Evaluating Distributed Functional Languages for Telecommunications Software

J.H. Nyström **P.W. Trinder** **D.J. King**

Heriot-Watt University **Motorola**

January 23, 2002

Overview

- Goals
- Motivation
- Project plan
- Telecoms software characteristics
- High-level language properties
- First test implementation
- What's next

Project goals

Commercial

Convince Motorola that Erlang constitutes a viable technology for implementing distributed telecoms software.

Scientific

Investigate the impact of language constructs on distributed software construction.

Project motivation

Q: Has Erlang not already been shown to be viable telecoms technology by numerous successful applications?

A1: No, it has been shown that companies with deep understanding and experience of building such systems and can successfully use Erlang.

A2: Yes, but what are the properties of Erlang that makes it so suitable for implementing distributed applications?

Project plan: Commercial

- Construct three applications in Erlang of increasing size and realism, i.e., closer to a real product.
- Show that the applications have the essential properties required of a telecoms software with additional benefits.

Project plan: Scientific

- Implement the two first applications in GdH as well as Erlang.
- Compare the Erlang and GdH implementations with existing C++/CORBA and JAVA/RMI.
  - functionality
  - complexity
- Measure impact of language aspects.

Telecoms software characteristics

- Distributed
- Reliable
- Highly available
- Rapid production

Telecoms software challenges

- Fault-tolerant
- Scalable
- Resilient
- Dynamically adaptable

High-level language aspects

- Dynamic typing vs. static typing
- Strict vs. non-strict evaluation
- Explicit vs. implicit messaging and distribution

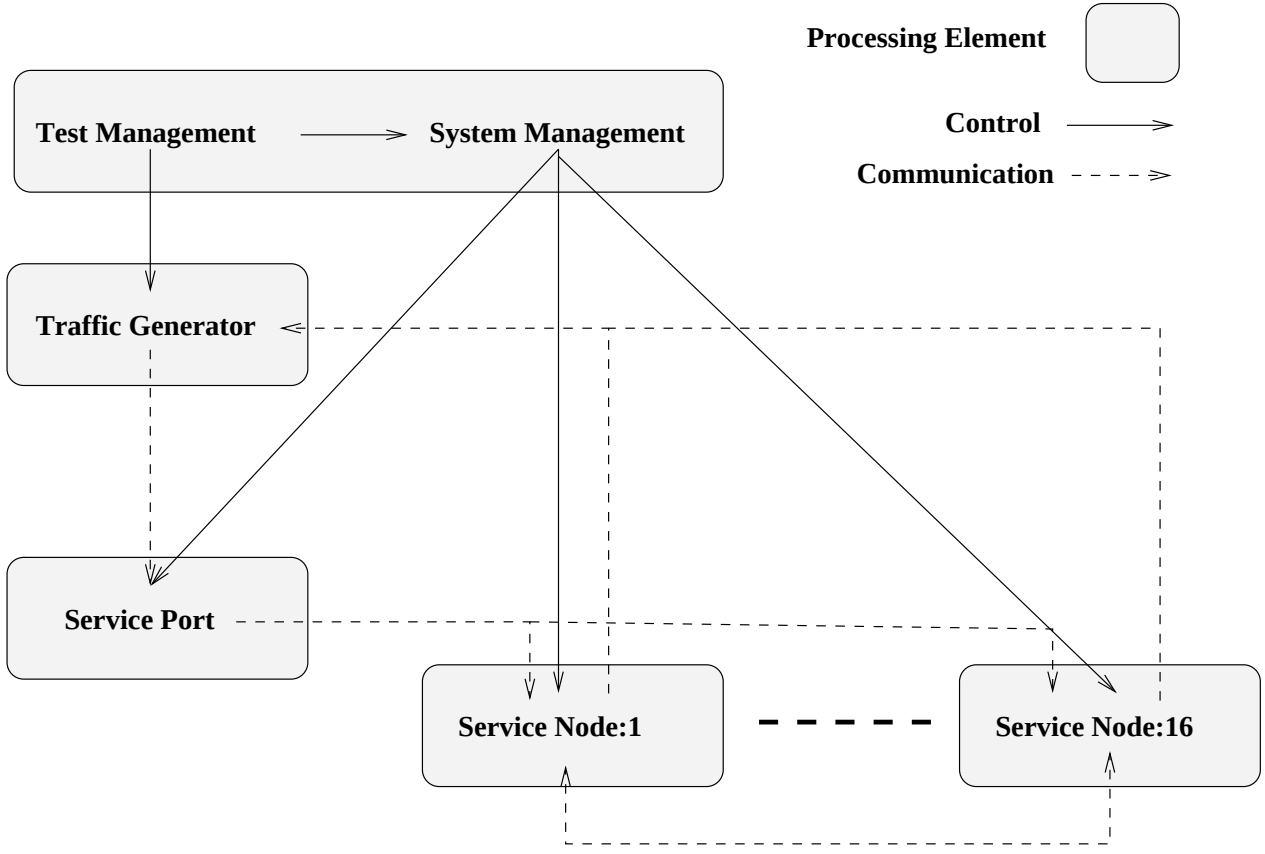
Dispatch Call Controller (DCC)

The server part of dispatch control within a mobile telephone network.

Crucial functionality:

- Dynamic scalability
- Resource reclamation
- Fault tolerance
- Soft real time performance

First test implementation



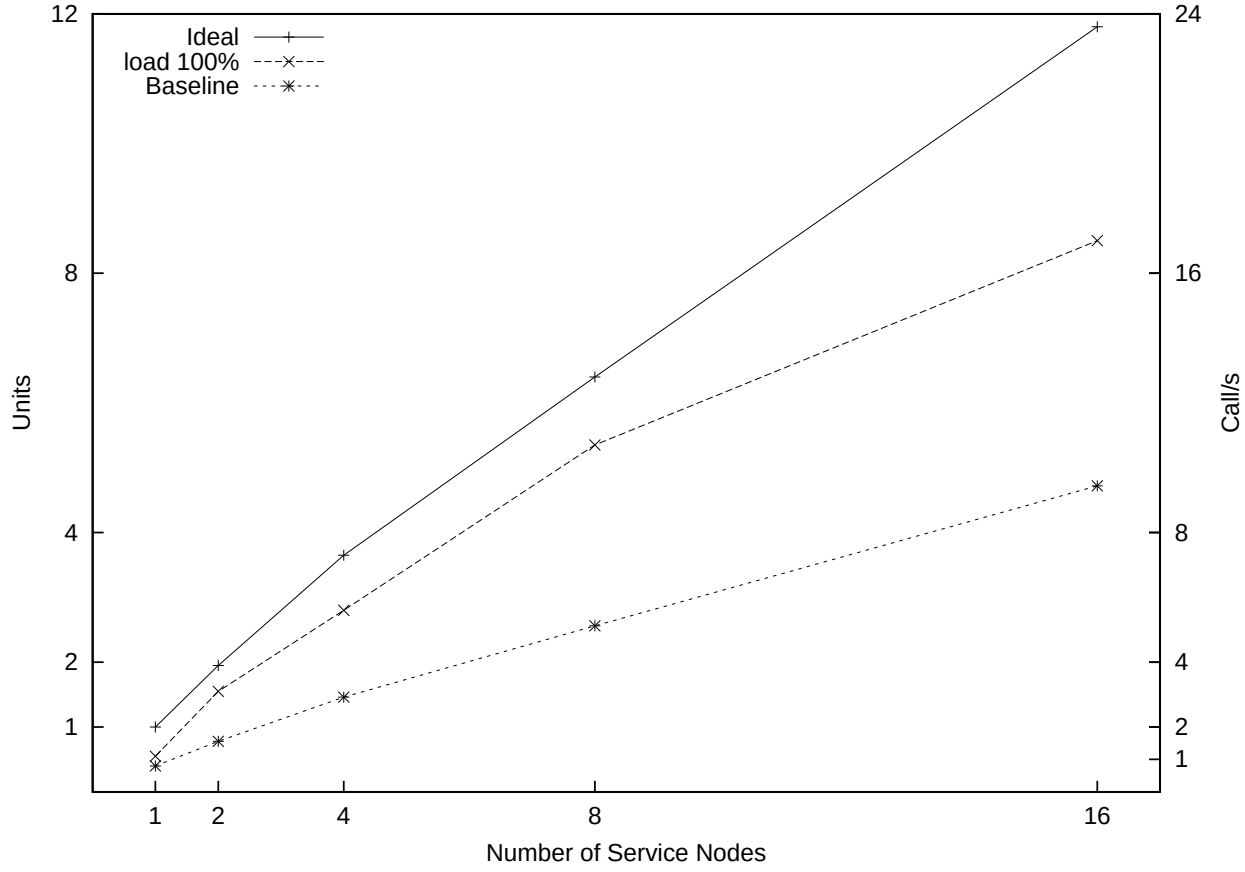
What doesn't the system handle

- Single point of failure.
- No dynamic code upgrade.
- Simplistic service.

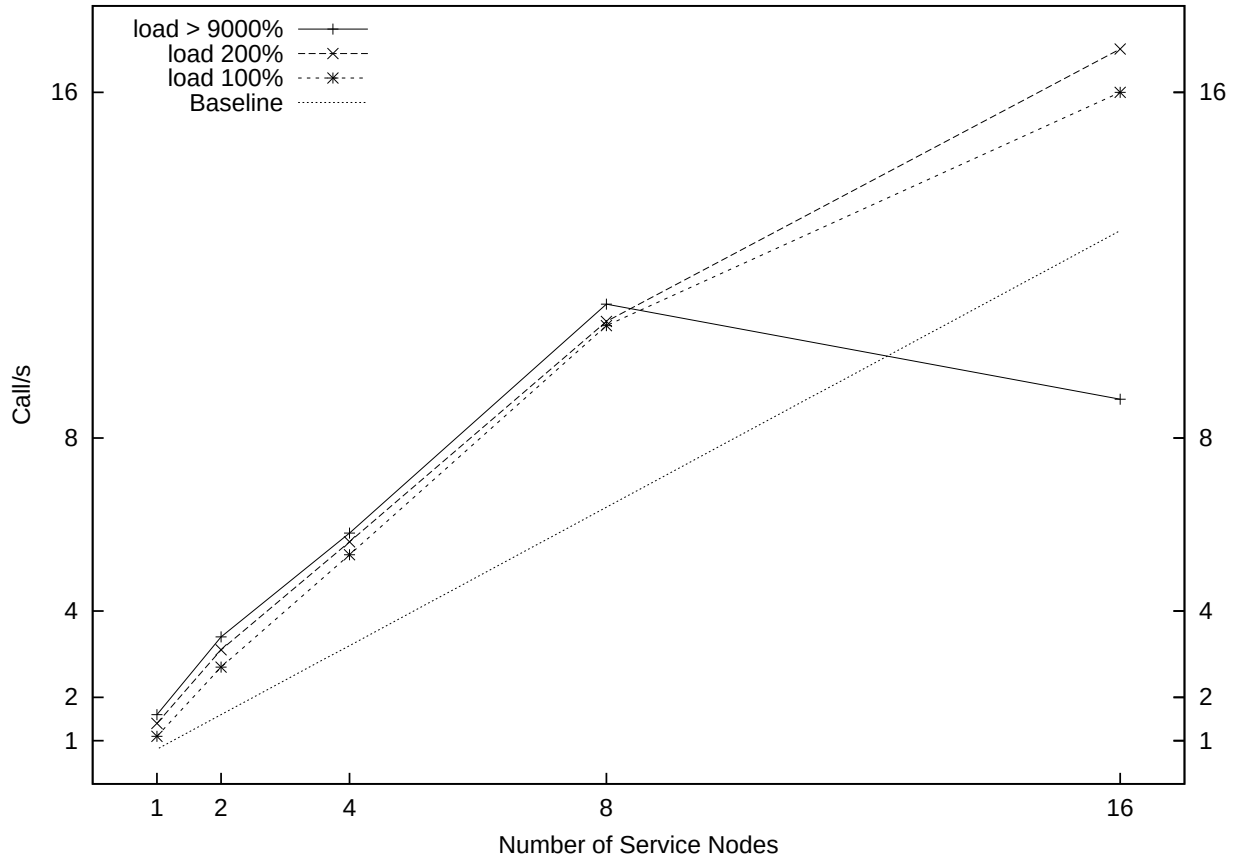
What does the system handle

- Fully fault-tolerant in the service nodes.
- Scales.
- Resilient.
- Service nodes can be added or removed dynamically.

Speedup graph



Resilience graph



What now?

- Remove single point of failure.
- Hot-code upgrade.
- Realistic service.
- Measure performance impact of
  - dynamic reconfiguration
  - fault-tolerance
  - hot-code upgrade

What's next?

Motorola Interaction

- Determine the next application.
- Disseminate results

Scientific

- Reimplement DCC in GdH.
- Comparison with C++/CORBA and JAVA/RMI implementations.

Summary

- New project
- Commercial and Scientific objectives
- First results promising
- *Challenges:* Find the right application and the right people