Performance Enhancing Proxies for Java™2 RMI over Slow Wireless Links

Stefano Campadello

Oskari Koskimies, Kimmo Raatikainen
Dept. of Computer Science, University of Helsinki
Finland

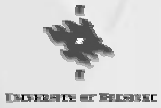
Heikki Helin
Sonera Ltd., Sonera Mobile Operator
Finland

Remote Method Invocation

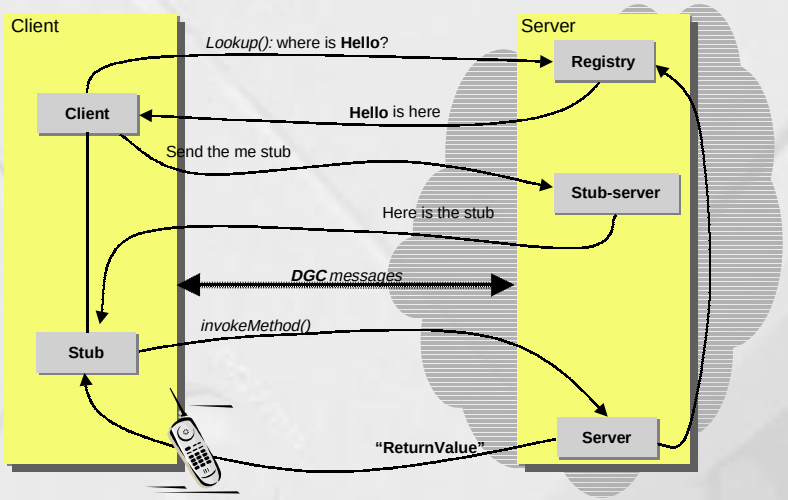
- RMI interface lets Java objects on different hosts communicate with each other in a transparent way
- Clients can invoke methods of a remote object as local methods
- Preserve the object oriented paradigm in distributed computing

Wireless Environment

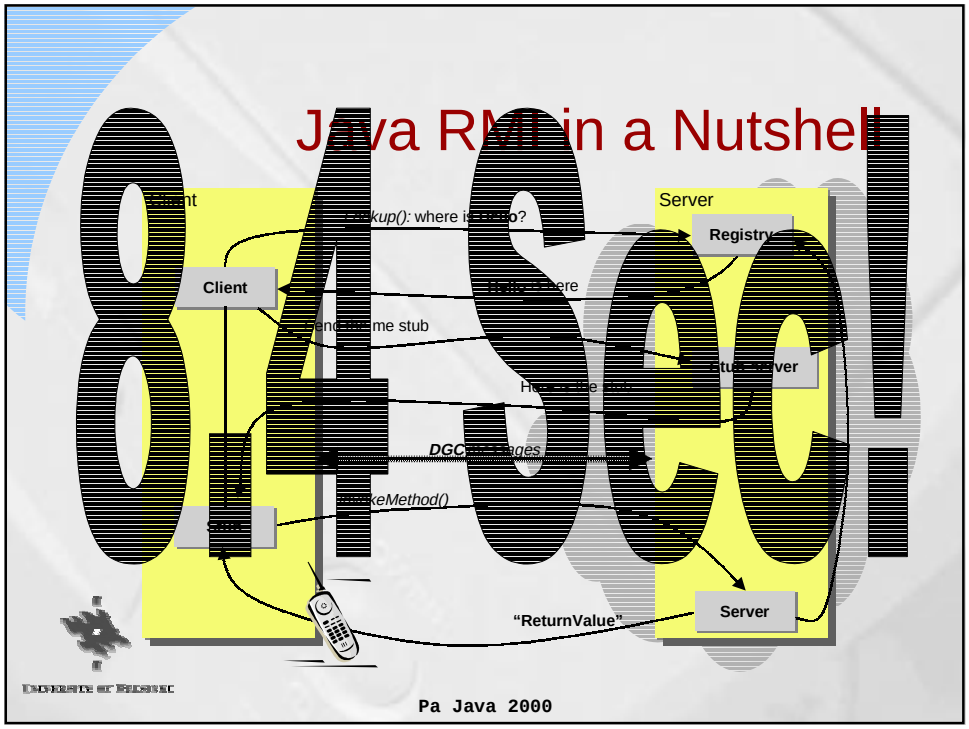
- Characteristics
  - Low bandwidth
  - High and variable delays
  - Sudden disconnections
  - High costs



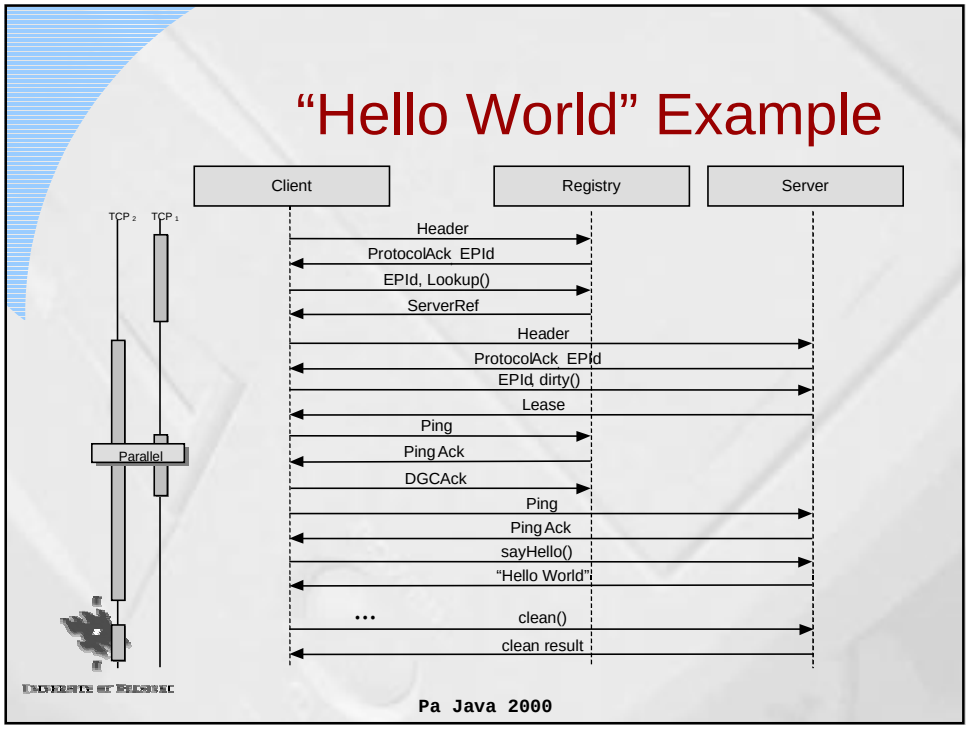
Java RMI in a Nutshell



Java RMI in a Nutshell



"Hello World" Example



Data traffic analysis

	Client to Server and Registry	Server and Registry to Client	Total
Registry Lookup	55 (6%)	276 (42%)	331 (20%)
Invocation Data	41 (4%)	37 (6%)	78 (5%)
DGC Data	831 (85%)	305 (46%)	1136 (69%)
Protocol Overhead	52 (5%)	40 (6%)	92 (6%)
Total	979 (100%)	658 (100%)	1637 (100%)

RMI Optimization

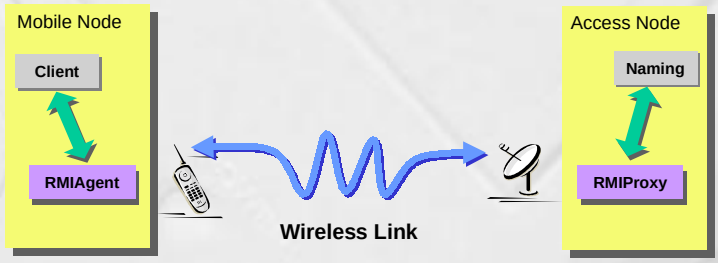
- Maintain compatibility with Java RMI specifications
- Avoid redundancy in communication protocol
- Use compression and caching to minimize data transmission

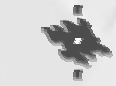
Java RMI Optimization

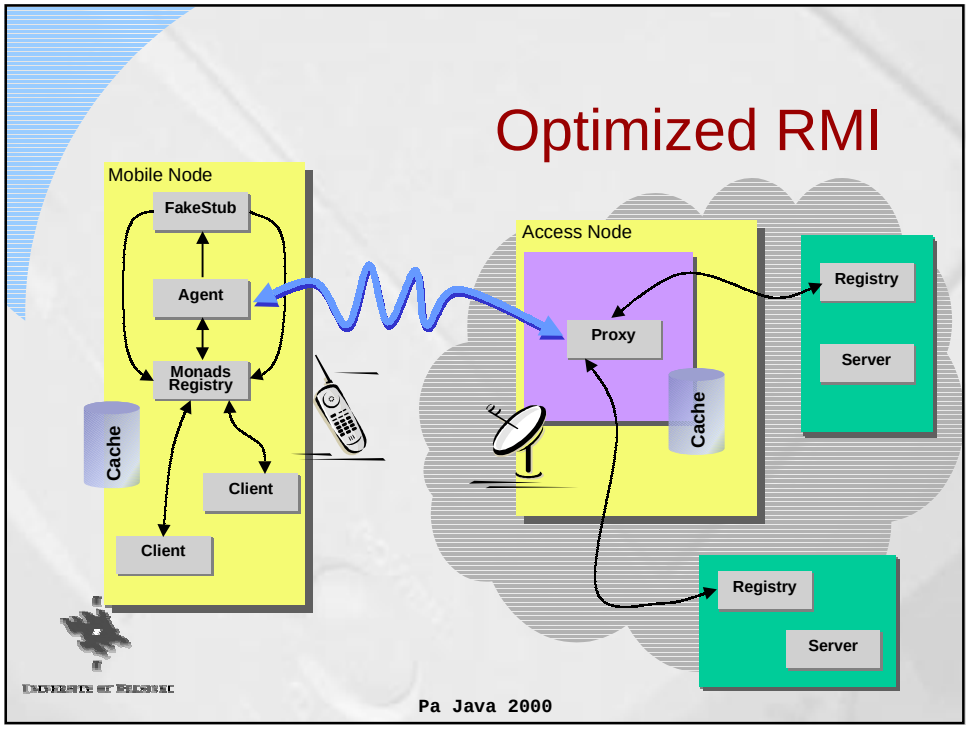
- Protocol
  - Use of Mediators to minimize the exchange of data through the wireless link.
- Data Communication
  - Optimized Communication: Compress and Optimize data communication
- Class Loading
  - If possible, avoid to download stubs

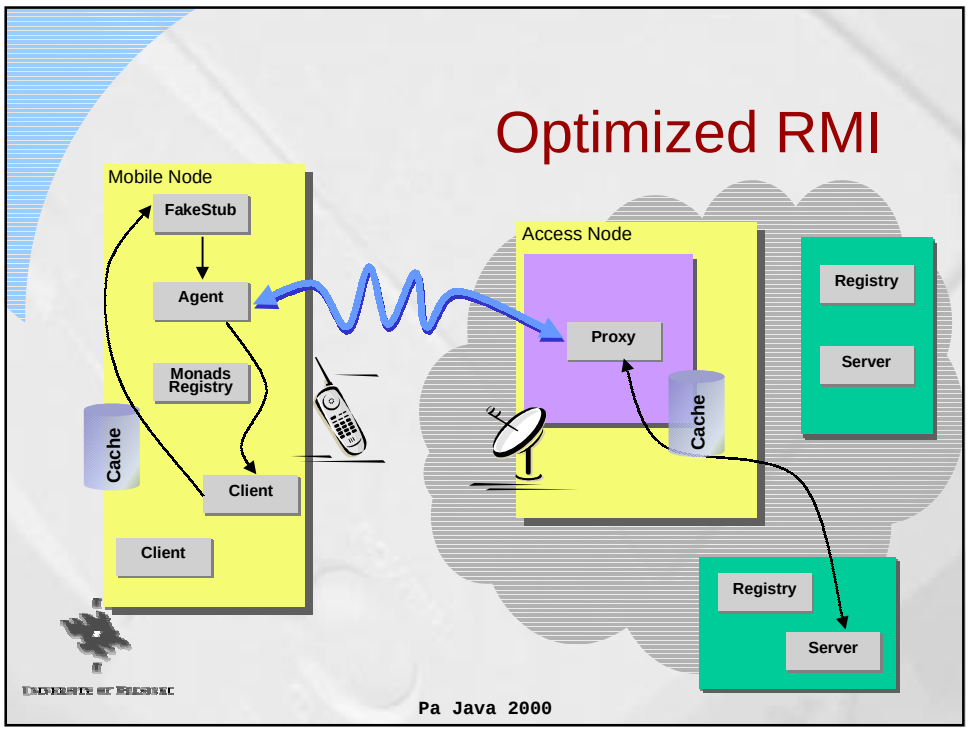
Protocol Optimization

- The idea is to de-couple the connection between the client and the server using mediators.

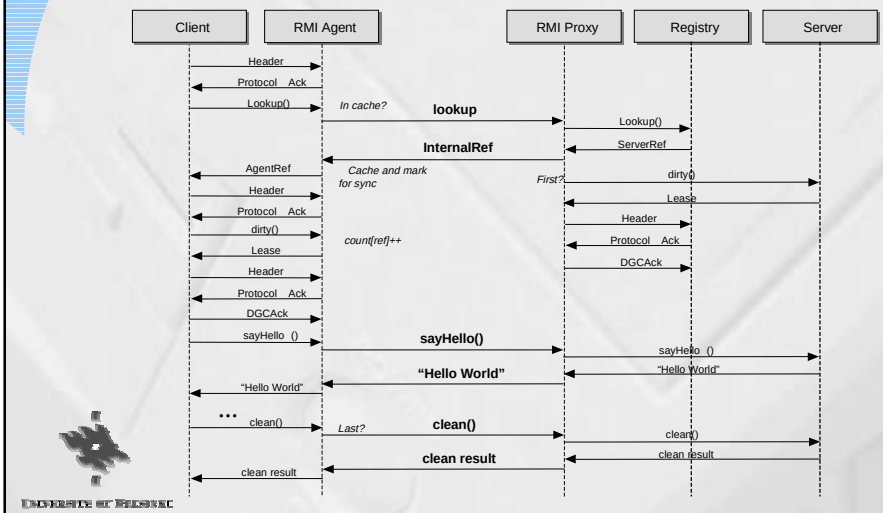





Optimized RMI
Mobile Node
FakeStub
Agent
Monads Registry
Cache
Client
Client
Access Node
Proxy
Cache
Registry
Server
Registry
Server
Pa Java 2000


Optimized RMI
Mobile Node
FakeStub
Agent
Monads Registry
Cache
Client
Client
Access Node
Proxy
Cache
Registry
Server
Registry
Server
Pa Java 2000

Optimized Remote Invocation



Comparison between Normal RMI and Optimized RMI

	Registry Invocation	Remote Invocation	Total
Java RMI	7.1 sec	1.3 sec	8.4 sec
Optimized RMI	1.7 sec	0.6 sec	2.3 sec
Improvement	417%	216%	365%

Normal RMI and Optimized RMI in a Use Case

	Registry Invocation	Remote Invocations		
		3300 bytes	7048 bytes	14760 bytes
Java RMI	18.9 sec	8.1 sec	14.8 sec	41.4 sec
Optimized RMI	2.9 sec	3.0 sec	4.8 sec	5.0 sec
Optimized RMI, no compression	3.0 sec	7.8 sec	13.7 sec	28.1 sec