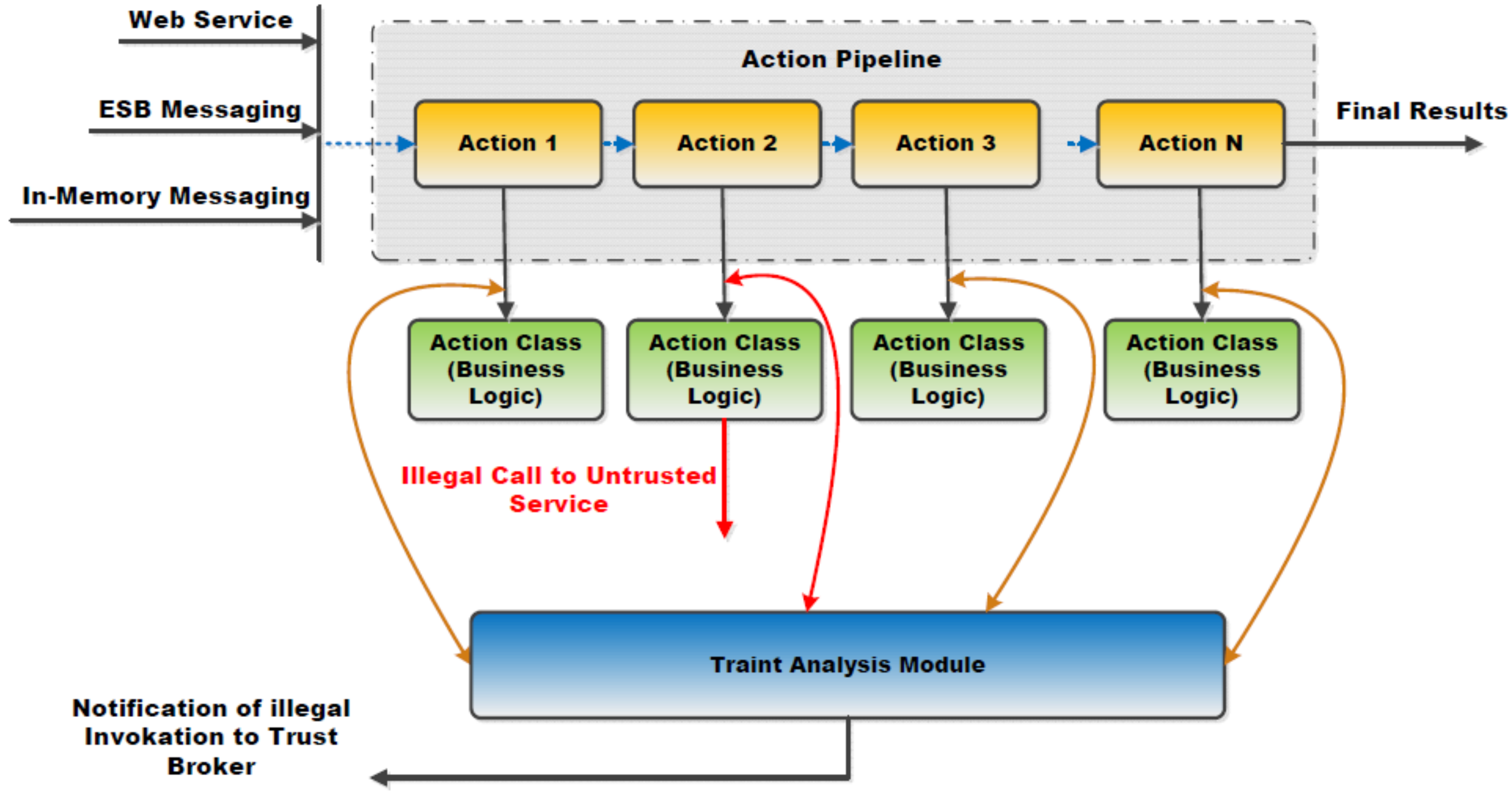
- What is Taint Analysis/IFC (Information Flow Control)?

- How it fits into the proposed architecture?
  – Independent of services (We do not need to change the services or access the source code of services)
  – Interception of Service execution (Service will remain transparent)

Taint Analysis

- Using AOP (Aspect Oriented Programming)
  - Instrumenting classes based on predefined pointcuts
  - Low performance overhead (ideal solution)
- How it works?
  - Load-time instrumentation
  - The whole Application server is under control
  - Flexible granularity
    - Package/Class level
    - Method level
    - Field level
  - Instrumenting classes in *action pipeline*

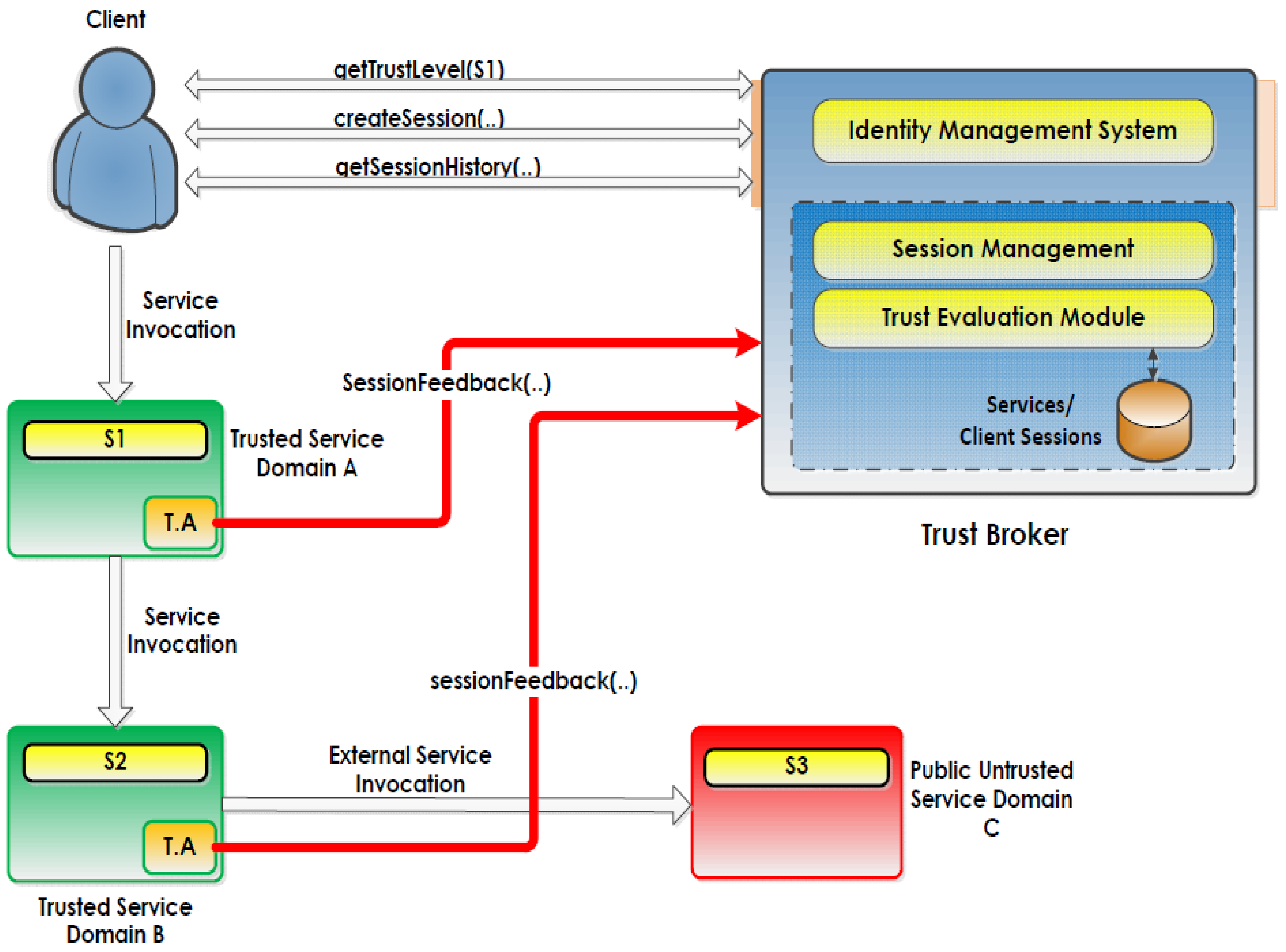
Taint Analysis of Services


Web Service
ESB Messaging
In-Memory Messaging
Action Pipeline
Action 1
Action 2
Action 3
Action N
Final Results
Action Class (Business Logic)
Action Class (Business Logic)
Action Class (Business Logic)
Action Class (Business Logic)
Illegal Call to Untrusted Service
Traint Analysis Module
Notification of illegal Invokation to Trust Broker

Taint Analysis

- Where to deploy this module?
  - Only in Trusted Service Domains
    - Detection of insider attacks
    - Detection of compromised services
    - Detection of outbound connections
  - In Public Domain
    - Enforcing service composition policies

Interaction of Taint Analysis and Trust Broker

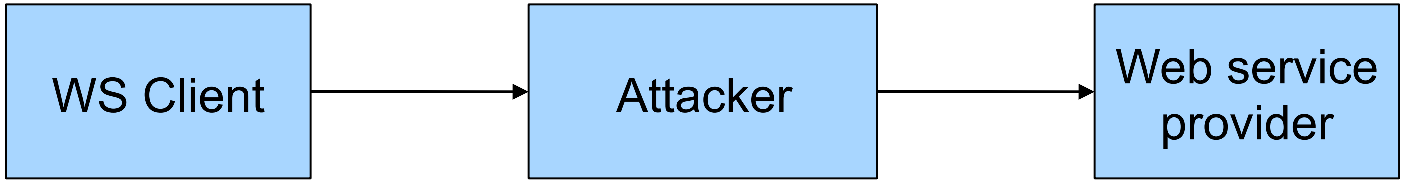


Evaluation of the Proposed Solution

- **Security Evaluation**
  - The implemented prototype will be evaluated in terms of its effectiveness in mitigating various attacks including the following attacks
  - XML Rewriting Attack
- **Performance Evaluation**
  - Response Time
  - Throughput

XML Rewriting Attack

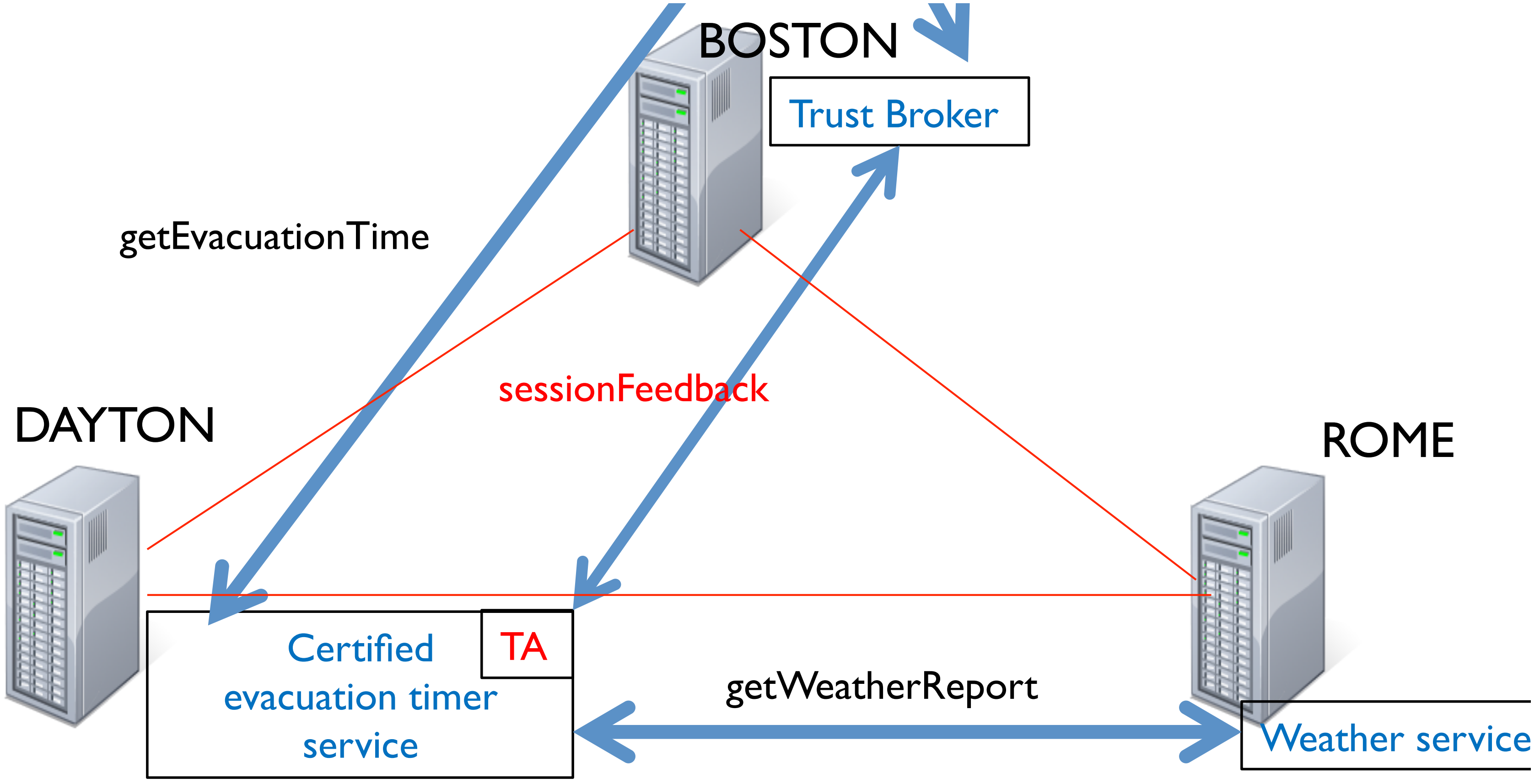
- XML rewriting attack commonly refers to the class of attacks which involve in modifying the SOAP message. (Replay, Redirect, Man in the middle, multiple header etc.)
- Exploring how certain XML rewriting attacks can be detected by the Taint Analysis component and Trust Broker



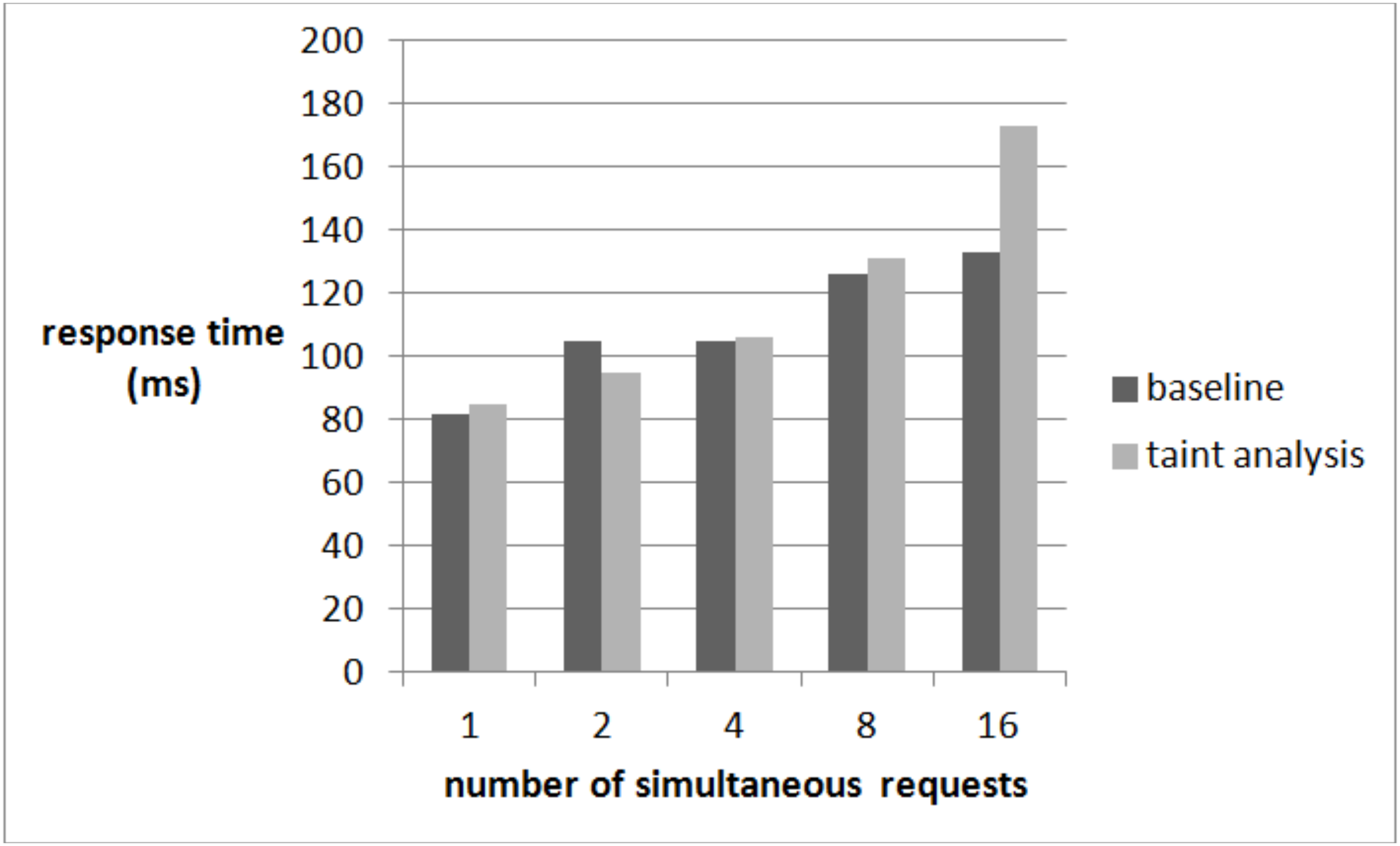
Cloud Performance Evaluation

- Why Cloud computing?
  - Service-oriented
  - Utility-based
  - Shared
  - SLA-driven
- Service domains were placed in Amazon EC2 AMIs (Amazon Machine Instances)
- We have conducted experiments to measure the performance impact of using cloud computing.
- We installed and configured the following components in the Amazon Cloud (same configuration as on-site):
  - Jboss ESB Server and services
  - Trust Broker
  - UDDI

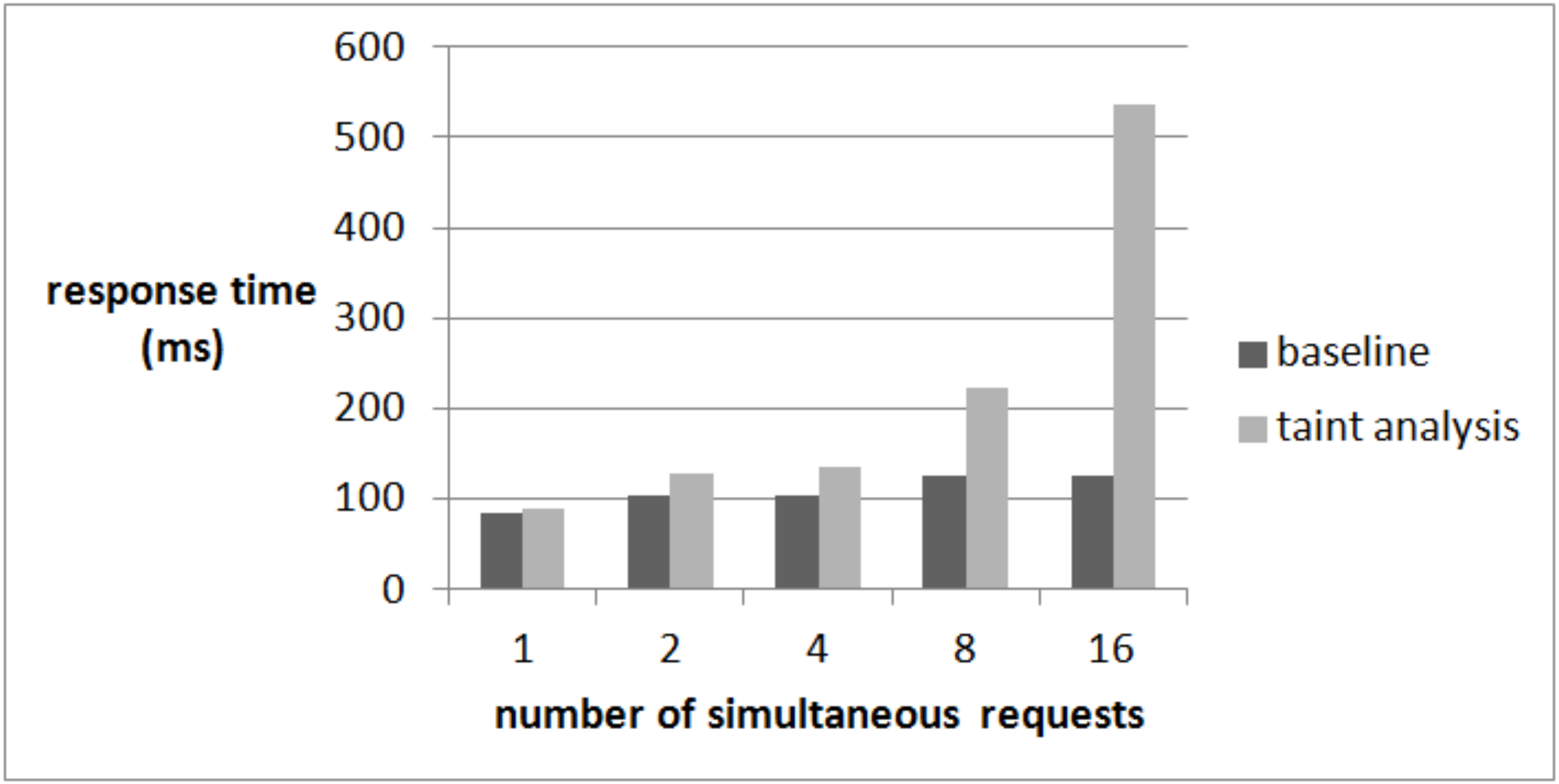
LAN Setup


BOSTON
Trust Broker
getEvacuationTime
sessionFeedback
DAYTON
ROME
Certified evacuation timer service
TA
getWeatherReport
Weather service

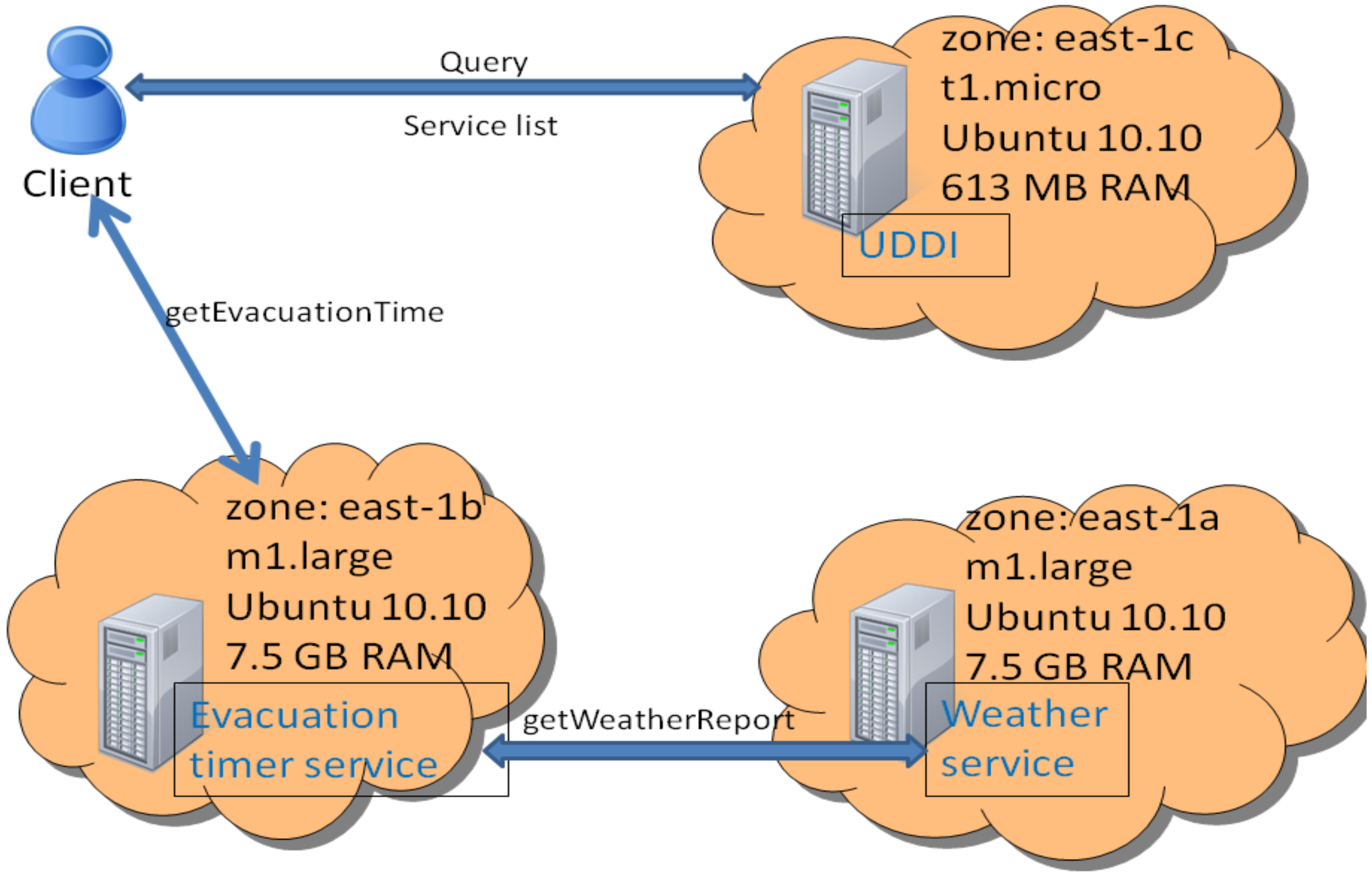
LAN-Setting 300 Requests



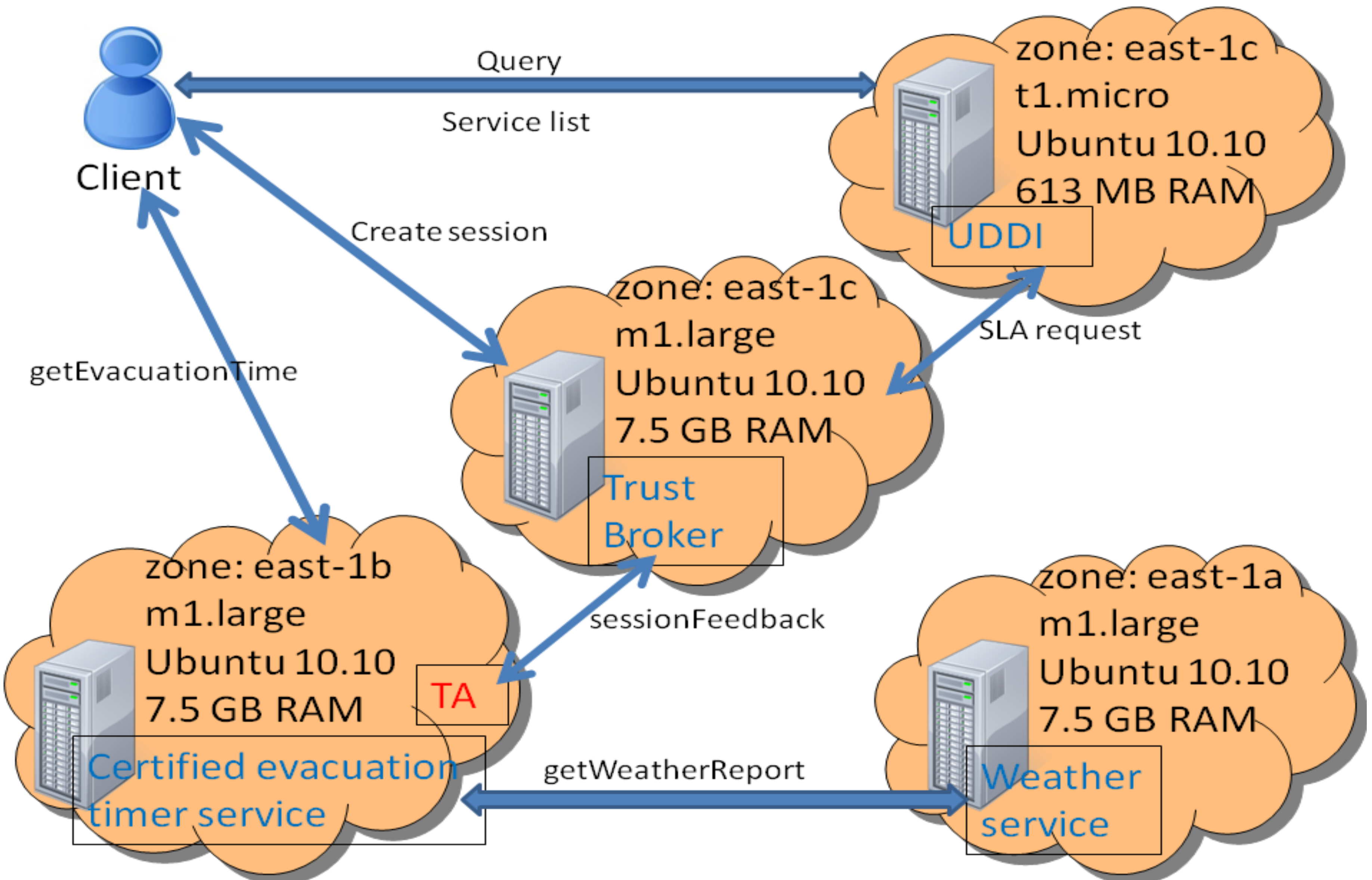
LAN-Setting 400 Requests



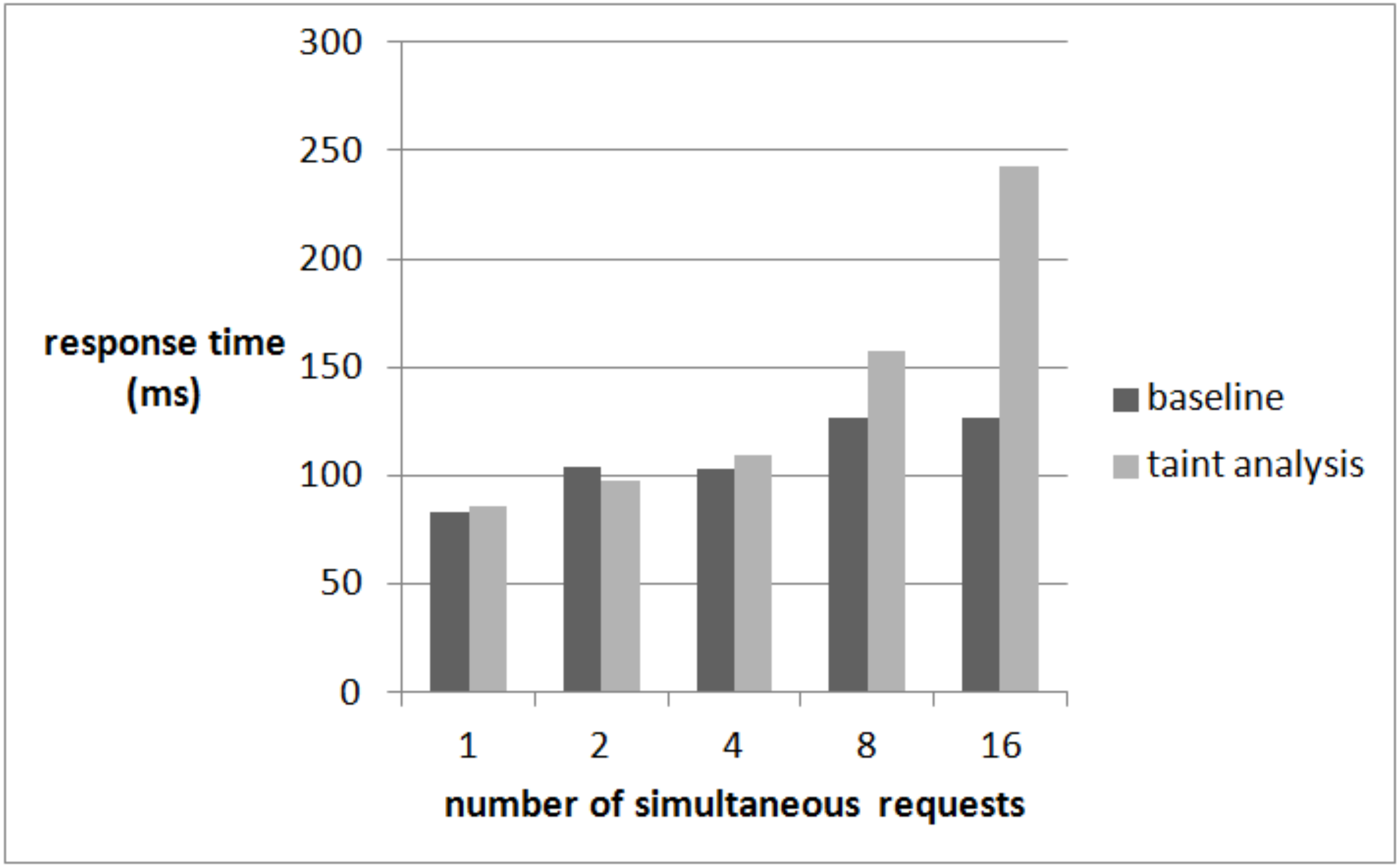
Cloud Setup – Baseline



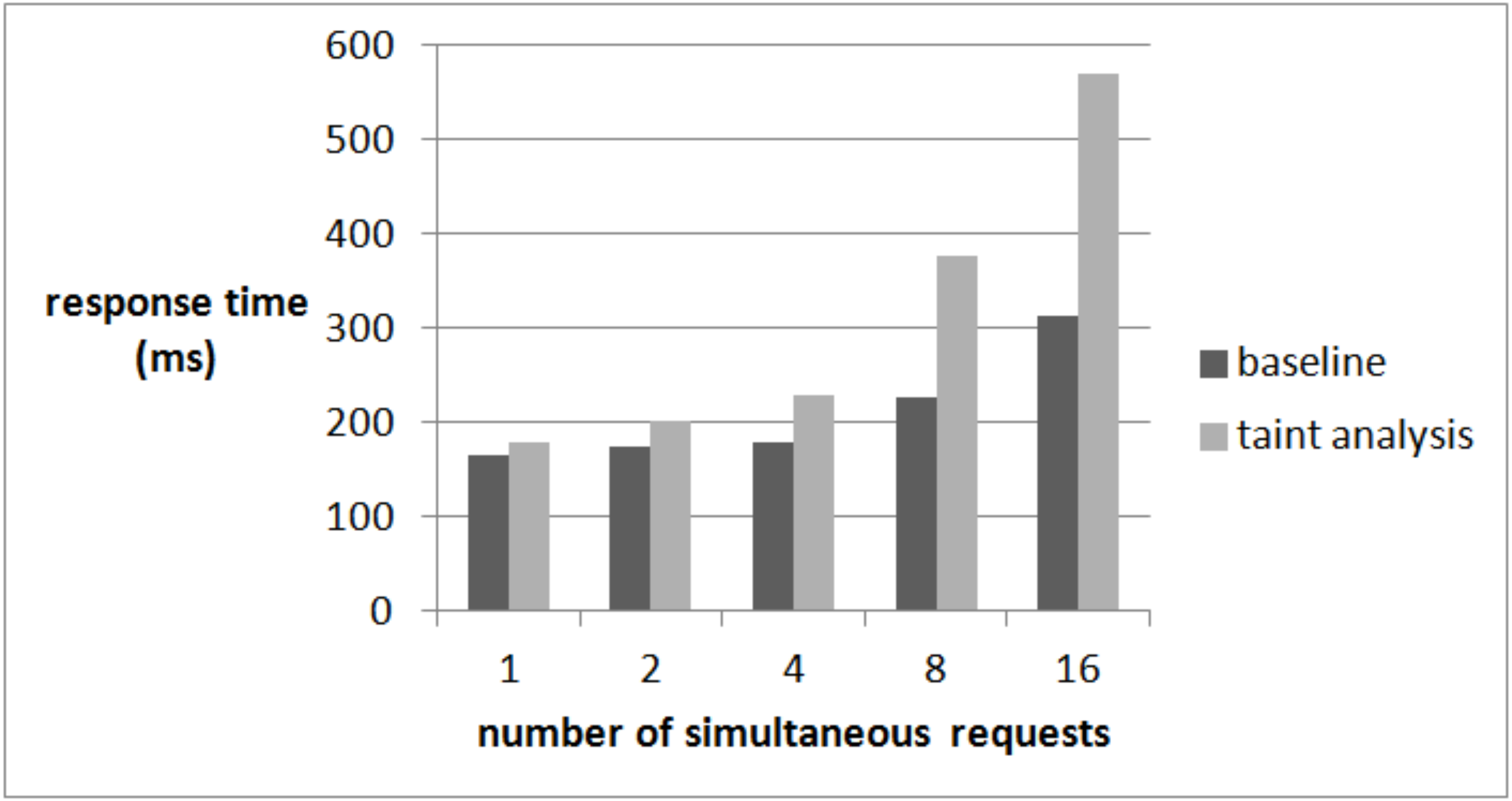
Cloud Setup – Taint Analysis



Amazon EC2- 300 Requests



Amazon EC2- 400 Requests



Conclusion

- Proposed end to end security challenges in SOA
- Proposed a flexible security architecture for auditing SOA
- Proposed new holistic monitoring system based on IFC and taint analysis
- Conducted experiments on different settings

Future Directions

- IFC Policy Enforcement
- Using TPMs along with taint analysis framework to provide a stronger security
- Providing active defense and attack-resiliency using cloud computing

Questions?