Divide and Conquer – Scalability and Variability for Adaptive Middleware

Ulrich Scholz [Ulrich.Scholz@eml-d.villa-bosch.de]
European Media Laboratory GmbH

Romain Rouvoy [rouvoy@ifi.uio.no]
Universitetet i Oslo

September 4, 2007

Principles of application adaptation

- Mobile applications have to
  - *Detect changes of*
    - System resources
    - User needs
    - User context
  - *React to changes by*
    - Adapting their structure
    - Configuring the mobile device

How do we model adaptation?

- Principles of compositional adaptation
  - *Each application is realized by different components*
  - *Application can be realized by many combinations of components (variants)*
  - *Each variant has different requirements and provides a different utility to the user*
- When context changes, the current variant might no longer provide optimal utility
- Adaptation algorithm
  - *Find the variant that currently provides the optimal utility*
  - *Change the present variant to this optimal variant*

Example of an Adaptation Algorithm

- Brute Force
  - *Builds an exhaustive list of application variants*
  - *Computes the utility of each variant*
  - *Selects the variant with the optimal utility*
  - *Replaces the current variant by the selected one*

Limitations of the Brute Force Algorithm

- Brute Force does not scale
  - *Number of application variants is exponential with regards to the number of components composing the application*
- Brute Force performs a global adaptation
  - *Each adaptation considers all the applications*
    - Adaptation of independent applications/users is not handled independently
    - Might lead to unnecessary adaptations
  - *Single point of failure*
    - No fault tolerance support due to the use of a centric approach

Objectives of Divide and Conquer (DnC)

- Scalability support
  - *Large numbers of applications and users*
- Distributed adaptation
  - *Adapt only parts for which it is really necessary*
  - *Exploit (almost) independence*
- Adaptation variability *wrt.* changes in the topology
  - *Allows (dis)appearance of users, machines, applications, and connectivity*
- Alternative adaptation mechanisms support
  - *Provide a variety of adaptation mechanisms and select the most appropriate one*

Core Concepts of DnC

- Decomposition tree
  - *Distributed organisation of the adaptation*
    - Based on a distributed middleware, *e.g.*, JXTA
  - *Small set of predefined reconfiguration primitives*
    - Primitives are combined to reconfiguration strategies

- Negotiation node
  - *DnC expoits "almost independence" of applications/users*
  - *Negotiation nodes prevent combined adaptations in **some cases** of **limited dependence***

Decomposition Tree Overview

- Set of all applications/components gets recursively divided
  - *Until remaining sets are small enough*
    - These can be solved by, *e.g.*, Brute Force
  - *Decomposition tree represents this process*
    - Tree is distributed along with the division
    - Nodes require only limited and sporadic reasoning

- Tree is updated according to changes in the applications and in the topology

Decomposition Tree Operations (Examples)

- **Join** two decomposition trees
  - *Triggered by network connection*
  - *Defines a new common root node*
- **Split** a decomposition tree
  - *Triggered by network disconnection*
- **Change** the decomposition
  - *If the current one is not sufficient*
  - *Guided by information provided by the application*

Negotiation Nodes (NN) Overview

- Encapsulate interactions between components
  - *Each component provides a utility function*
    - maps the offered properties to a user utility ∪ {reject}
    - underestimates utility
    - depends only on one resource (or on few of them)
- Utilities of components are combined by NN
- In case of context change
  - *Component provides a new utility function*
  - *NN re-calculates distribution*
  - *If necessary, adaptation is initiated*
  - *If no acceptable distribution is found: NN fails*
- Prevent full-scale adaptation in some cases

Example of Negotiation Node

- Distribution of 1GB of main memory
  - *Between a math application & an Internet radio*
  - *Context changes: Math application needs more and more memory*

$$utility_{radio}(memory) = \begin{cases} 1.0 & \text{if } 150\,kByte \le memory \\ 0.5 & \text{if } 50kByte \le memory < 150kByte \\ reject & \text{if } memory < 50kByte \end{cases}$$

$$utility^{1}_{math}(memory) = \begin{cases} 1.0 & \text{if } 400\,kByte \le memory \\ reject & \text{if } memory < 400kByte \end{cases}$$

$$utility^{2}_{math}(memory) = \begin{cases} 1.0 & \text{if } 900\,kByte \le memory \\ reject & \text{if } memory < 900kByte \end{cases}$$

$$utility^{3}_{math}(memory) = \begin{cases} 1.0 & \text{if } 2\,GByte \le memory \\ 0.5 & \text{if } 1\,GByte \le memory < 2GByte \\ reject & \text{if } memory < 1GByte \end{cases}$$

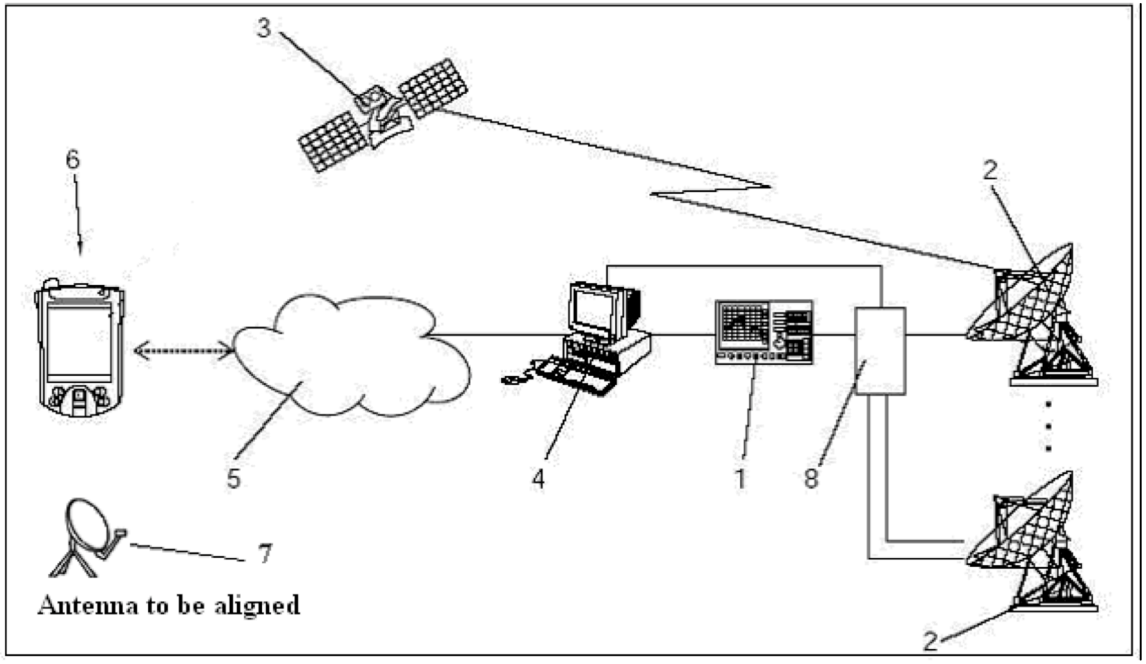
DnC Assumptions

- DnC is based on several assumptions
  - *Aim is **locally "good"** adaptation, not globally optimal one*
  - *There is a lot of almost **independence** to exploit*
  - ***Interaction is limited and easy to capture by simple utility functions***
  - *Topology **changes "slowly"***
  - *Global adaptation method is available as fallback*

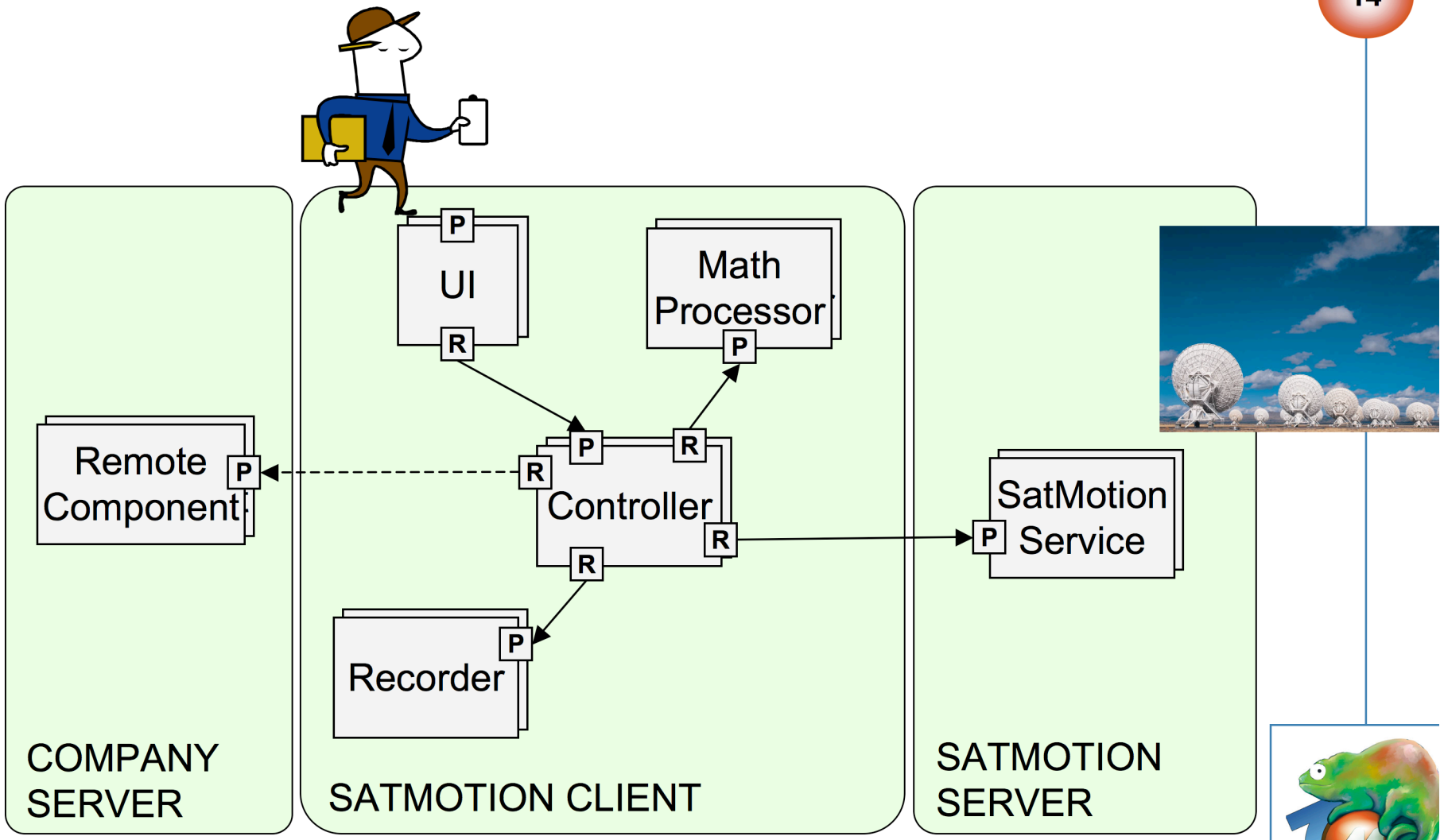
- Assumptions allow a calm and locally confined adaptation

Illustration on the SatMotion scenario

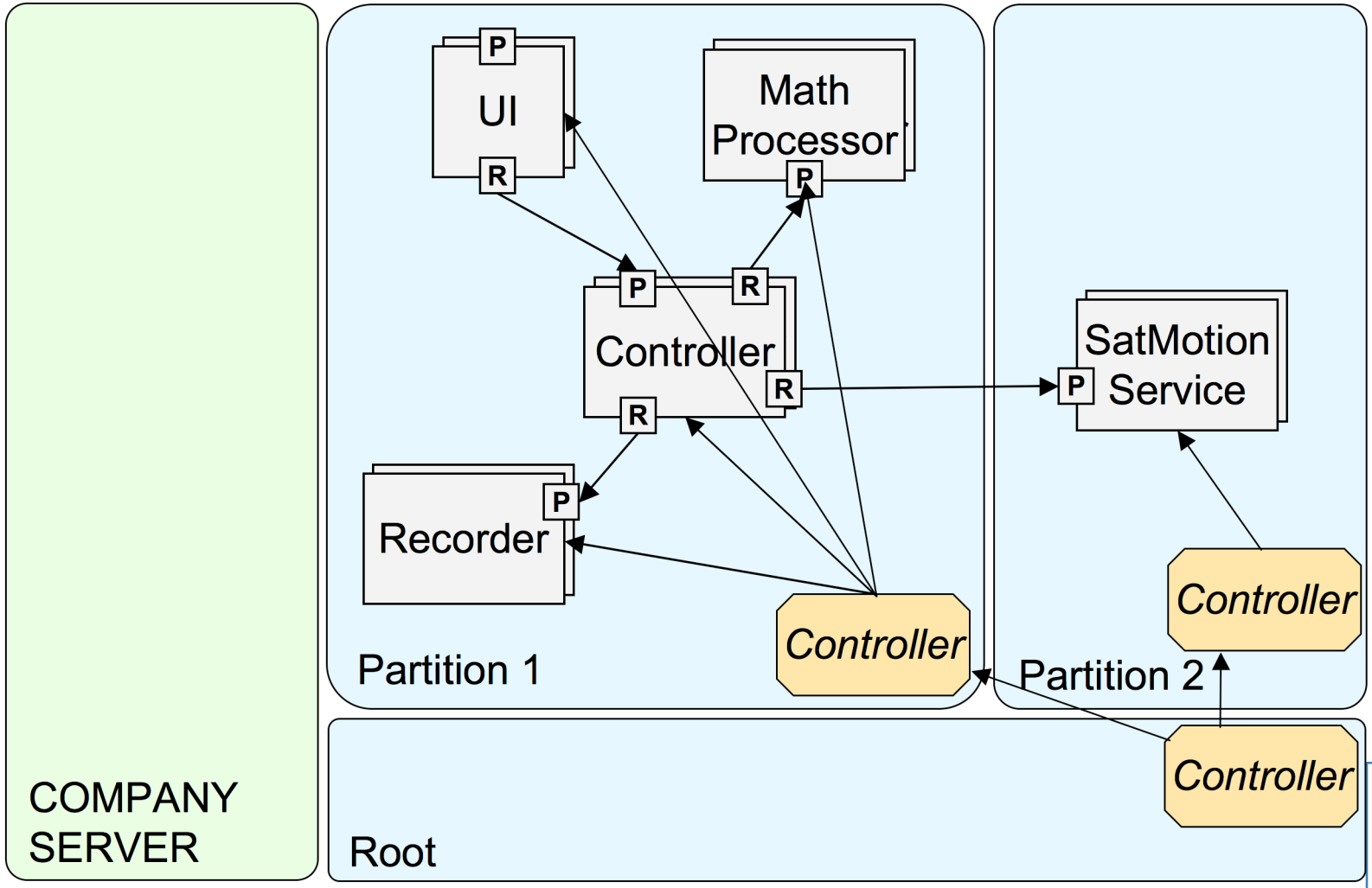
- Application controlling and commanding a measurement instrument from a wireless handheld terminal



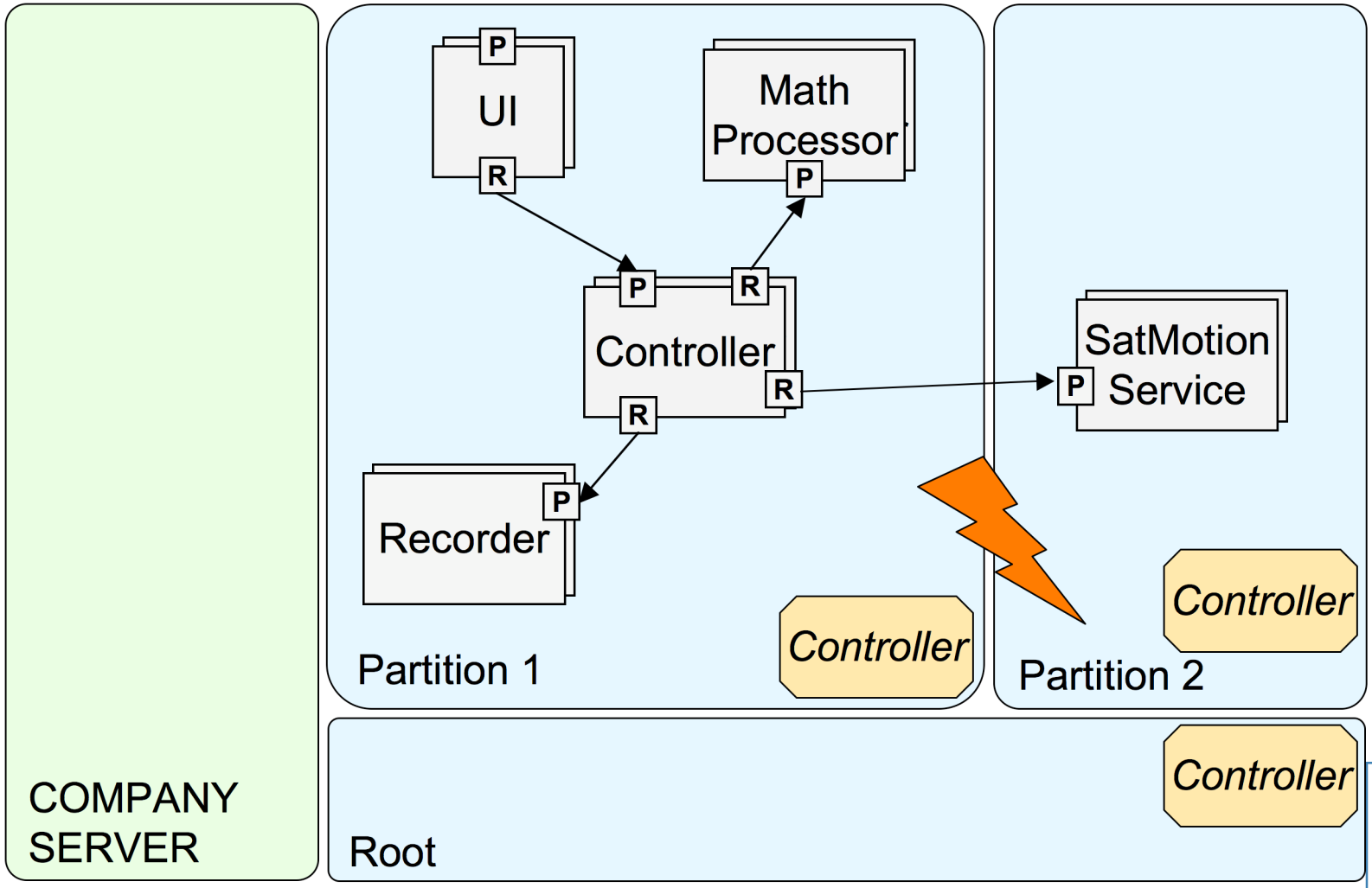
Overview of SatMotion architecture



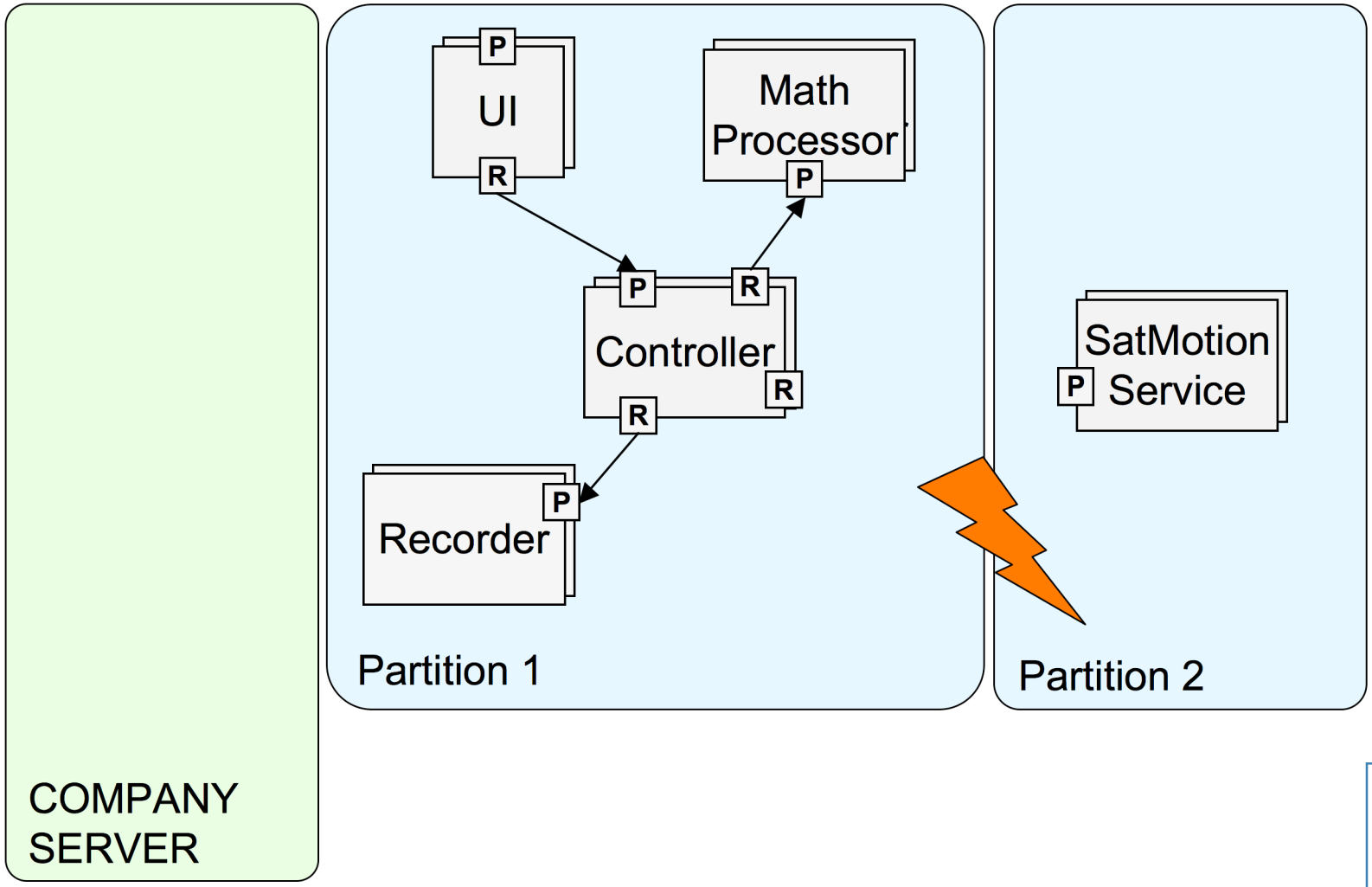
Initial Decomposition Tree



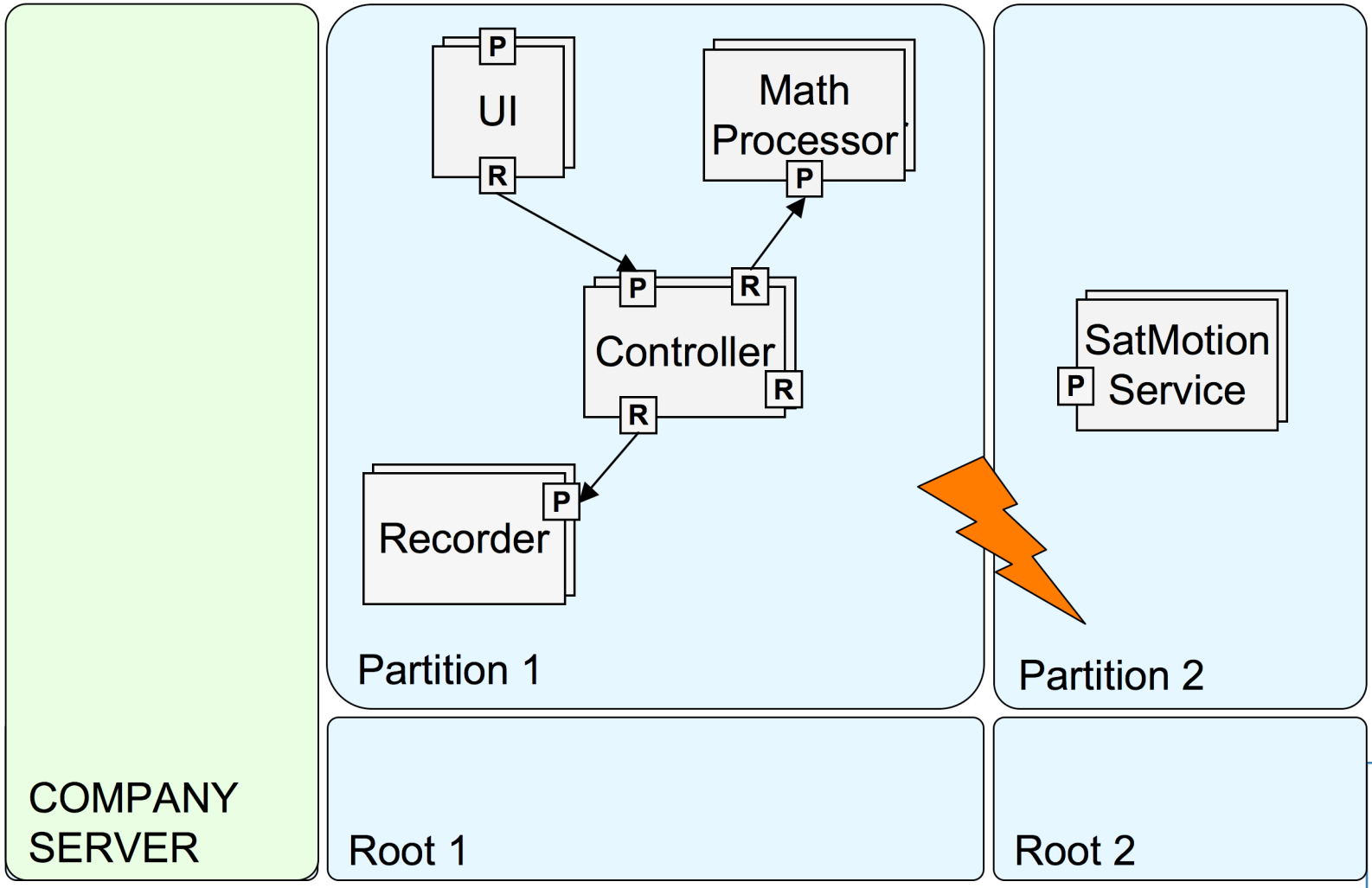
Context Change: Network failure



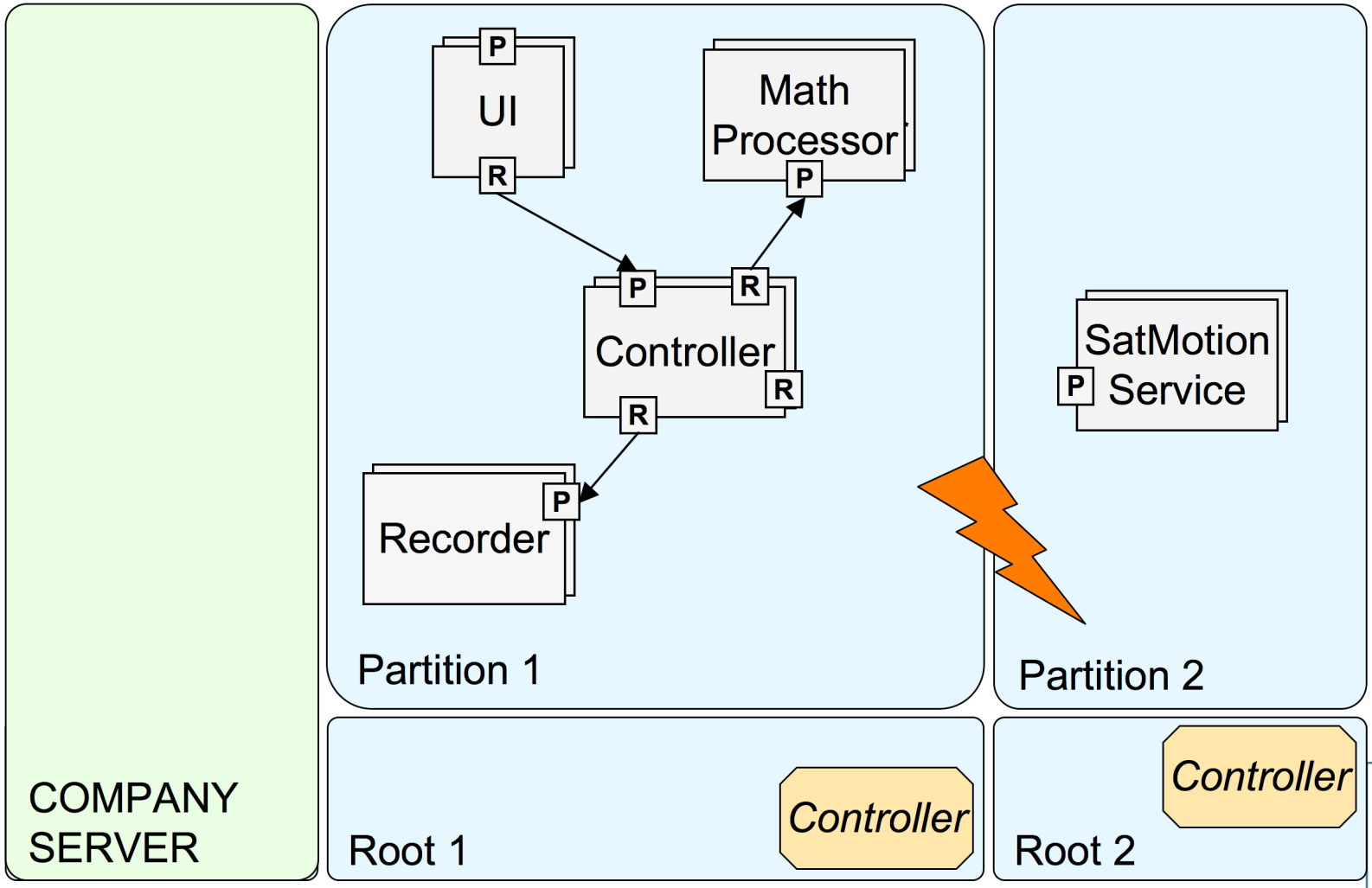
Context Change: Network failure



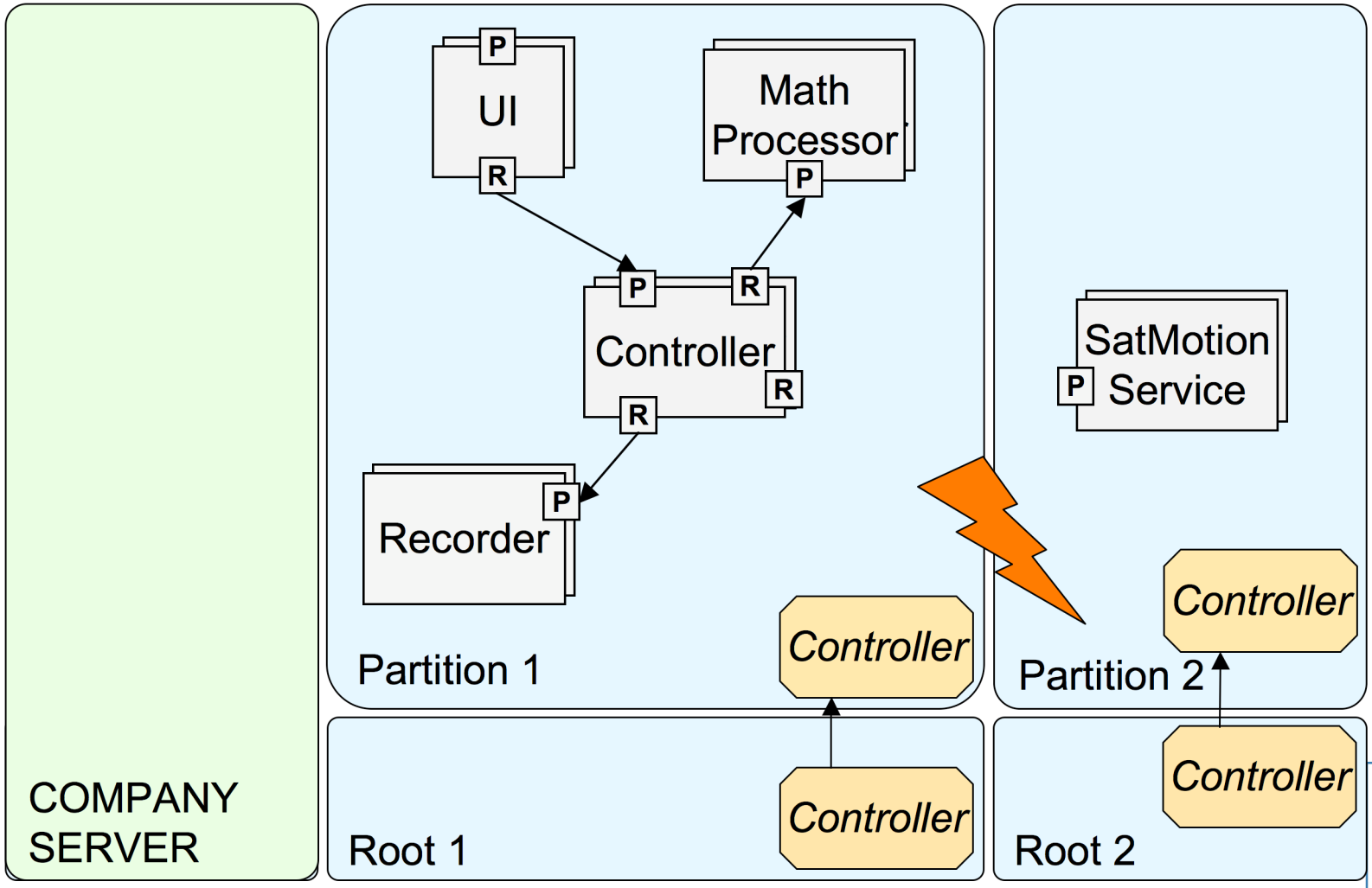
Context Change: Network failure



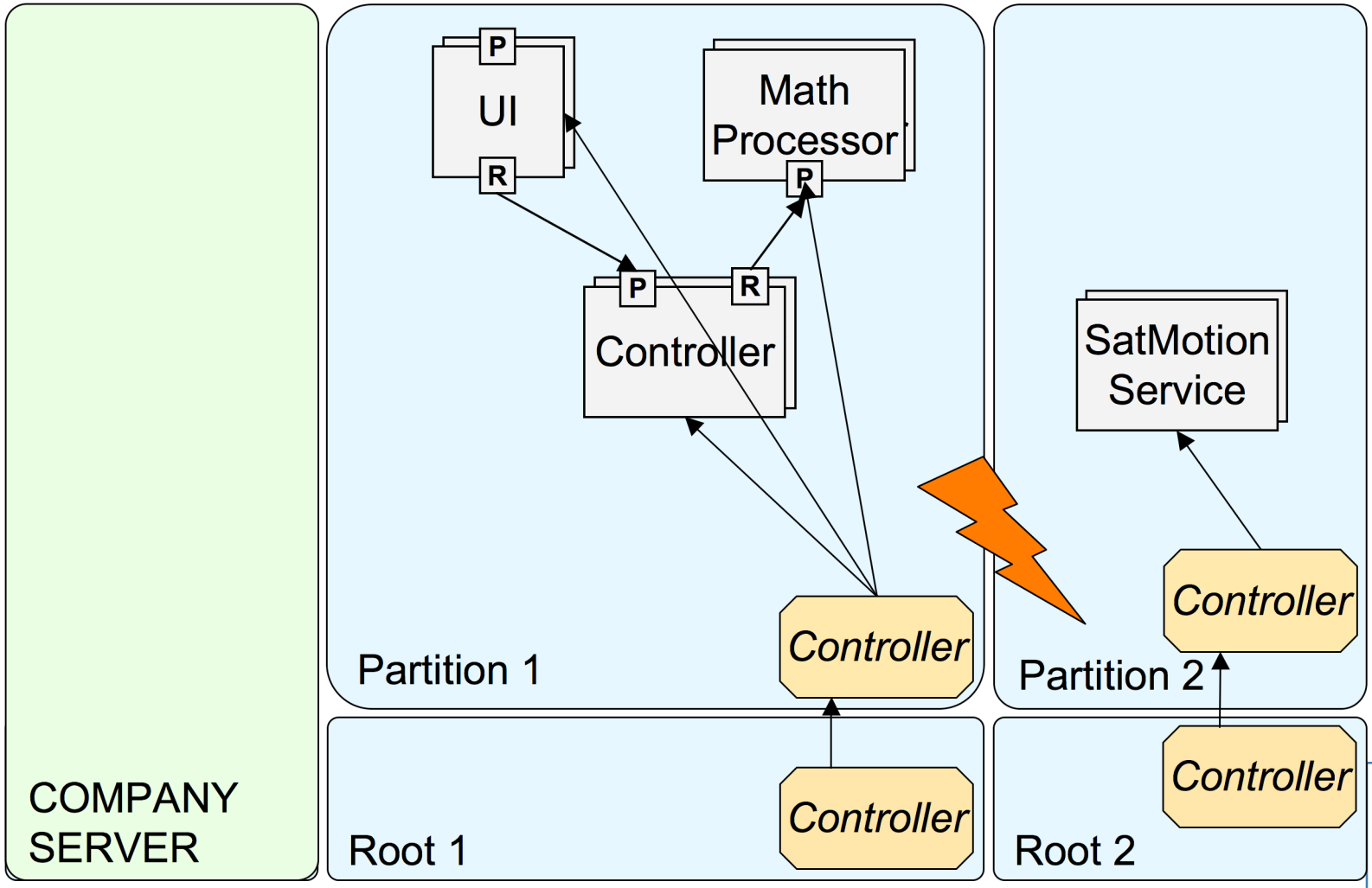
Context Change: Network failure



Context Change: Network failure



Context Change: Network failure



16

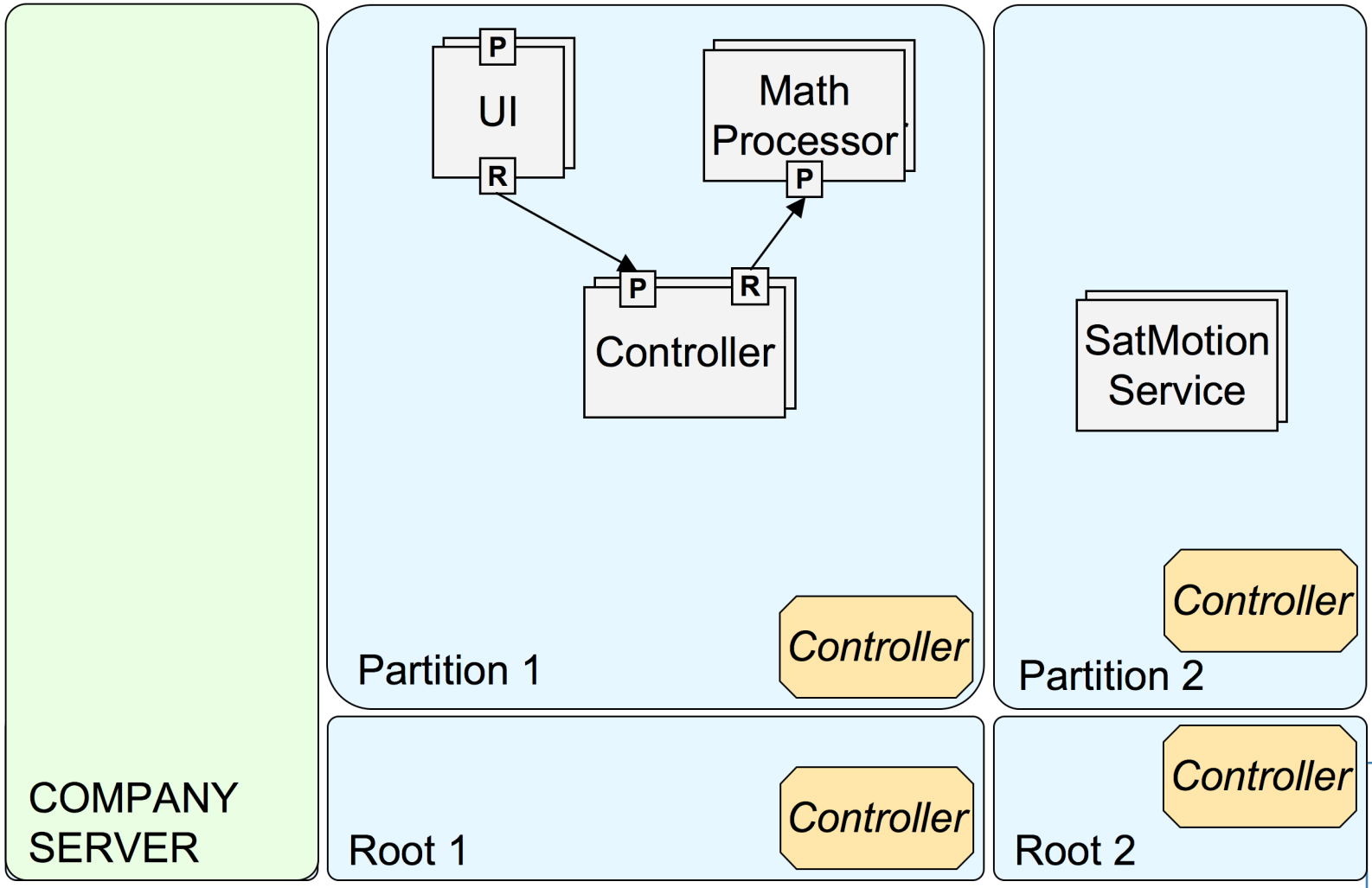
Split

Update

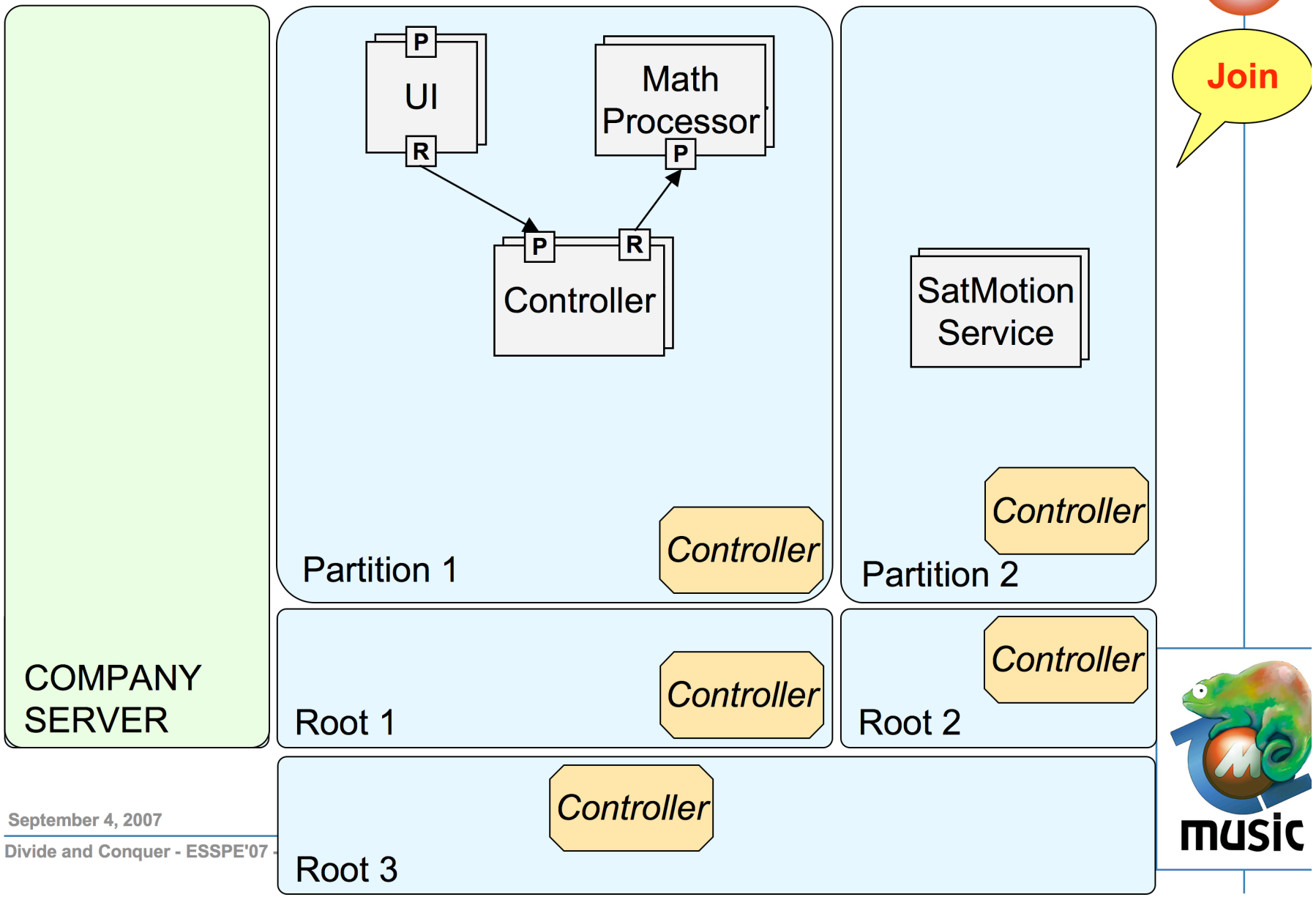
Select

Adapt

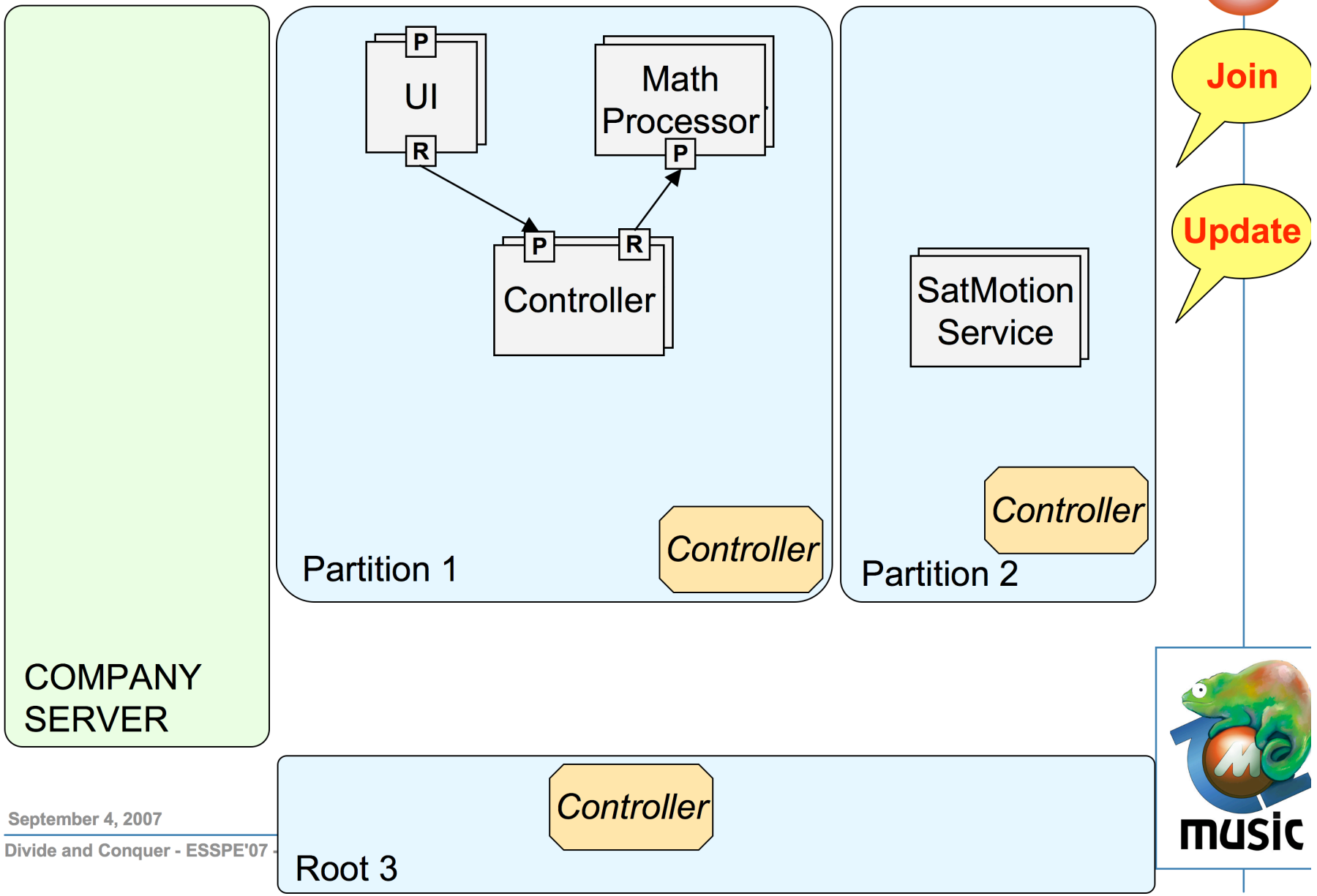
Context Change: Network Up Again



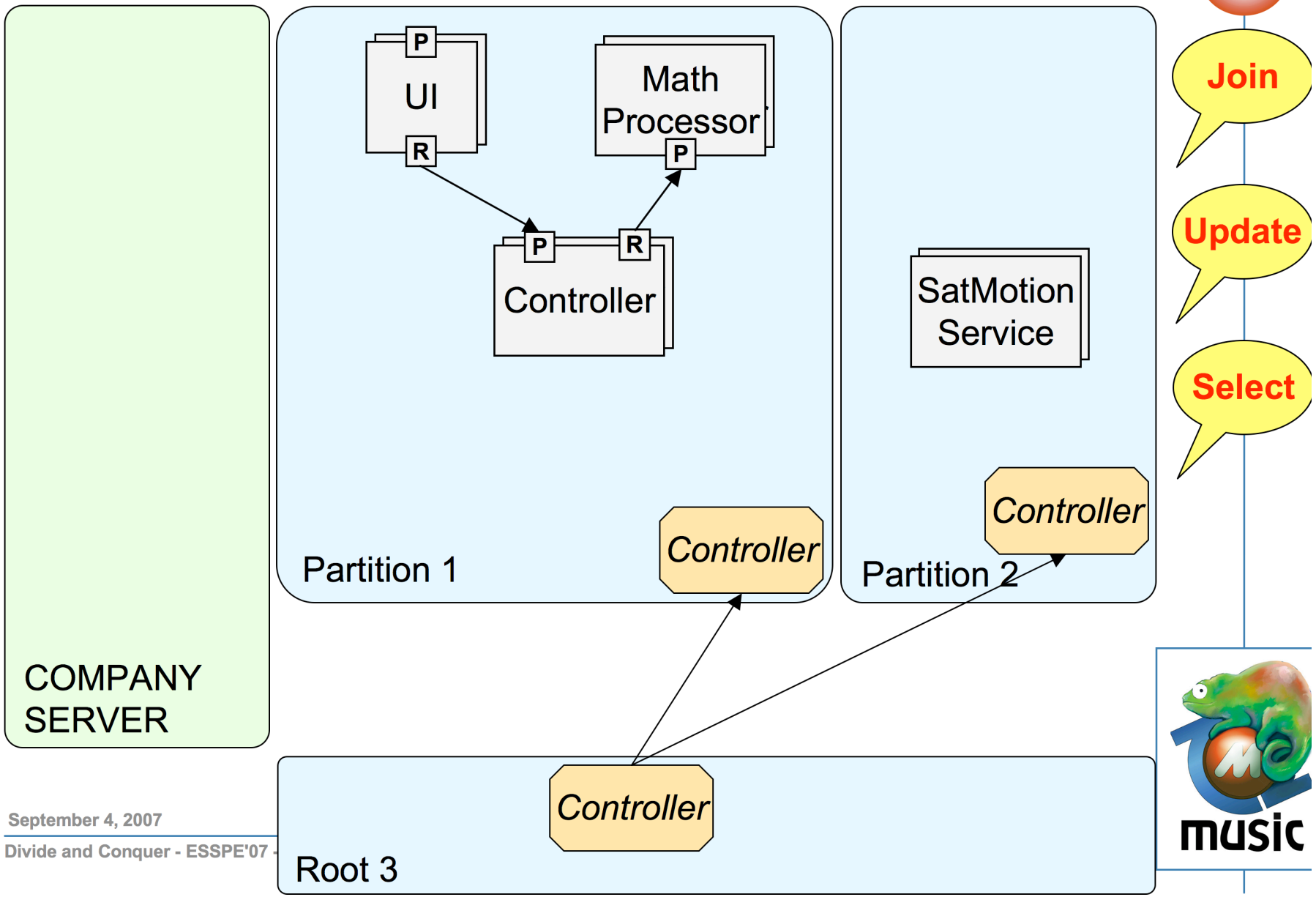
Context Change: Network Up Again



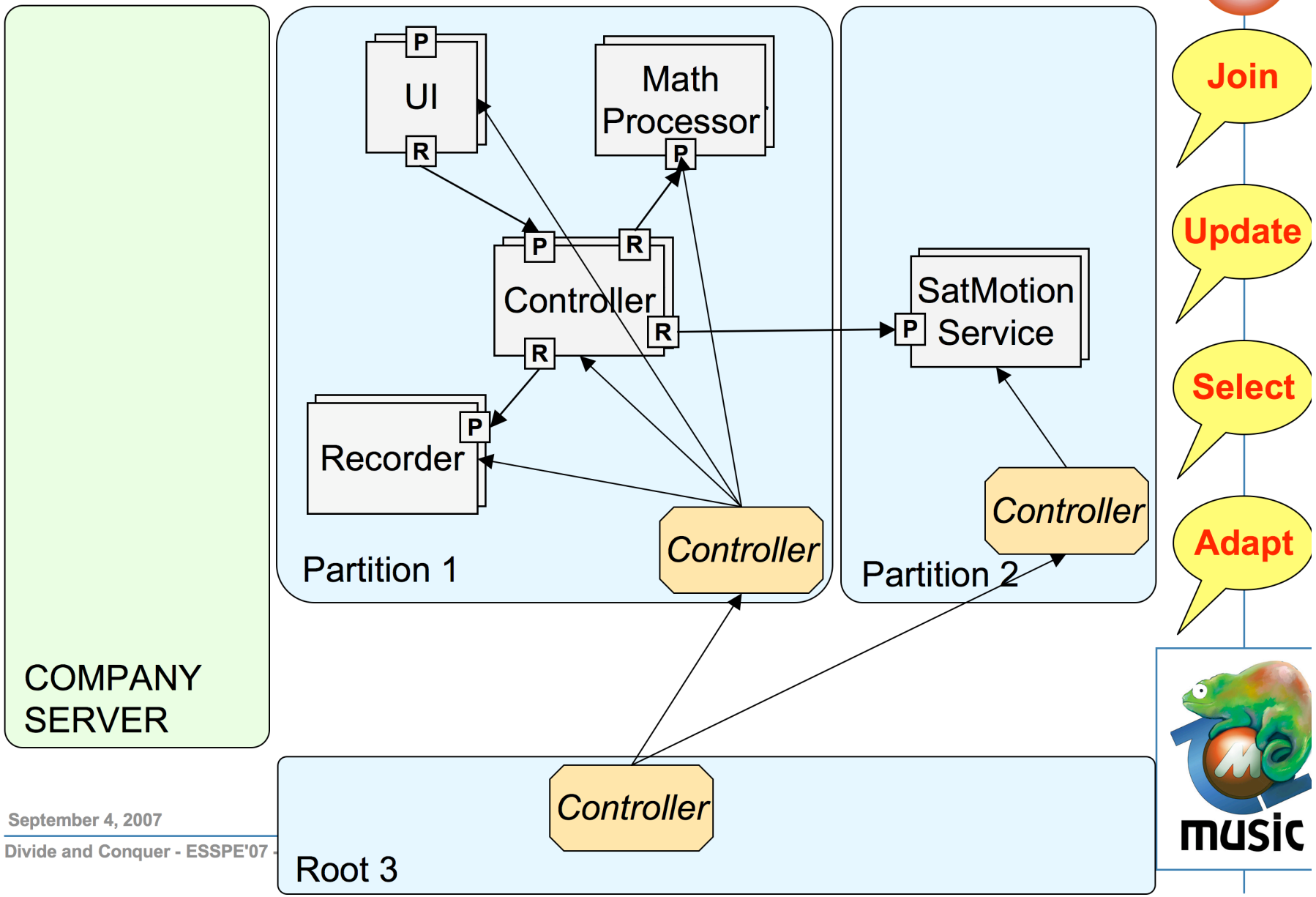
Context Change: Network Up Again



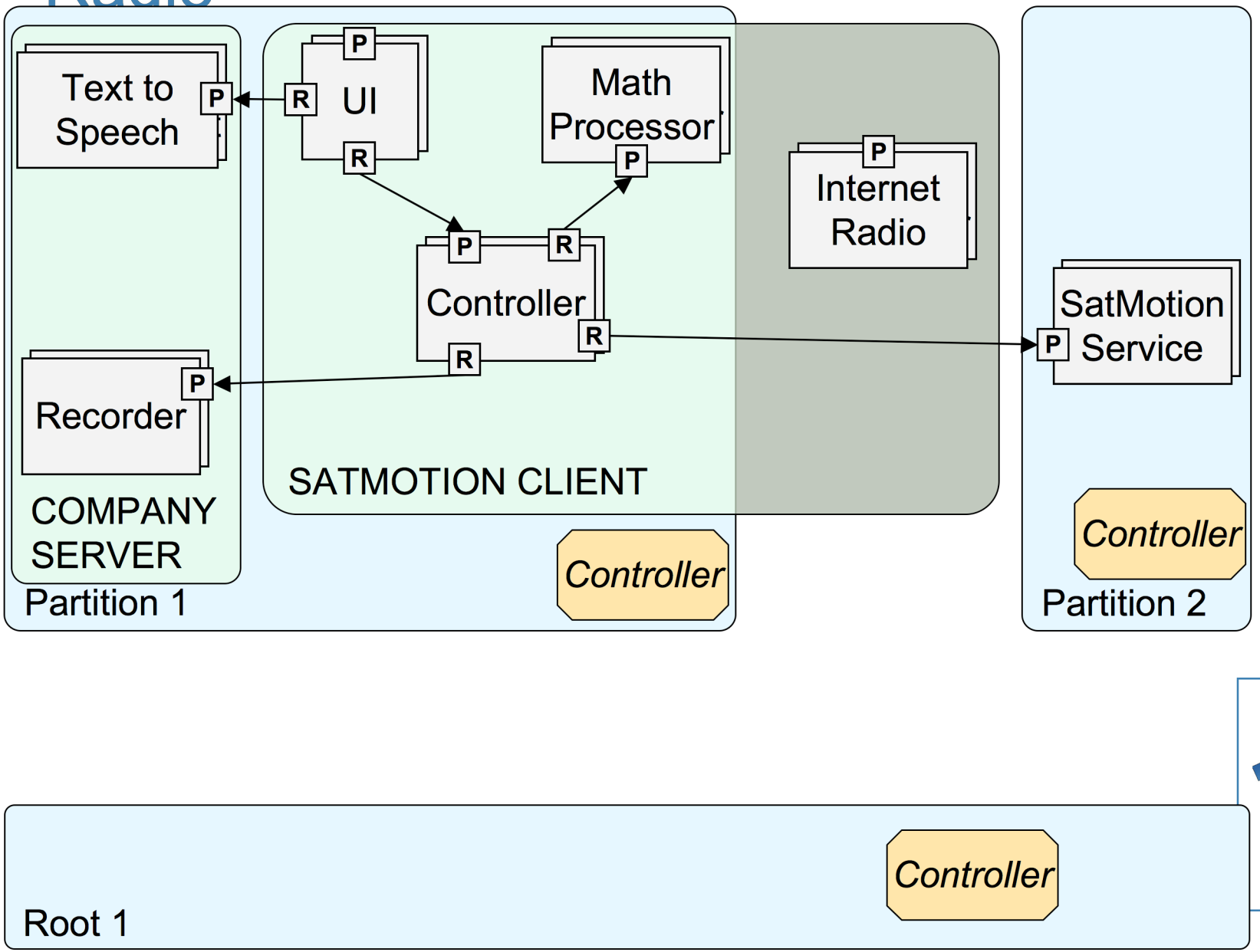
Context Change: Network Up Again



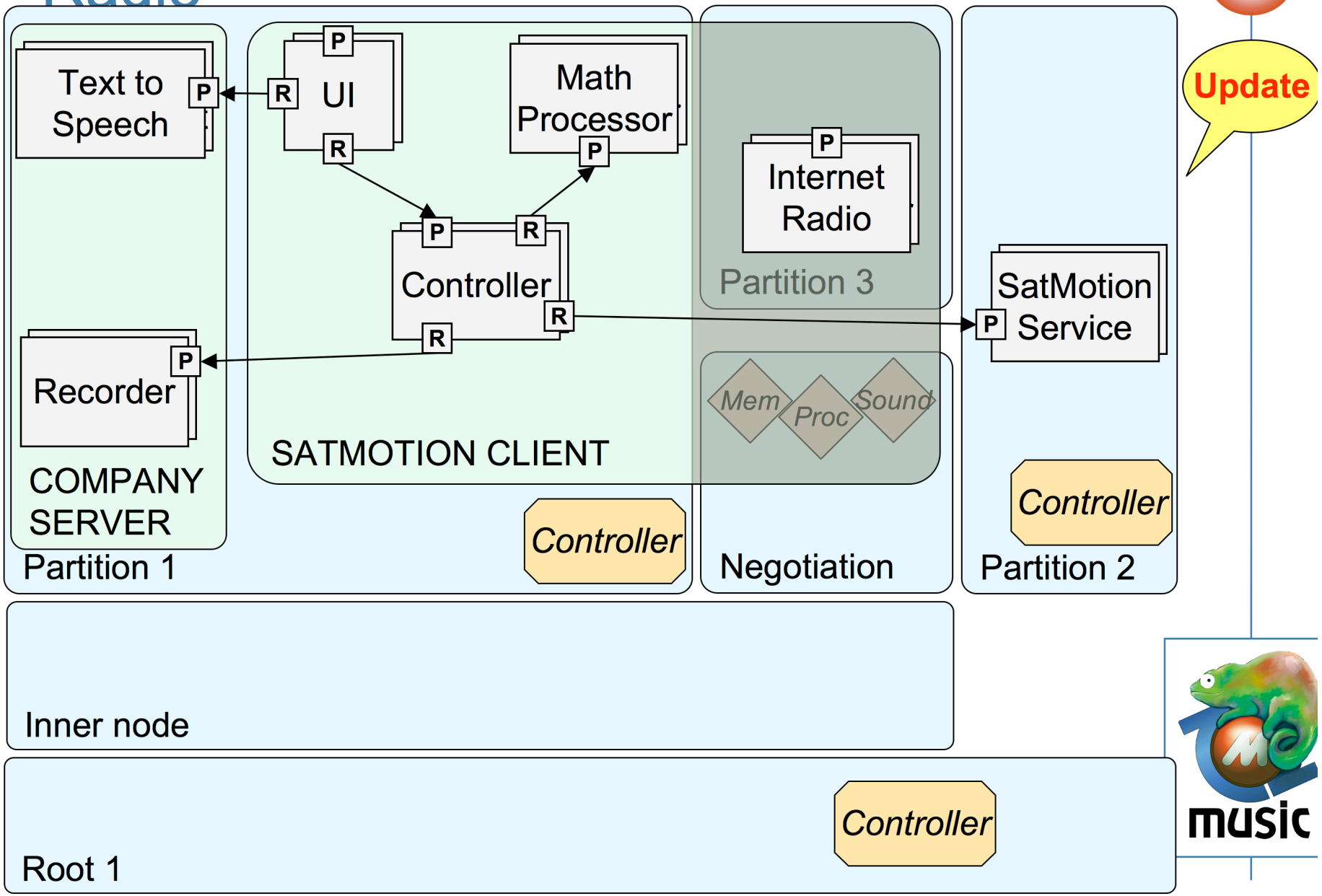
Context Change: Network Up Again



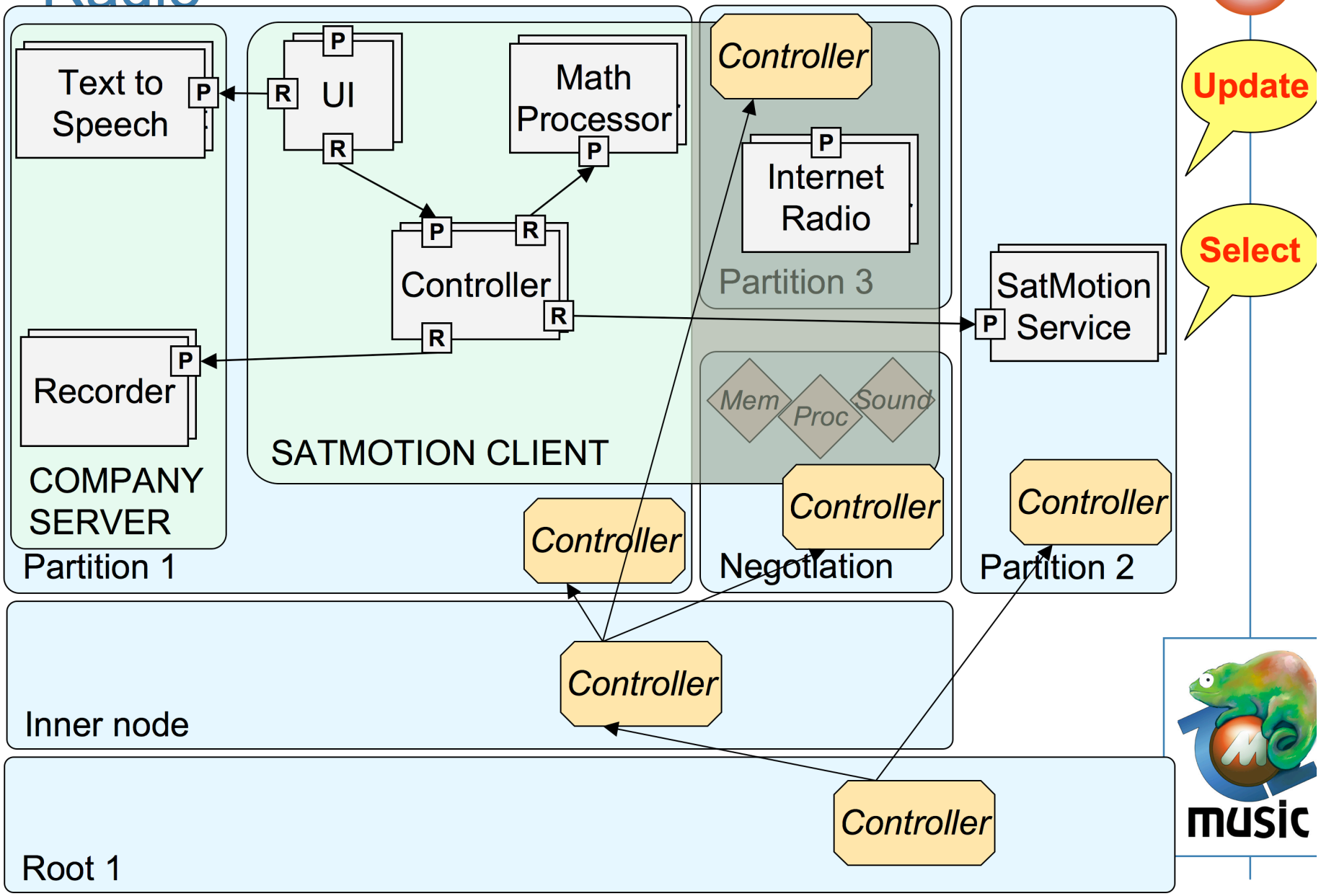
Context Change: Starting an Internet Radio



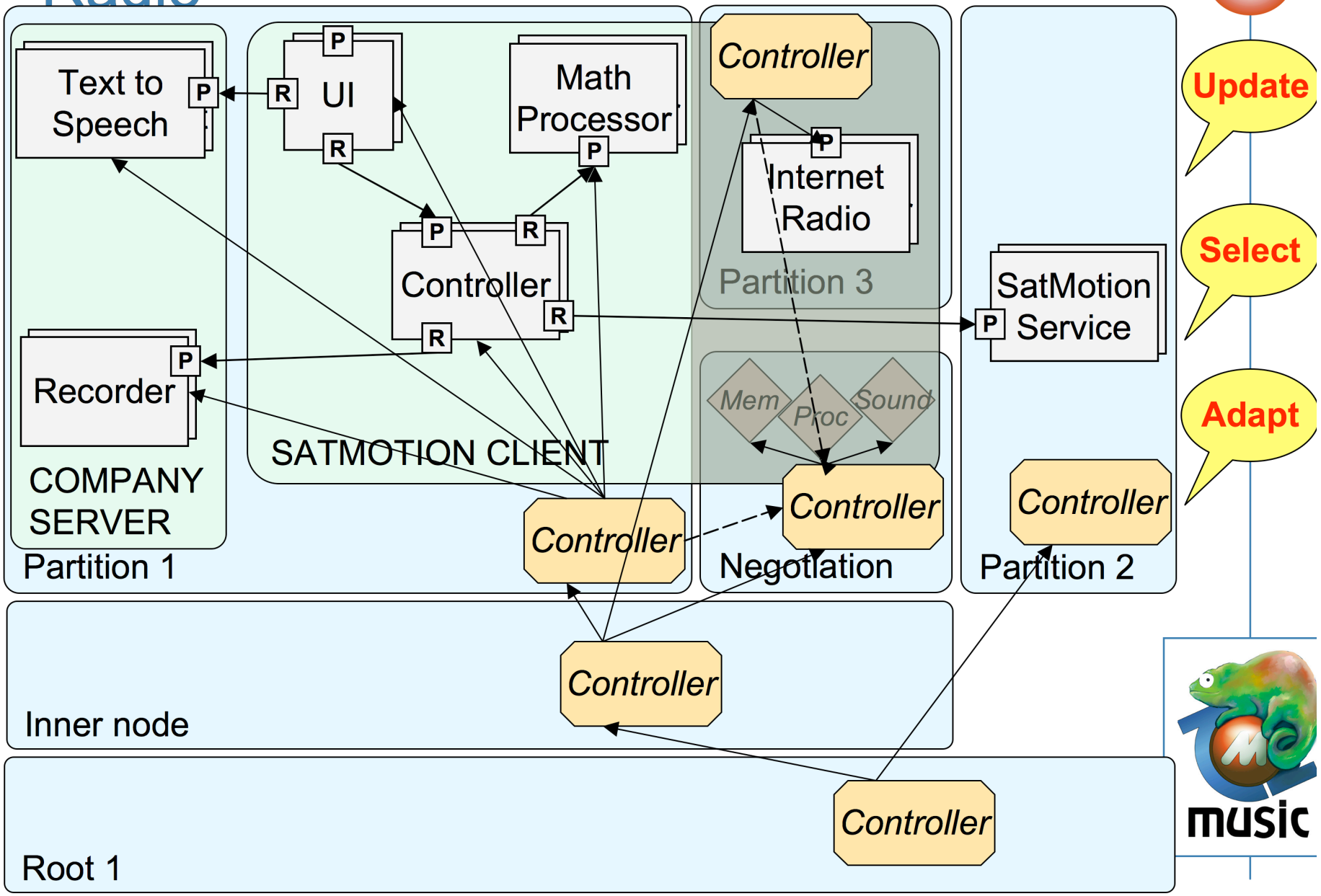
Context Change: Starting an Internet Radio



Context Change: Starting an Internet Radio



Context Change: Starting an Internet Radio



Thanks!

Questions?

EUROPEAN MEDIA LABORATORY GMBH