MIDDLEWARE 2009

DECEMBER 3rd, 2009
URBANA, ILLINOIS

DR-OSGi: Hardening Distributed Components With Network Volatility Resiliency

Young-Woo Kwon[1], Eli Tilevich[1], and Taweesup Apiwattanapong[2]

[1]Software Innovations Lab
Dept. of Computer Science
Virginia Tech

[2]NECTEC

Motivating Example: *"DNA Hound"*

```
try {
//a remote service invoca
}
catch (RemoteException e
//exception handling coc
```

```
try {
//a remote service invocation
}
catch (RemoteException e) {
//exceptio
}
```

```
try {
//a remote service invocation
```

```
try {
}
```

Too much custom-coded exception handling!!

```
//exception handlin
}
```

```
//a remote service inv
}
catch (RemoteExceptio
//exception handling c
}
```

```
try {
//a remote service invocation
}
catch (RemoteException e) {
//exception handling code
}
```

```
invocation
ption e) {
ing code
```

Background

- Component Infrastructures
  - OSGi
    - Service-oriented component platform for Java (Encapsulation with bundles, Management Architecture)
  - R-OSGi
    - Remote OSGi to support distributed services (Proxy based distribution, Exceptions to signal network failure)
- Network Volatility
  - Temporary network outage
    - Random channel errors, node mobility, and congestion

Background

- Disconnected operations
  - Commonly used techniques
    - Caching
    - Queuing
    - Replication
    - Hoarding
    - Multi-modal operations
- Aspect Oriented Programming
  - Modularizing cross-cutting concerns

Disconnected Remote OSGi: DR-OSGi

- Treat network volatility as a disease
  - Unavoidable and omnipresent
  - Cannot always prepare for it in advance
    - E.g., naïve implementation practices, execution environment changes, etc.
- Leverage the collected wisdom of distributed system developers
  - Disconnected operations, fault-tolerant designs
    - Custom-coded for individual applications
    - Not reusable and hard to customize

Disconnected Remote OSGi: DR-OSGi

- Improve on state of the art in middleware
  - Advanced middleware (e.g., R-OSGi)
    - Raise exceptions in response to network volatility
    - Reestablish connection once the network is up again
    - Require programmer handle exceptions manually
- Treat symptoms of volatility systematically
  - Use hardening strategies (e.g., medicine) for QoS
  - Detect volatility, instantiate and deploy a strategy
  - Return to normal operation once the network is up again

Treating Symptoms

- Flu
  - Cannot eliminate the flu
  - Treat symptoms
  - Improve the quality of life
- Network Volatility
  - Cannot avoid network volatility
  - Treat symptoms
  - Improve the quality of service (QoS)

DR-OSGi

Disconnected Operations

- Caching
- Queuing
- Replication
- Hoarding

Contributions

1. An approach for hardening distributed component applications with network volatility resiliency

2. Programming abstractions to express hardening strategies
   - Customizable hardening strategies for networks and applications
   - Reusable hardening strategies across applications

3. A reference middleware implementation that can harden existing distributed component applications

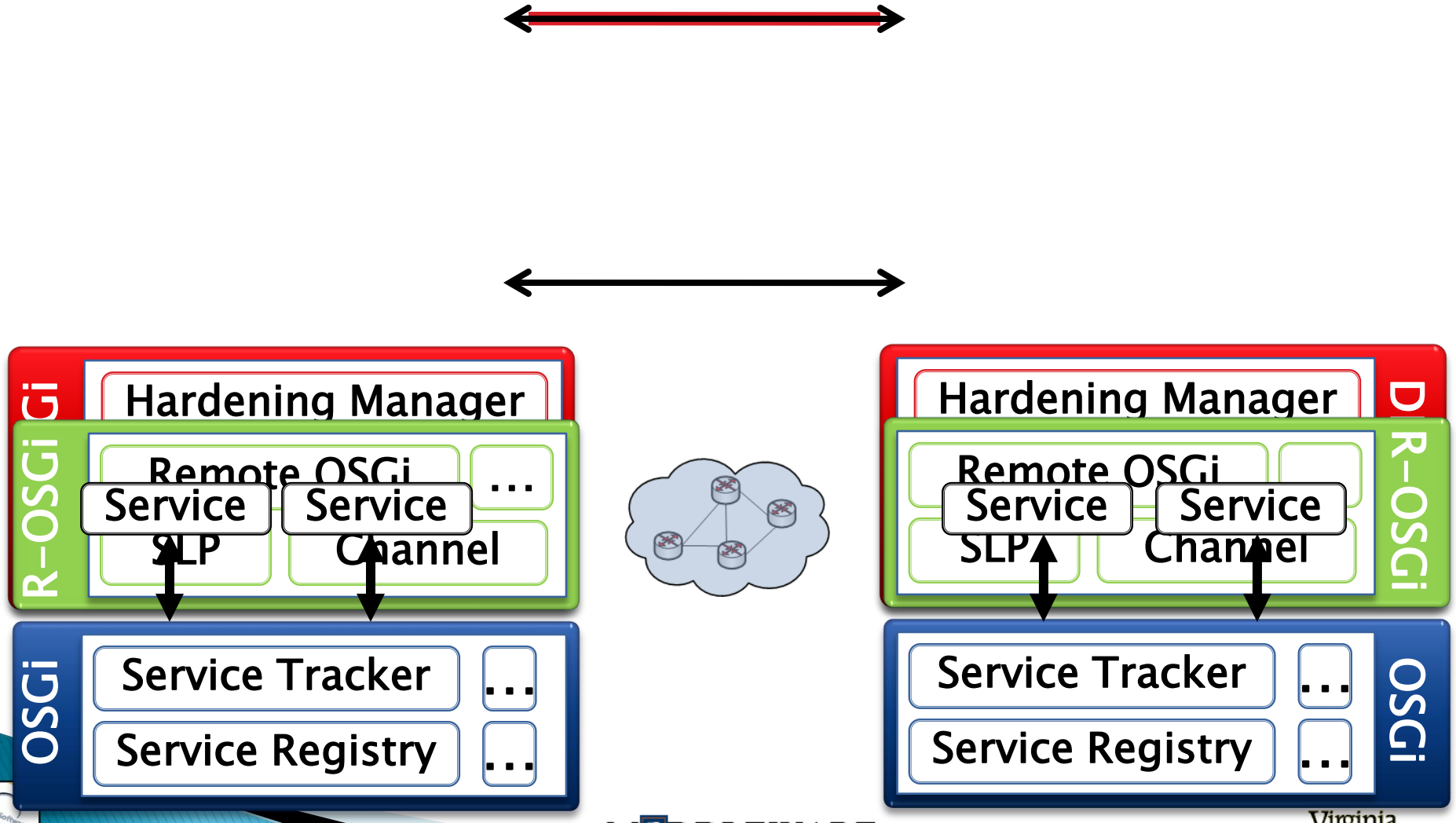
Roadmap

- DR-OSGi Design & Architecture
- Treating Symptoms of Network Volatility
- Evaluation
  - Benchmarks
  - Case Study
- Future Work
- Conclusion

DR-OSGi Design Objectives

- Transparent
  - Should not affect the core functionality of the underlying application
- Flexible
  - One should be able to start and stop a hardening strategy dynamically
- Extensible
  - Hardening strategies should be easy to program, modify, and extend

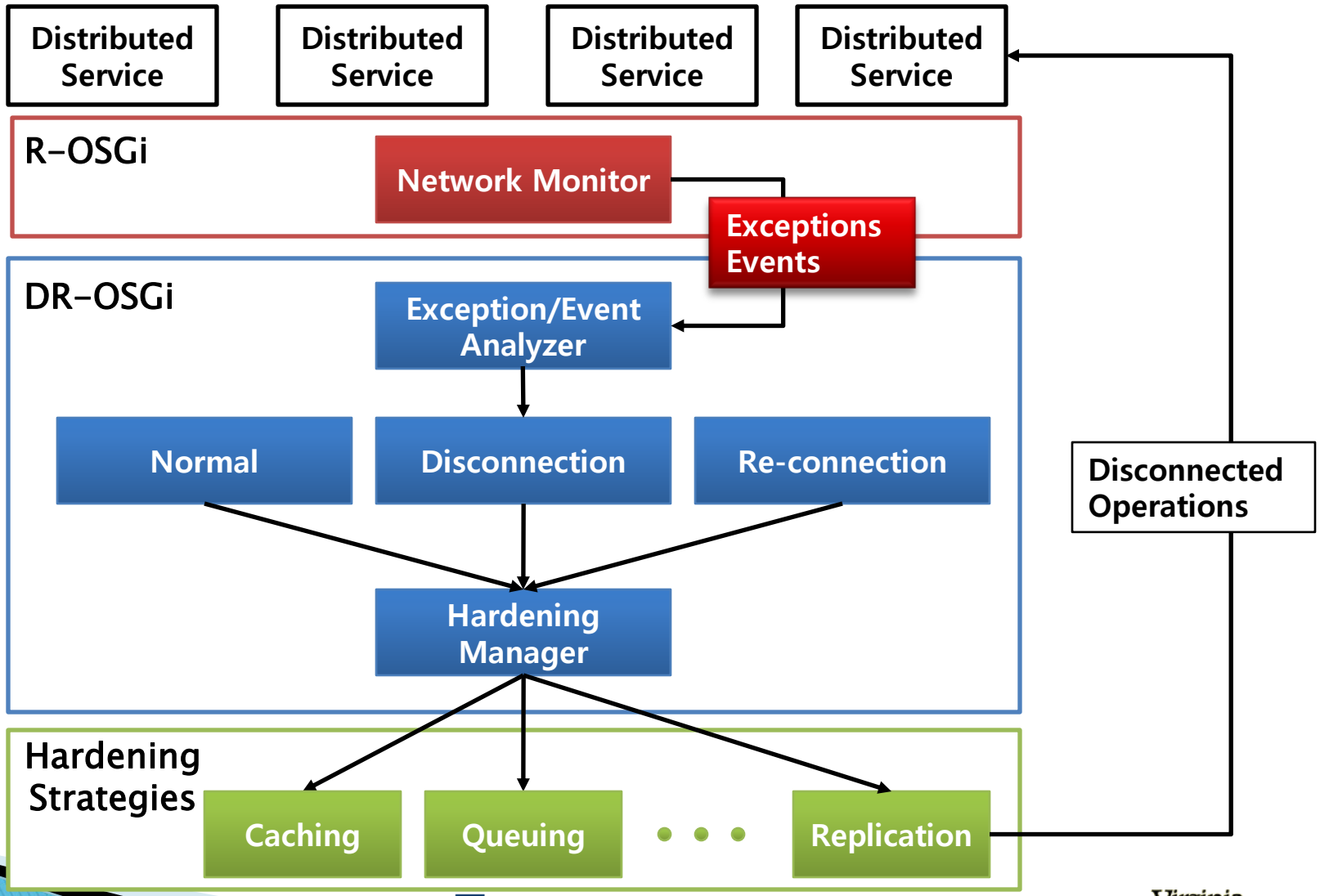
DR-OSGi Architecture



Roadmap

- DR-OSGi Design & Architecture
- Treating Symptoms of Network Volatility
- Evaluation
  - Benchmarks
  - Case Study
- Future Work
- Conclusion

DR-OSGi

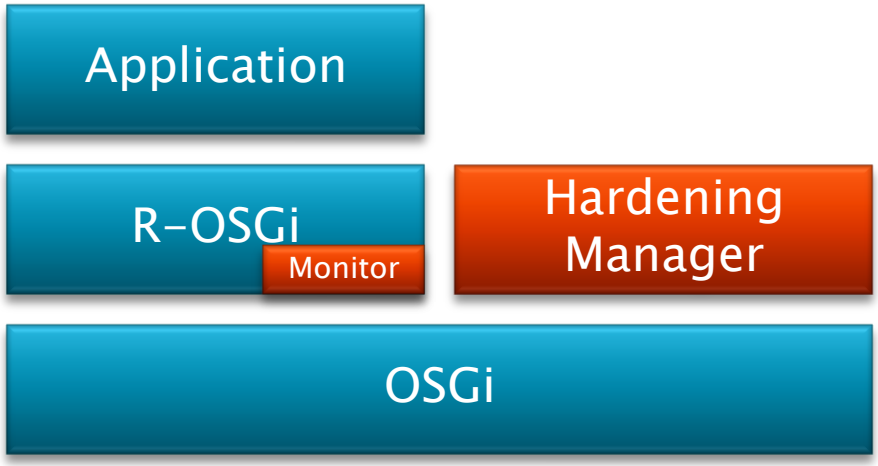

Distributed Service
Distributed Service
Distributed Service
Distributed Service
R-OSGi
Network Monitor
Exceptions Events
DR-OSGi
Exception/Event Analyzer
Normal
Disconnection
Re-connection
Hardening Manager
Disconnected Operations
Hardening Strategies
Caching
Queuing
Replication

Treating Symptoms of Network Volatility

- Detect network volatility
  - Hardening manager
    - Monitor network condition
    - Catch network related exceptions
    - Manage hardening strategies
- Cope with network volatility
  - Hardening manager
    - Notify network condition to hardening strategies
  - Hardening strategy
    - Provide disconnected operations

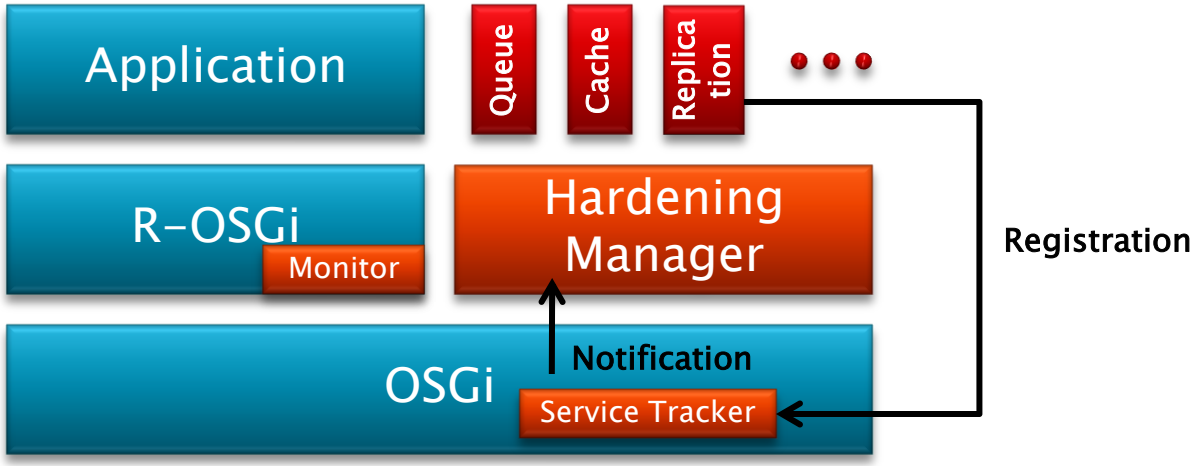
Hardening Manager

- Hardening manager as an aspect
  - Network monitor is woven at runtime
  - JBoss AOP for runtime weaving
    - Weave a standard OSGi bundle into the R-OSGi at bundle life cycle (e.g., start/stop)



Hardening Manager

- Management
  - Hardening strategy bundles
    - Track registration/unregistration of hardening strategy bundles
  - Distributed component applications
    - Manage applications' disconnected operations



Hardening Strategy

- Hardening strategy bundles
  - Standard OSGi bundles
  - A simple deployment configuration file
- Reusable hardening strategies
  - Across different distributed applications
- Extensible hardening strategies
  - Can derive new strategies
    - Customize library bundles to create specialized domain-specific strategies

Hardening Strategy

- Programming model

```
public interface DisconnectionListener {
    public Object disconnectedInvoke( … );
    public Object reconnected( … );
    public void remoteInvoke( … );
    public void serviceAdded( … );
    public void serviceRemoved( … );
}
```

```
public  class Queuing implements DisconnectionListener { … }
```

- Configuration

```
RemoteServiceName=org.mypackage.MyBundle
HardeningServiceName=org.otherpackage.CachingHardening
```

Roadmap

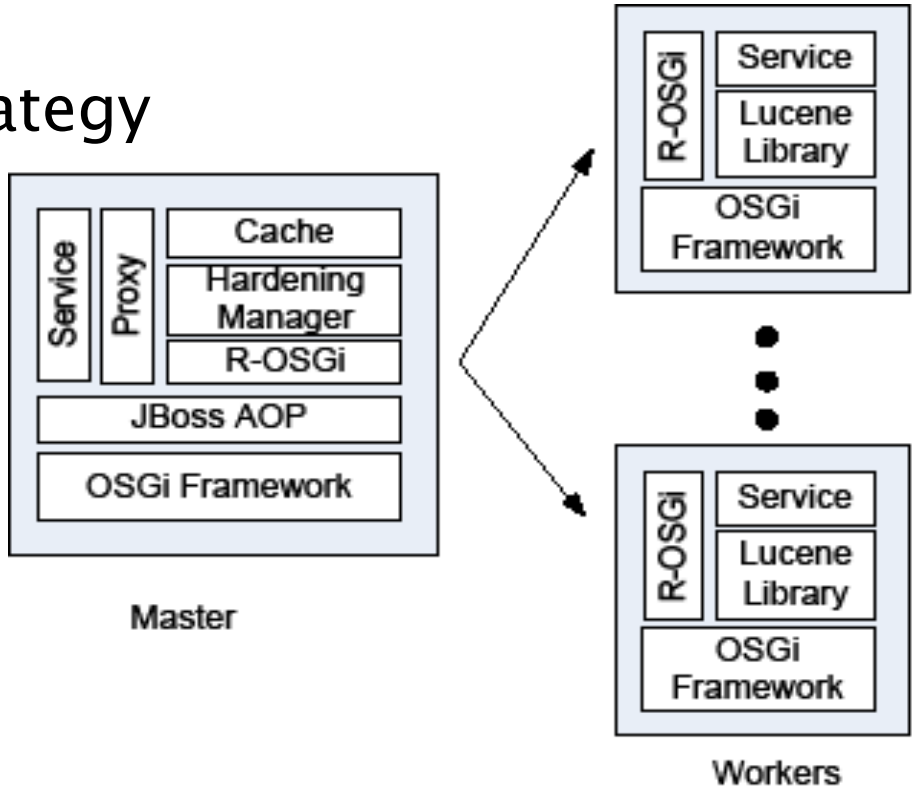
- DR-OSGi Design & Architecture
- Treating Symptoms of Network Volatility
- Evaluation
  - Benchmarks
  - Case Study
- Future Work
- Conclusion

Evaluation

- Remote Log
  - Based on existing OSGi log service
  - Queuing hardening strategy
- Remote User Admin
  - Based on existing User Admin service
  - Caching hardening strategy

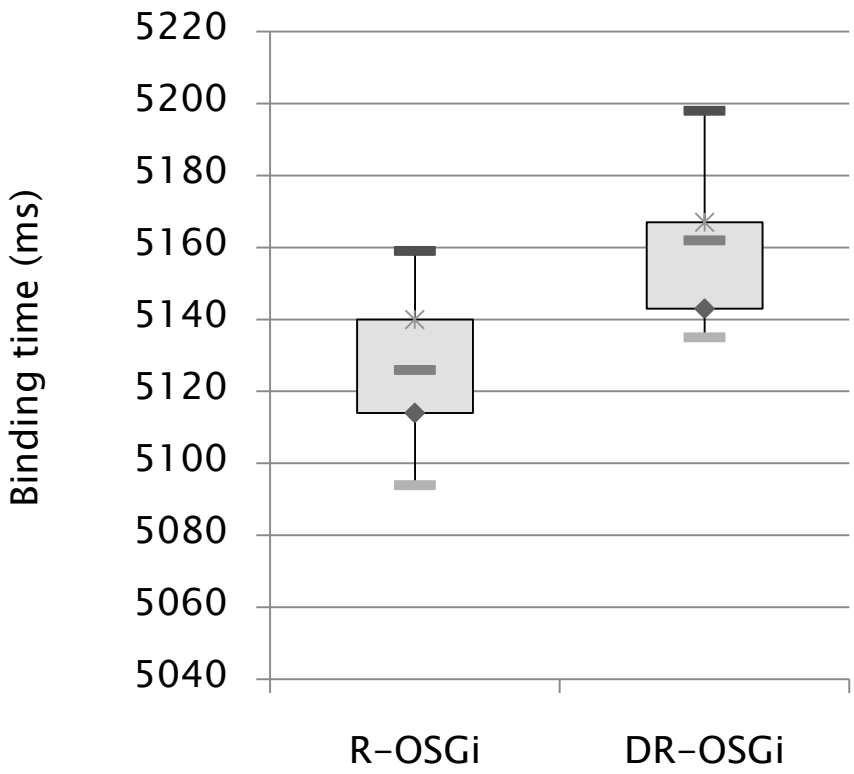
Evaluation

- Distributed Search Engine
  - Lucene search engine library
  - Master-worker model
  - Caching hardening strategy
  - Benchmarks



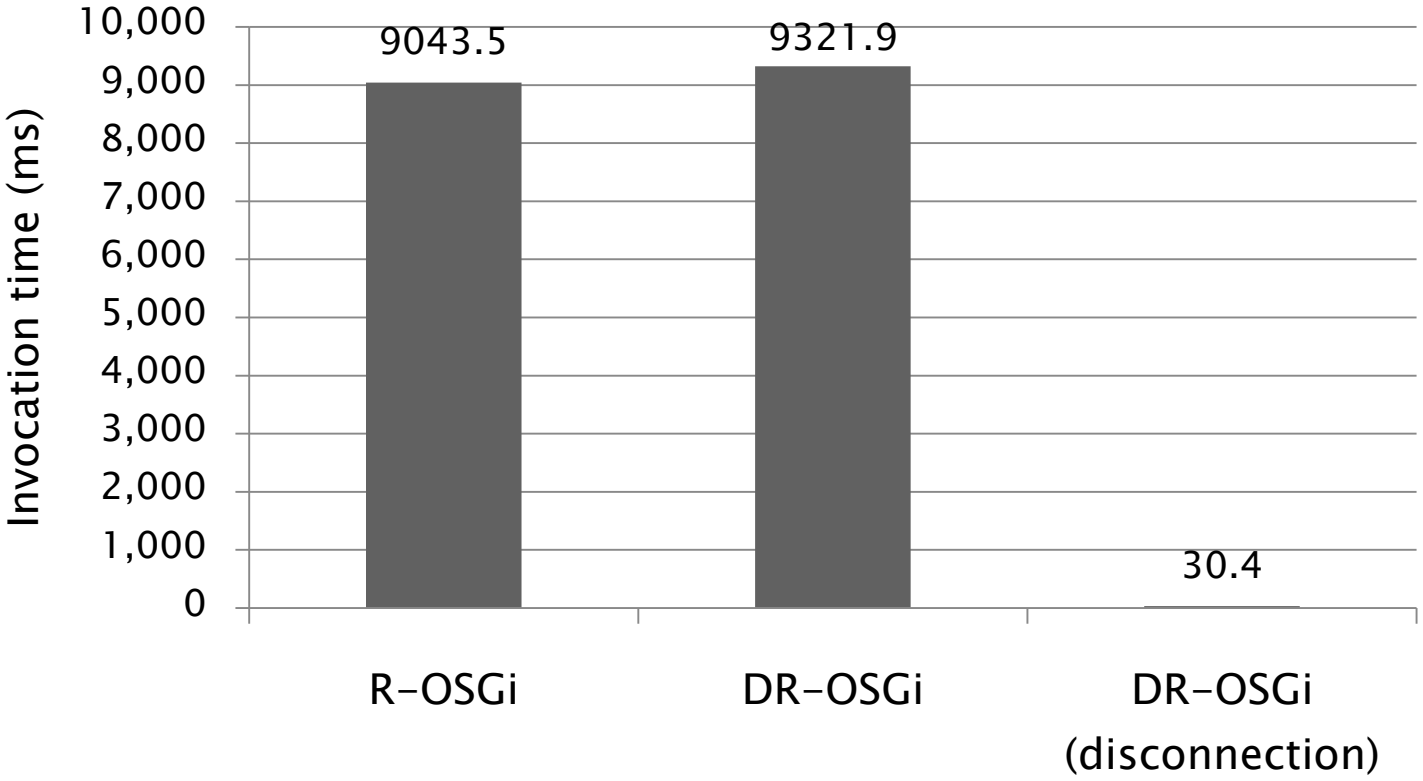
Benchmarks

- Binding time



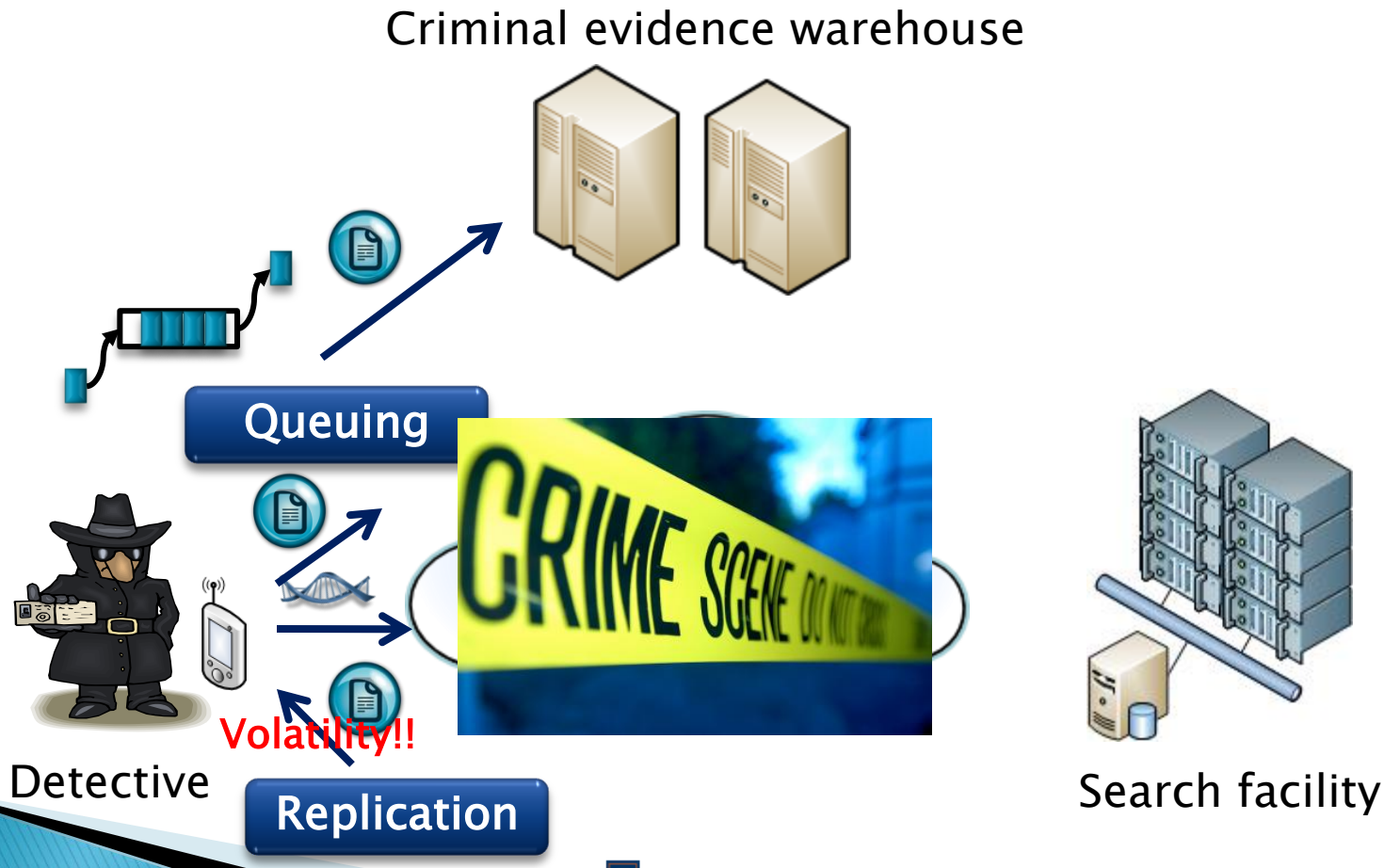
Benchmarks

- Invocation time



Case Study

- DNA Hound System



Roadmap

- DR-OSGi Design & Architecture
- Treating Symptoms of Network Volatility
- Evaluation
  - Benchmark
  - Case Study
- Future Work
- Conclusion

Future Work

- Fault tolerant distribution middleware
  - Network failure, Software failure, Security, etc
- Application to other distributed component infrastructures
  - Ex. Remote Service of OSGi R4.2

Conclusions

- Distributed applications suffer from temporary network outages
- Distribution middleware should provide fault tolerance mechanism
  - Improve reliability and QoS
- DR-OSGi
  - Systematic hardening approach to cope with network volatility
  - Reusable and extensible hardening strategies

Thank you!

Related Work

- Disconnected operations
  - Rover toolkit
  - Mobile Extension
  - Odyssey
  - FarGo-DA
- Middleware support for new functionality
  - Adaptive CORBA template (ACT)
- Middleware support for fault-tolerance
  - GRAFT
- Systematic security hardening