# Middleware for Gossip Protocols

Michael Chow and Robbert van Renesse
Cornell University

# Motivation

- Gossip protocols are highly robust
- Problematic when an error does occur
  - E.g. Amazon S3 – 6 hours to fix an otherwise simple problem
  - Want to offer a way to fix such problems without having to take down the entire system

## Contributions

Design, implementation, and analysis of gossip middleware that supports rapid code updating

## Talk Outline

- Versions and Deployments
- Architecture
- Evaluation
- Conclusion and Future Work

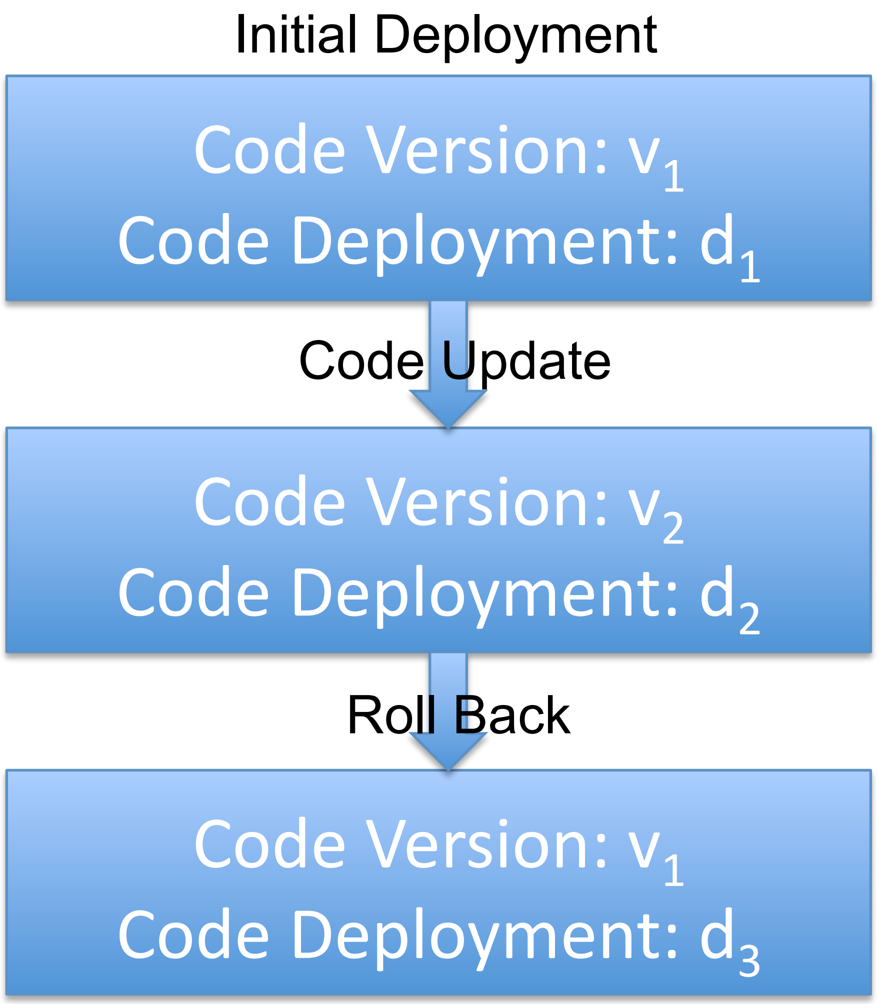
## Talk Outline

- Versions and Deployments
- Architecture
- Evaluation
- Conclusion and Future Work

# Versions and Deployments

- *Modules*: Gossip application instances
- Each module assigned a *Deployment Number*
  - Identifies originating node and time of deployment
  - Used to determine whether or not nodes are running the correct version of the application
  - Does not correspond with code version

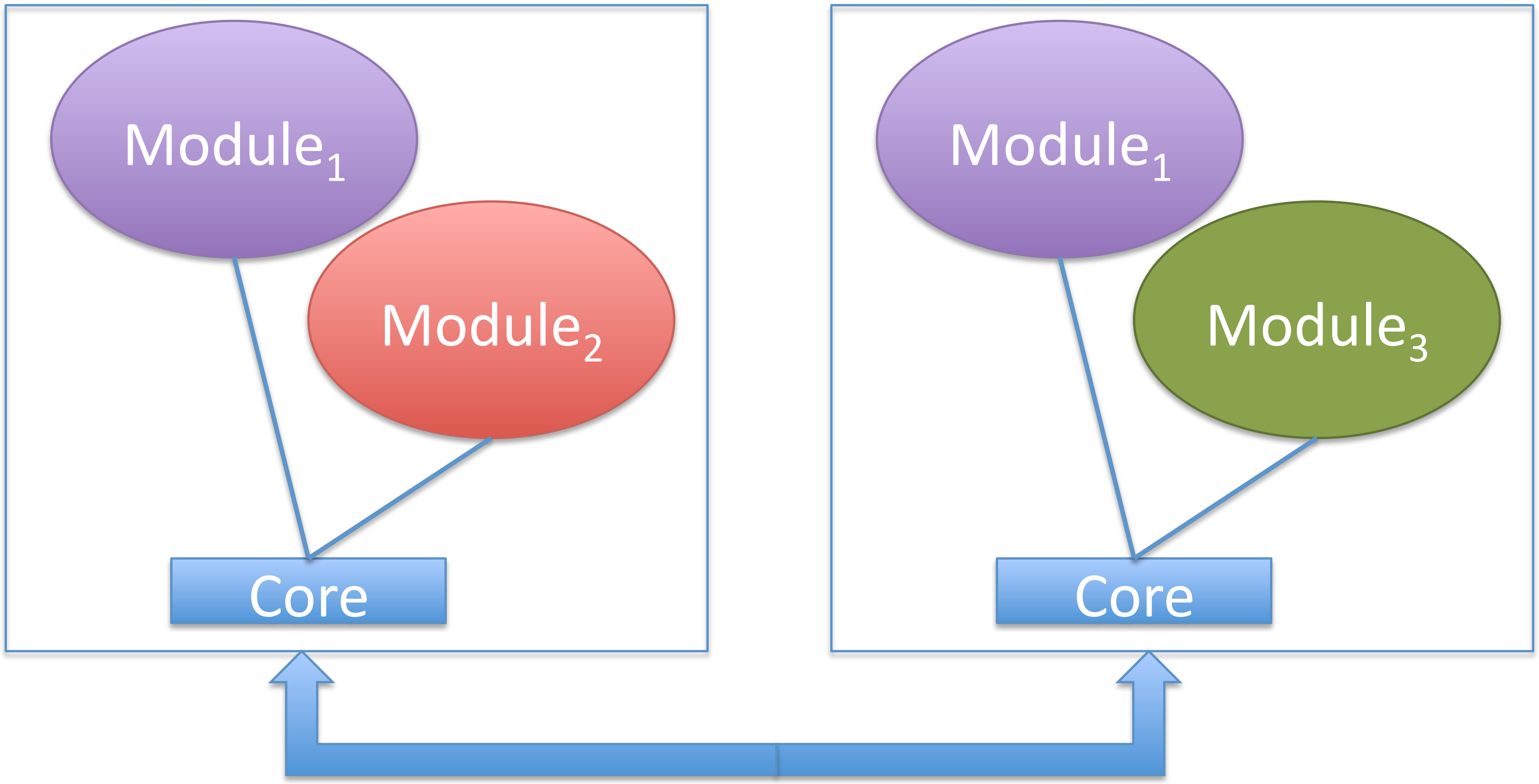
# Versions and Deployments



## Talk Outline

- Code Updating
- Architecture
- Evaluation
- Conclusion and Future Work

# Architecture



## Core

- Provides Module Management and Updating
- Core gossips deployment numbers and corresponding code versions
- Core itself cannot be updated this way
- *Challenge: keep core small*
- Approach: core leverages ongoing gossip between modules

## Module Management

- Core maintains a configuration file that contains:
  - List of Modules and current versions (identified by hash codes of the class files)
  - Deployment Number
- Keeps track of which modules and corresponding versions are currently running
- Cores gossip Configuration files

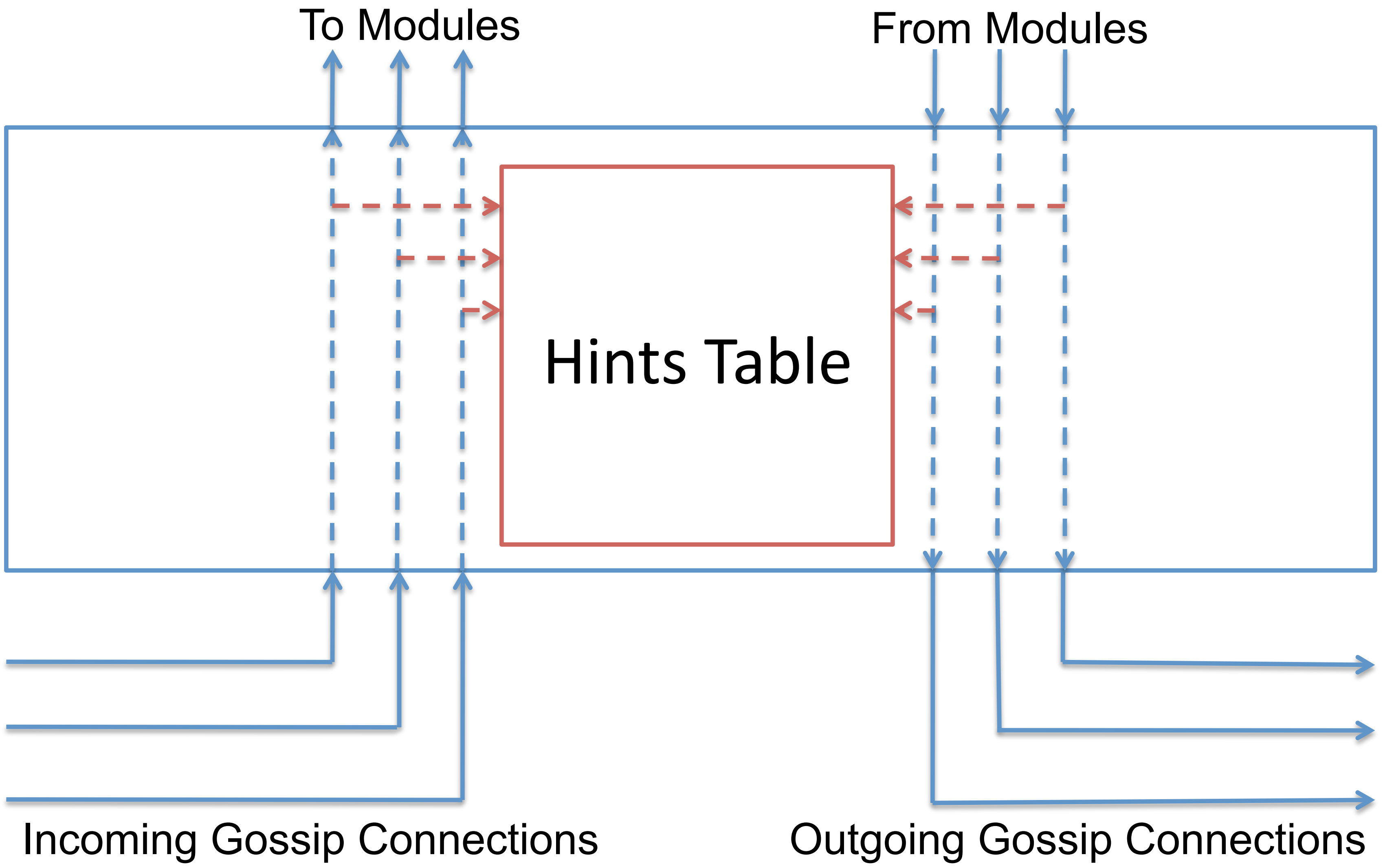
## Gossip Mediation

- Core mediates gossip between modules
- Two advantages
  1. Core piggybacks module deployment number on existing gossip traffic which keeps core simple
  2. Core uses HTTP to minimize problems with firewalls

## Backup Gossip

- Cores need to be able to update code even if all modules have failed
- Cores implement a rudimentary but robust gossip protocol
  - Static list of rendezvous nodes
  - Intercepted membership hints from module gossip

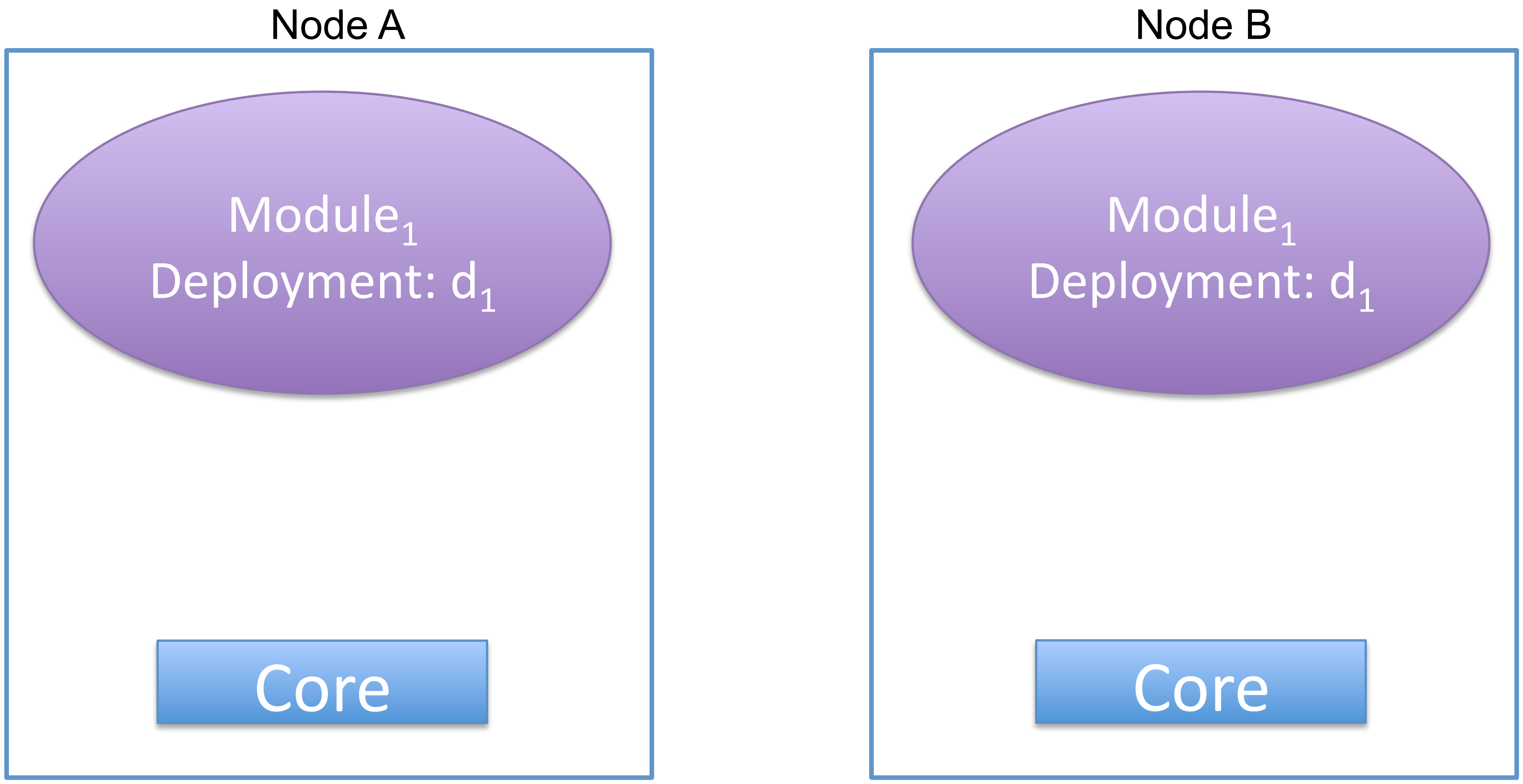
# Core


To Modules
From Modules
Hints Table
Incoming Gossip Connections
Outgoing Gossip Connections

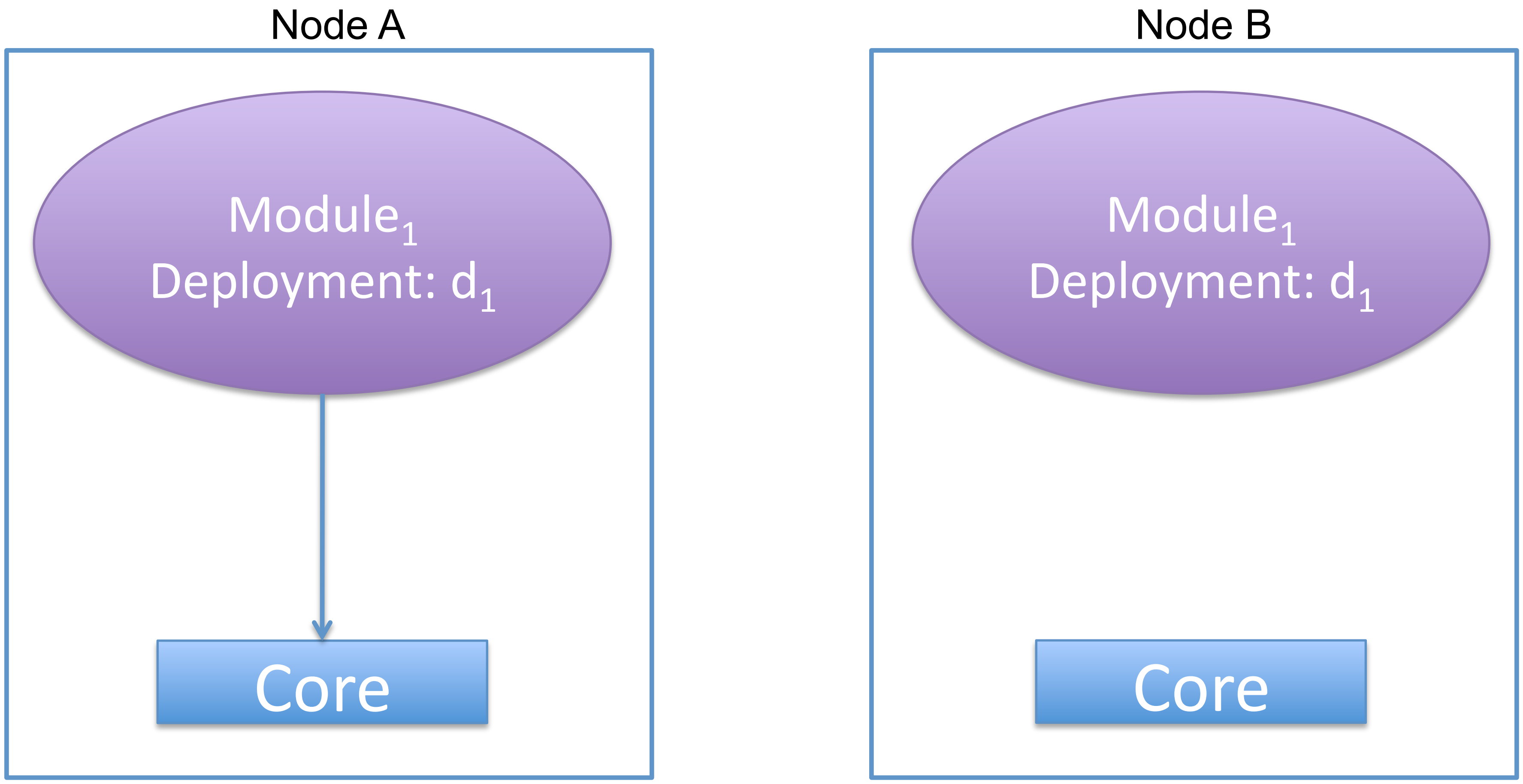
## Examples of gossip interactions

- Normal case: core piggybacks deployment numbers and checks for matched modules
- Mismatched deployment numbers: core initiates code update
- Modules fail to gossip usefully: core gossips configuration information

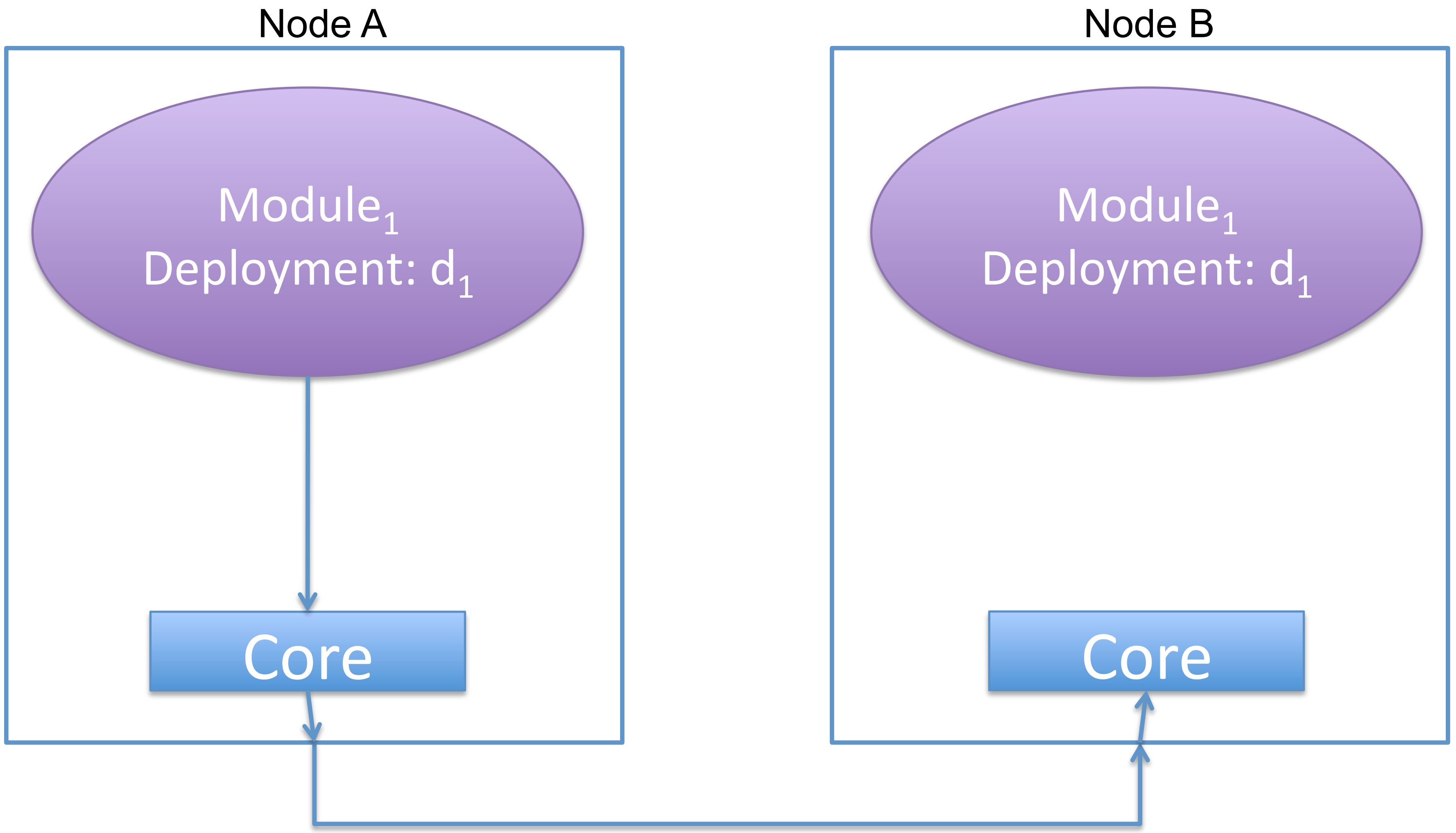
# Normal Case


Node A
Module$_1$
Deployment: $d_1$
Core
Node B
Module$_1$
Deployment: $d_1$
Core

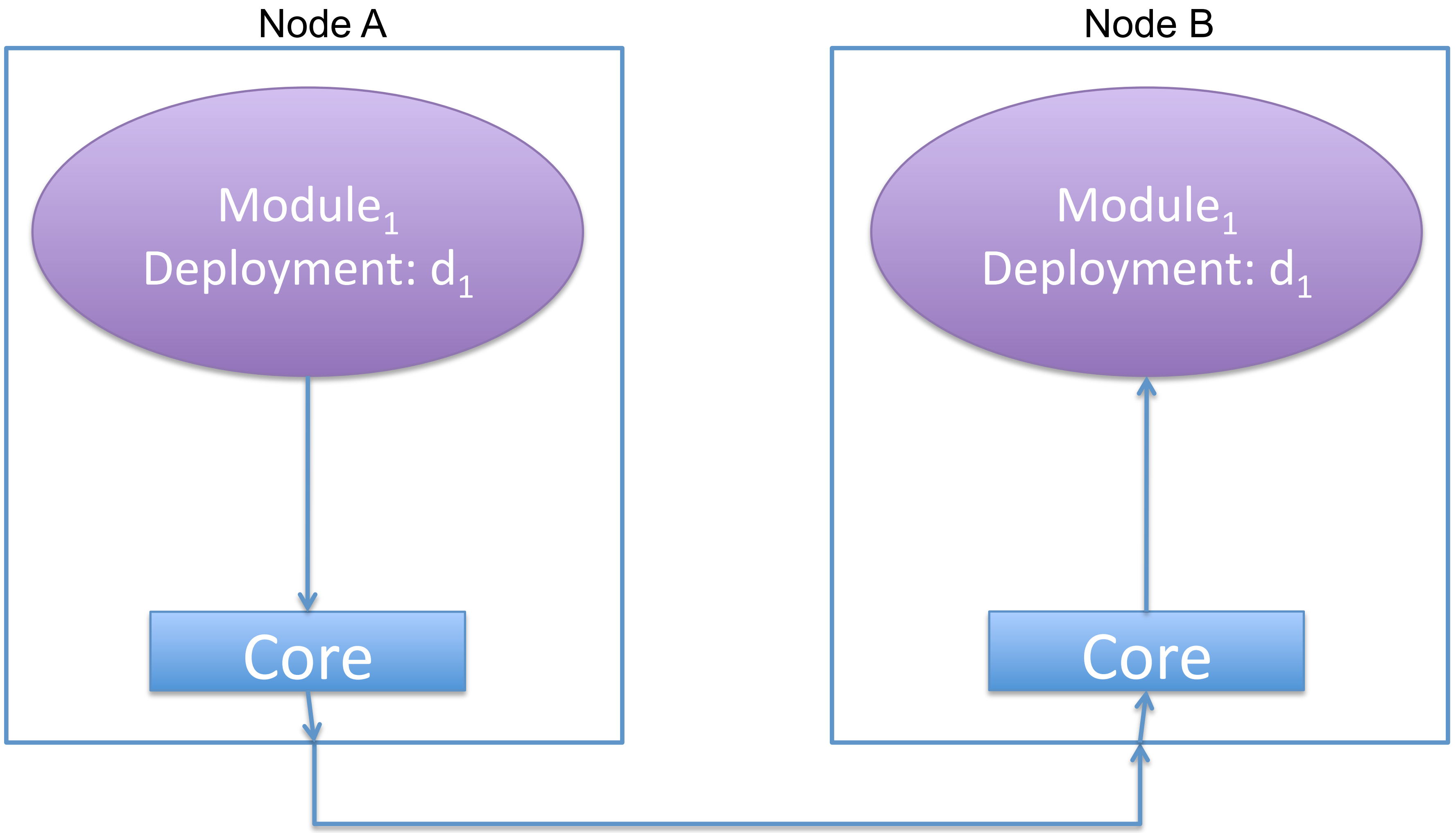
# Normal Case


Node A
Node B
$\text{Module}_1$
$\text{Deployment: } d_1$
$\text{Module}_1$
$\text{Deployment: } d_1$
Core
Core

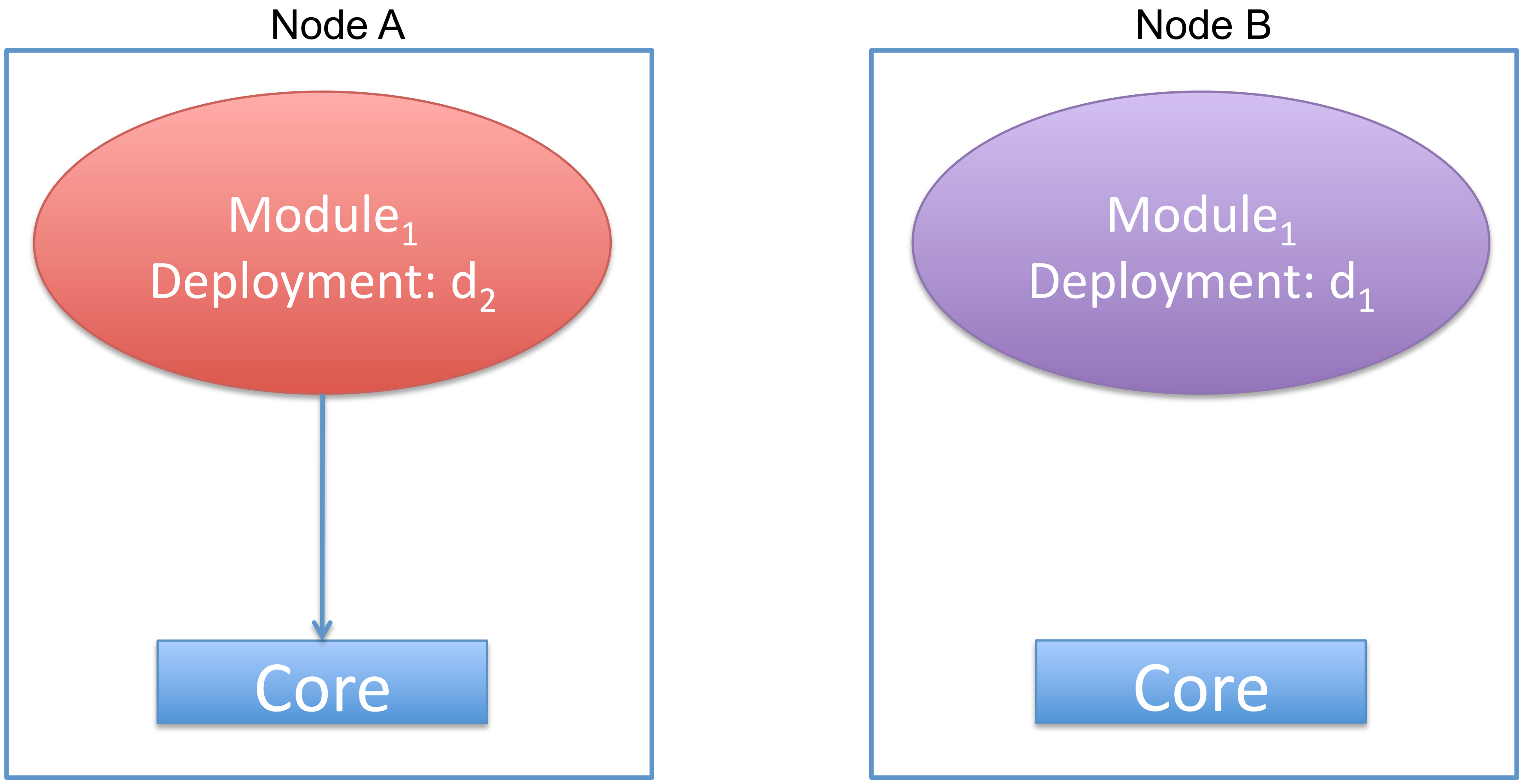
# Normal Case


Node A
Node B
Module$_1$
Deployment: $d_1$
Core
Module$_1$
Deployment: $d_1$
Core

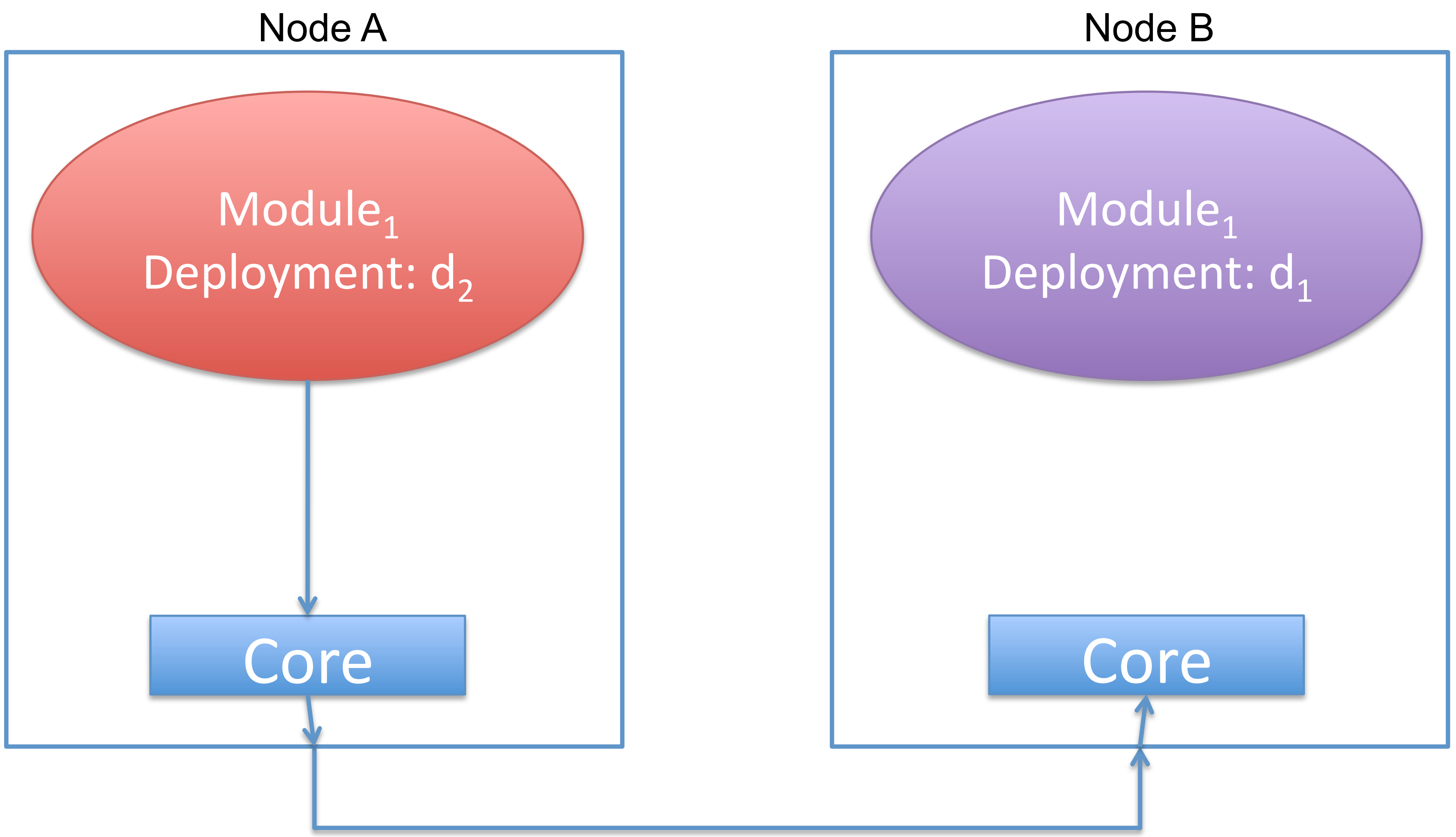
# Normal Case


Node A
Node B
$\text{Module}_1$
$\text{Deployment: } d_1$
$\text{Module}_1$
$\text{Deployment: } d_1$
Core
Core

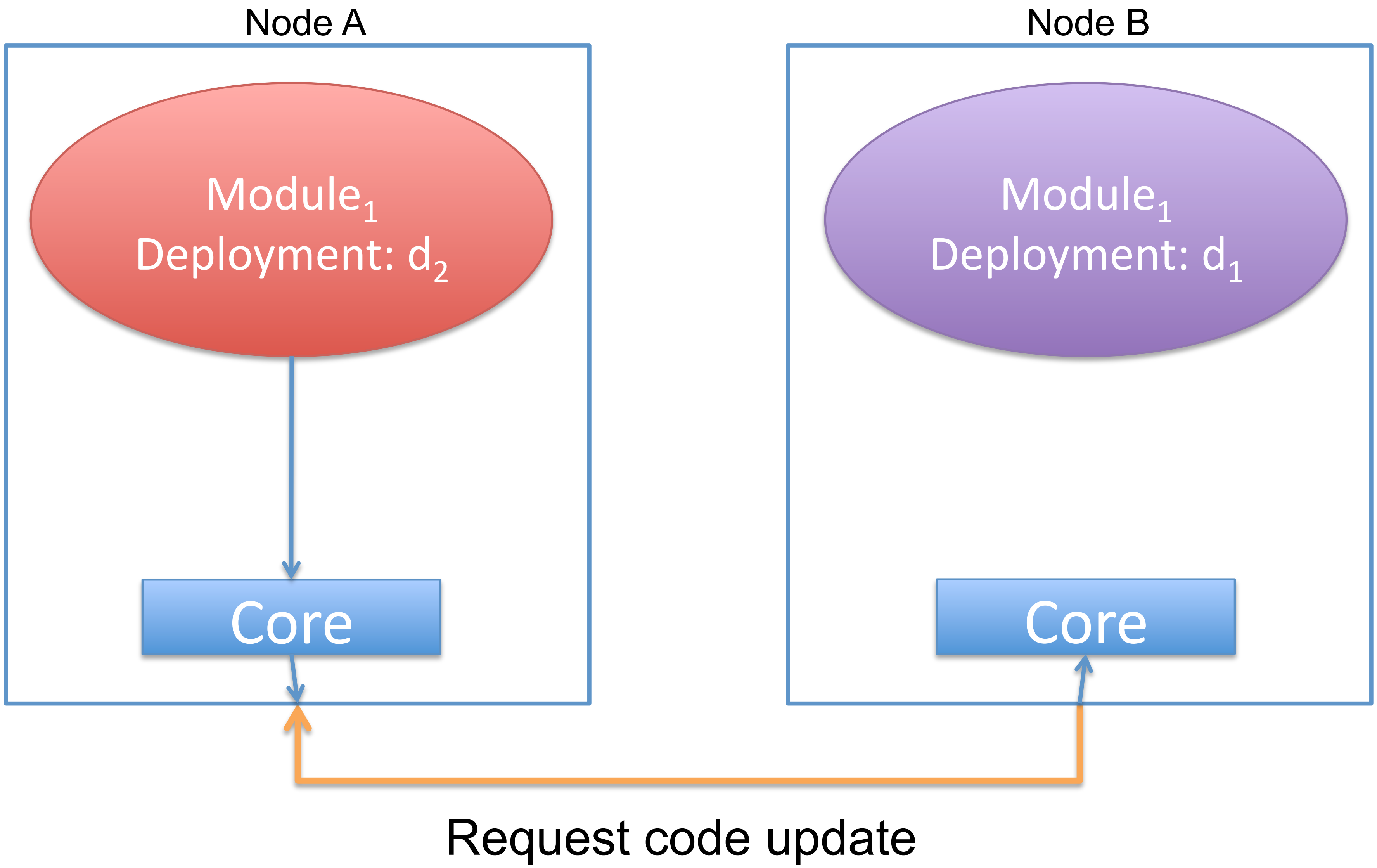
# Mismatched Deployment Numbers



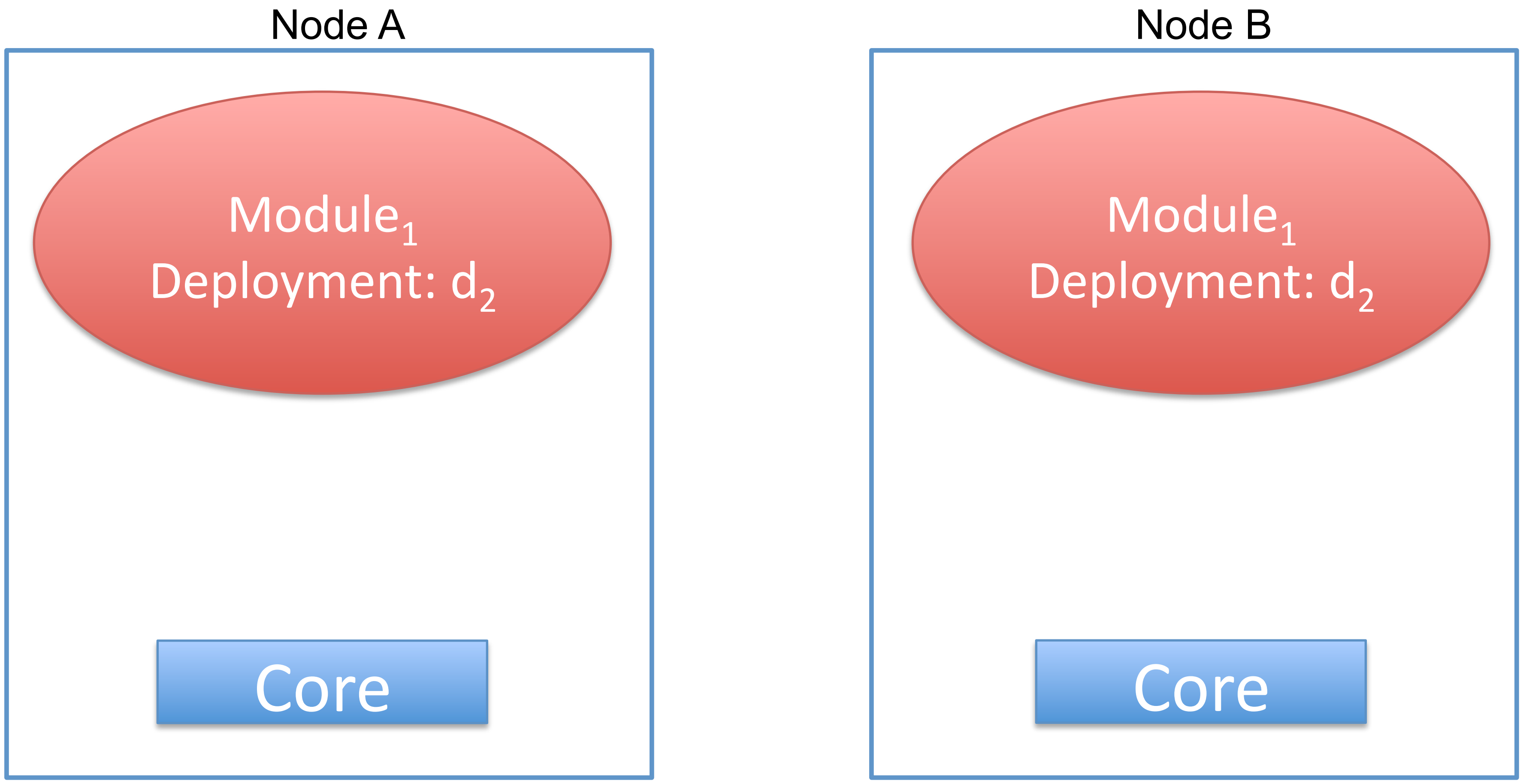
# Mismatched Deployment Numbers



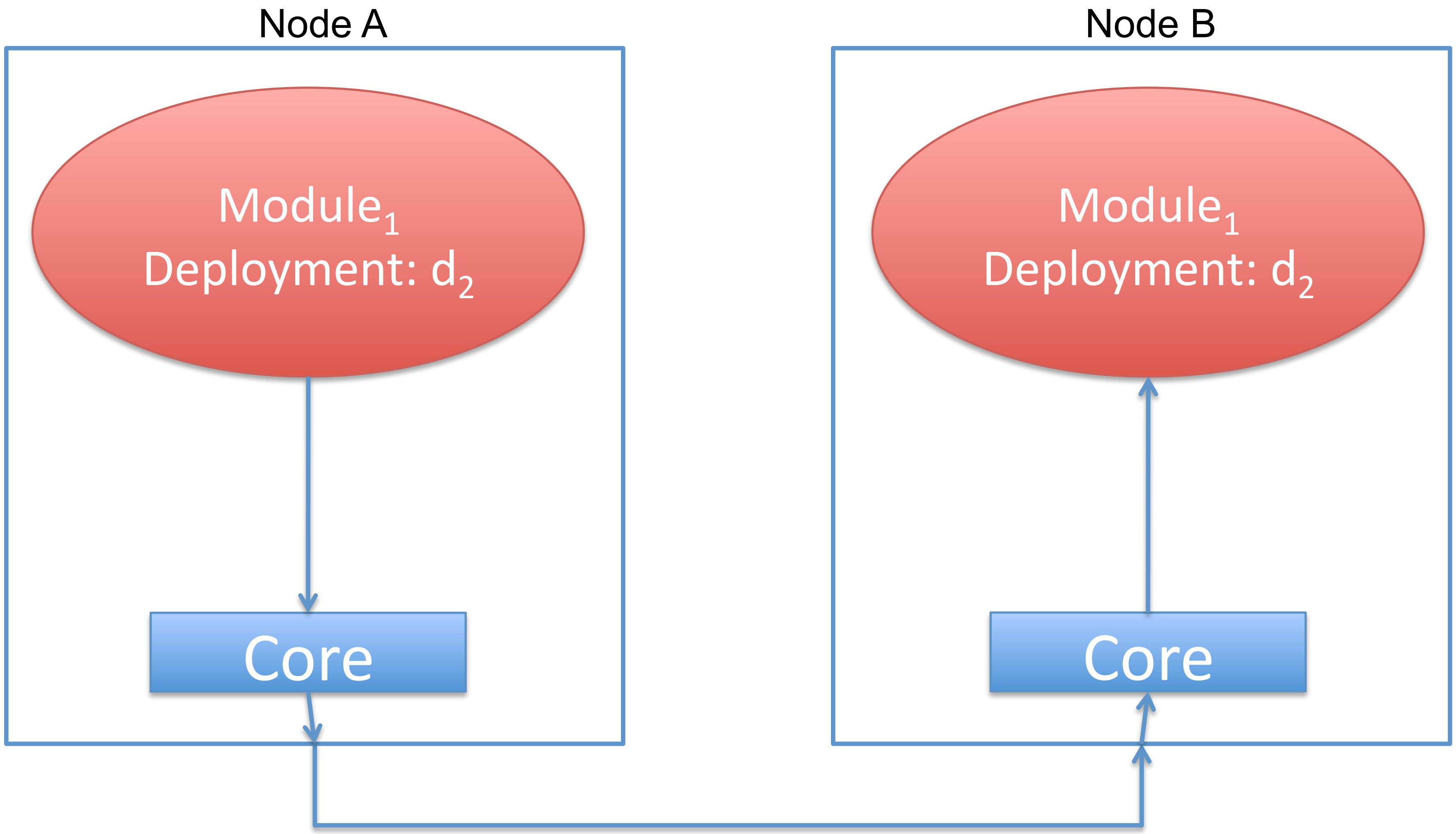
# Mismatched Deployment Numbers



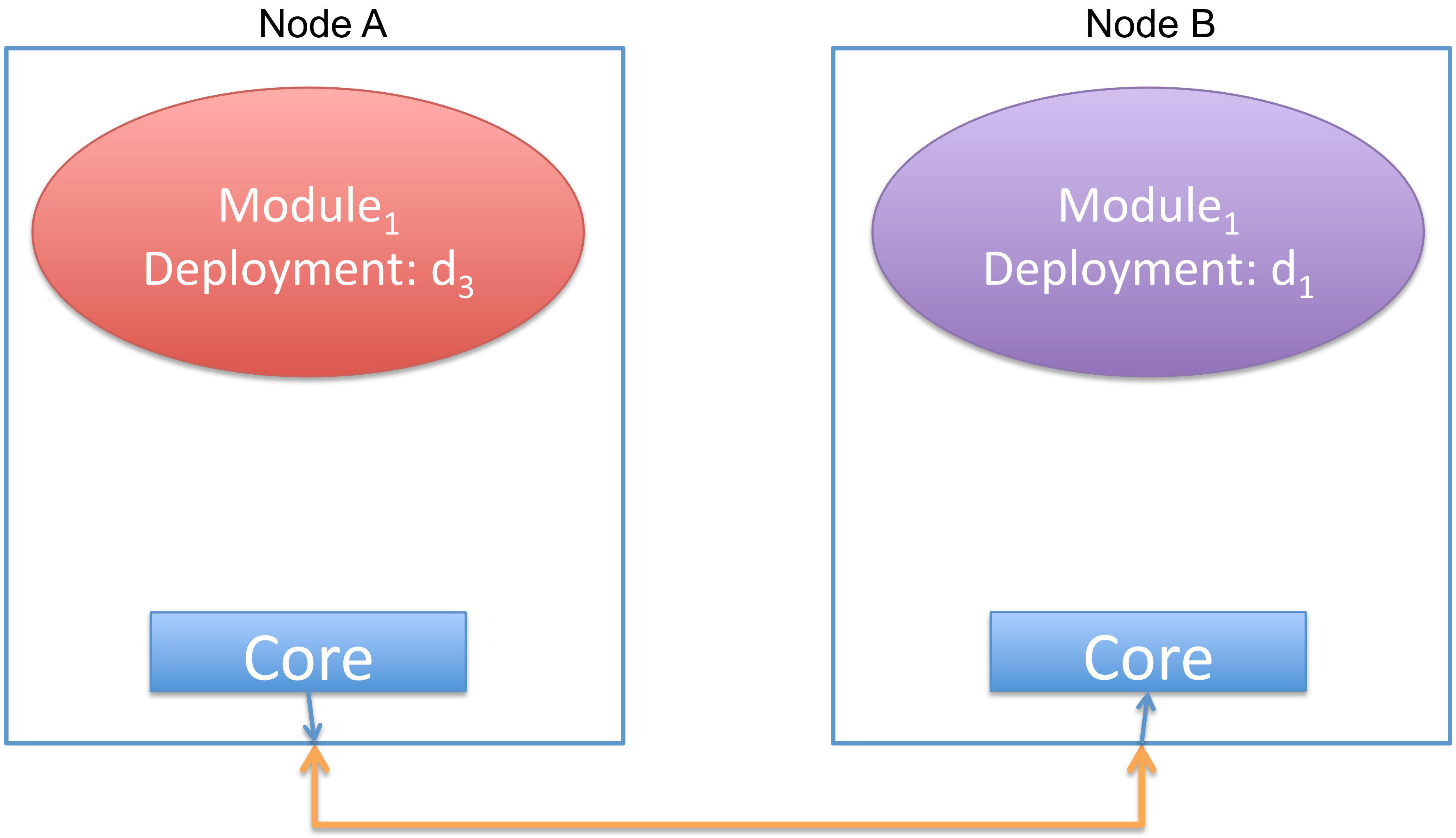
# Mismatched Deployment Numbers



# Mismatched Deployment Numbers

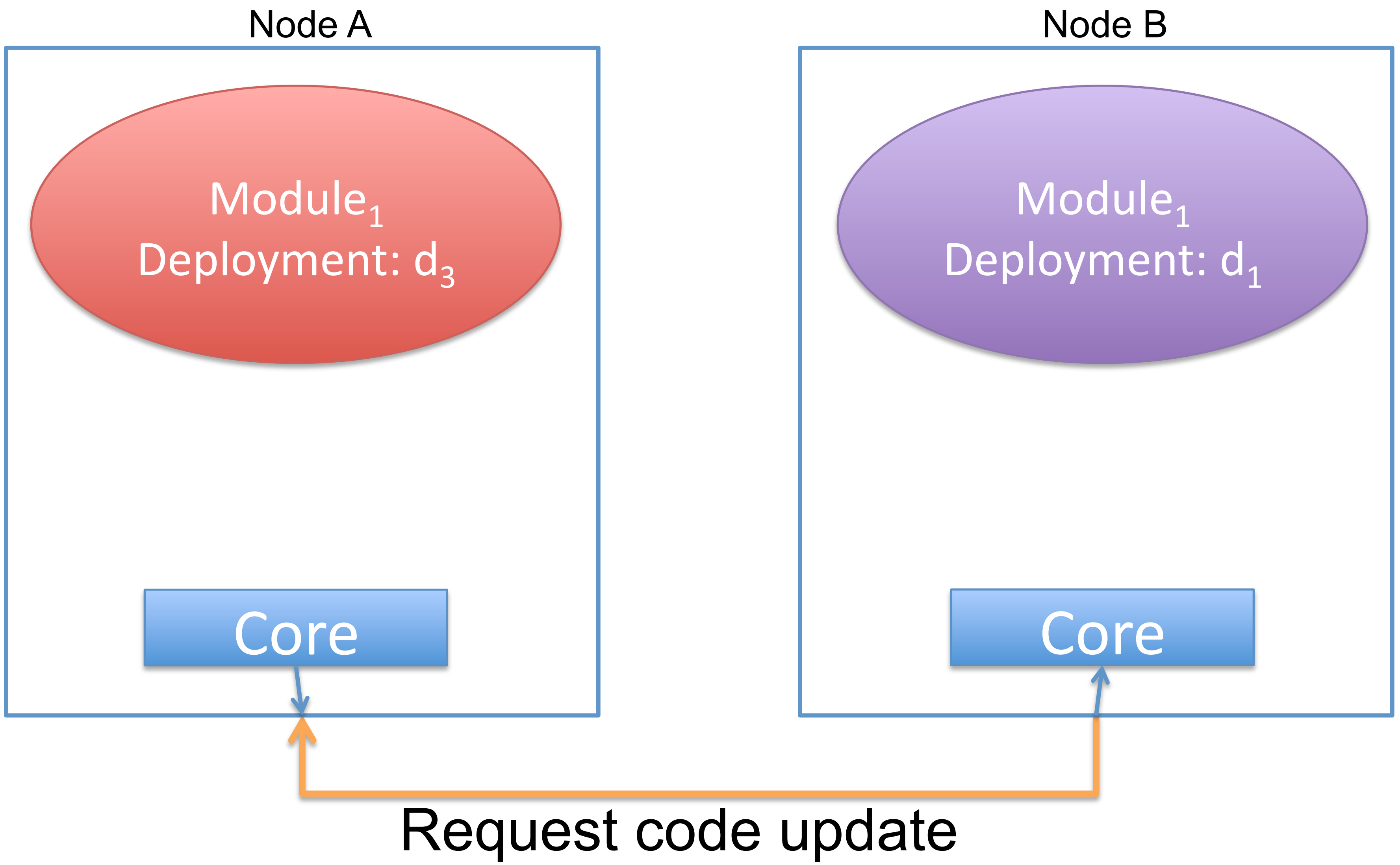


# Failure to Gossip usefully

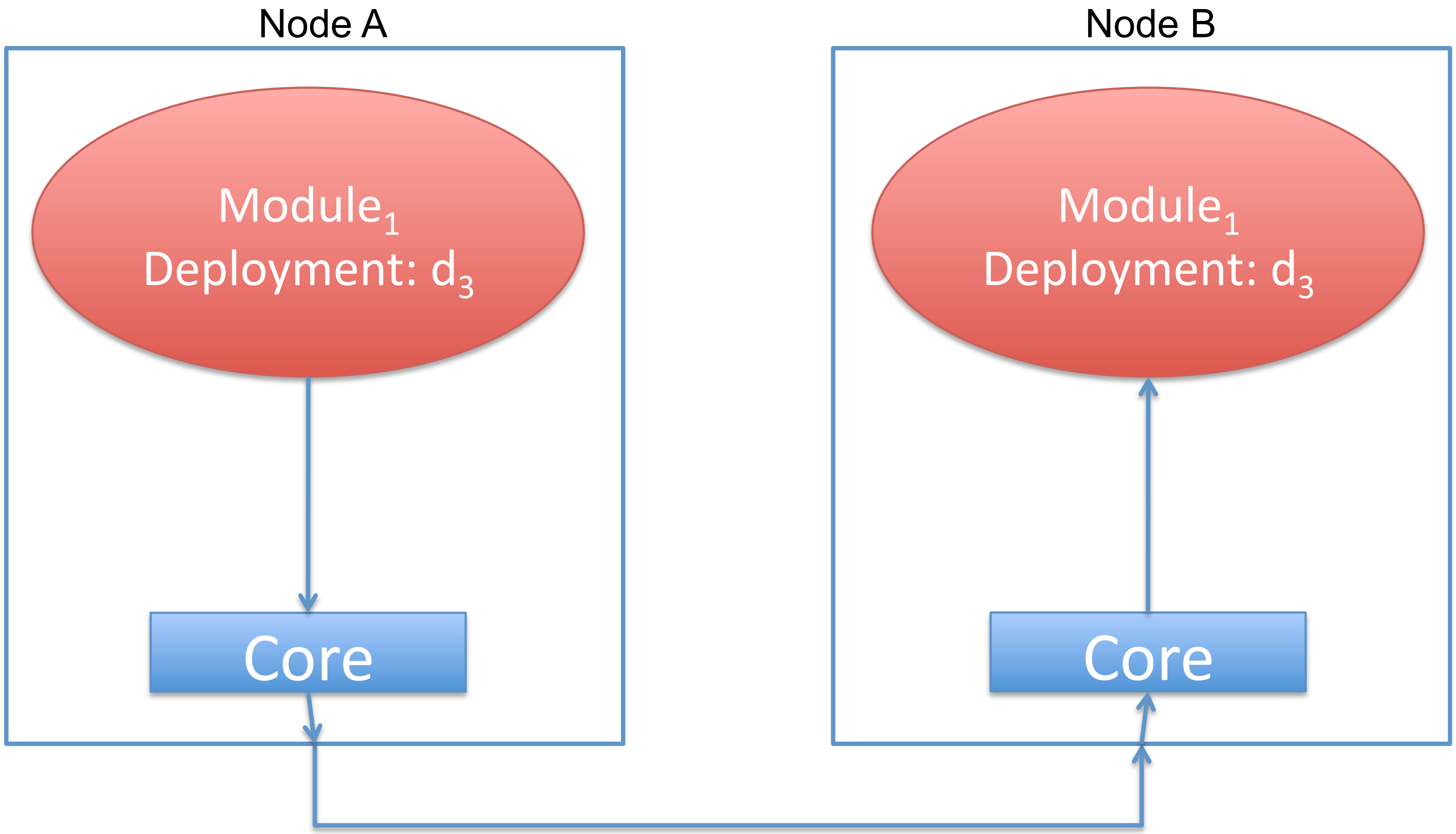


Exchange configuration deployment number

# Failure to Gossip usefully



# Failure to Gossip usefully



## Talk Outline

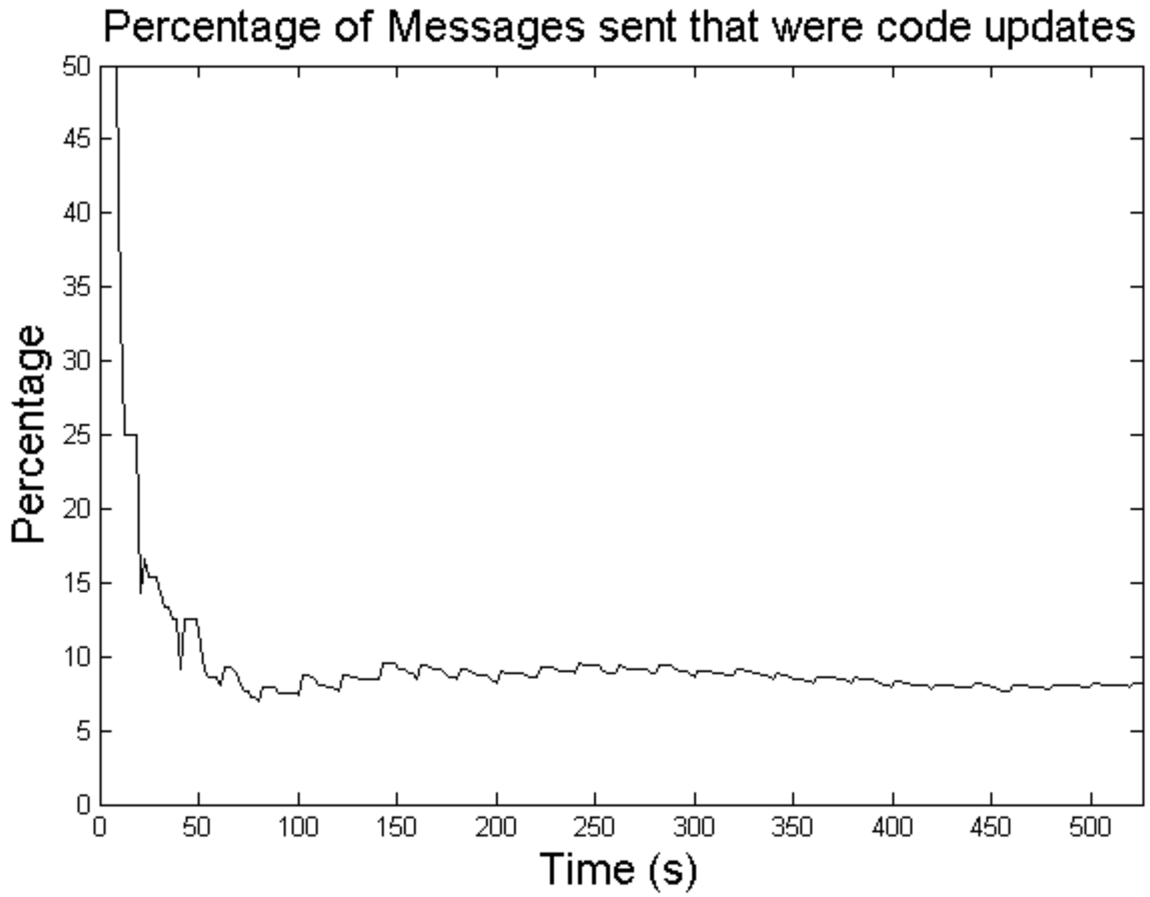
- Code Updating
- Layered Architecture
- Evaluation
- Conclusion and Future Work

## Evaluation

- Tested on 100 local instances with 10 serving as rendezvous servers
- Application: A Simple Membership Protocol

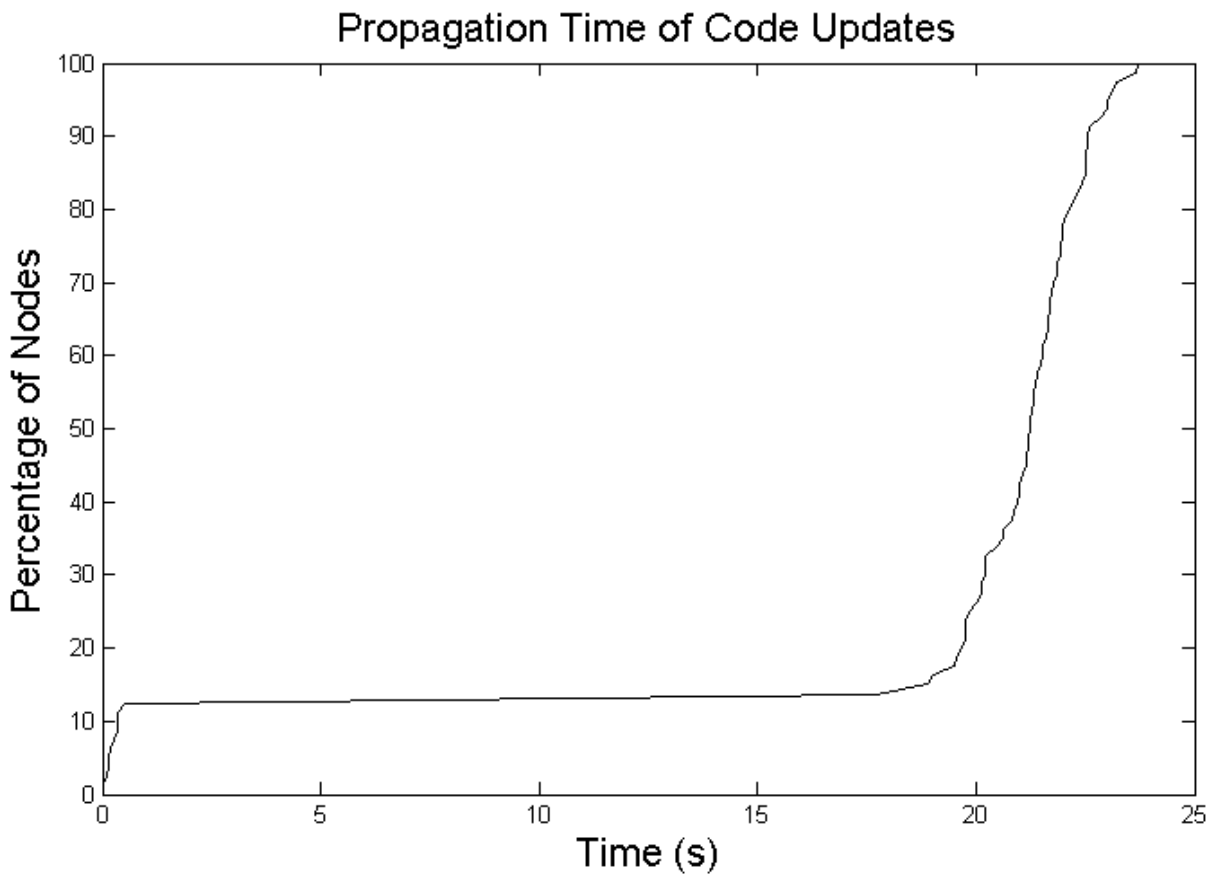
# Evaluation

- How much overhead does the core add?



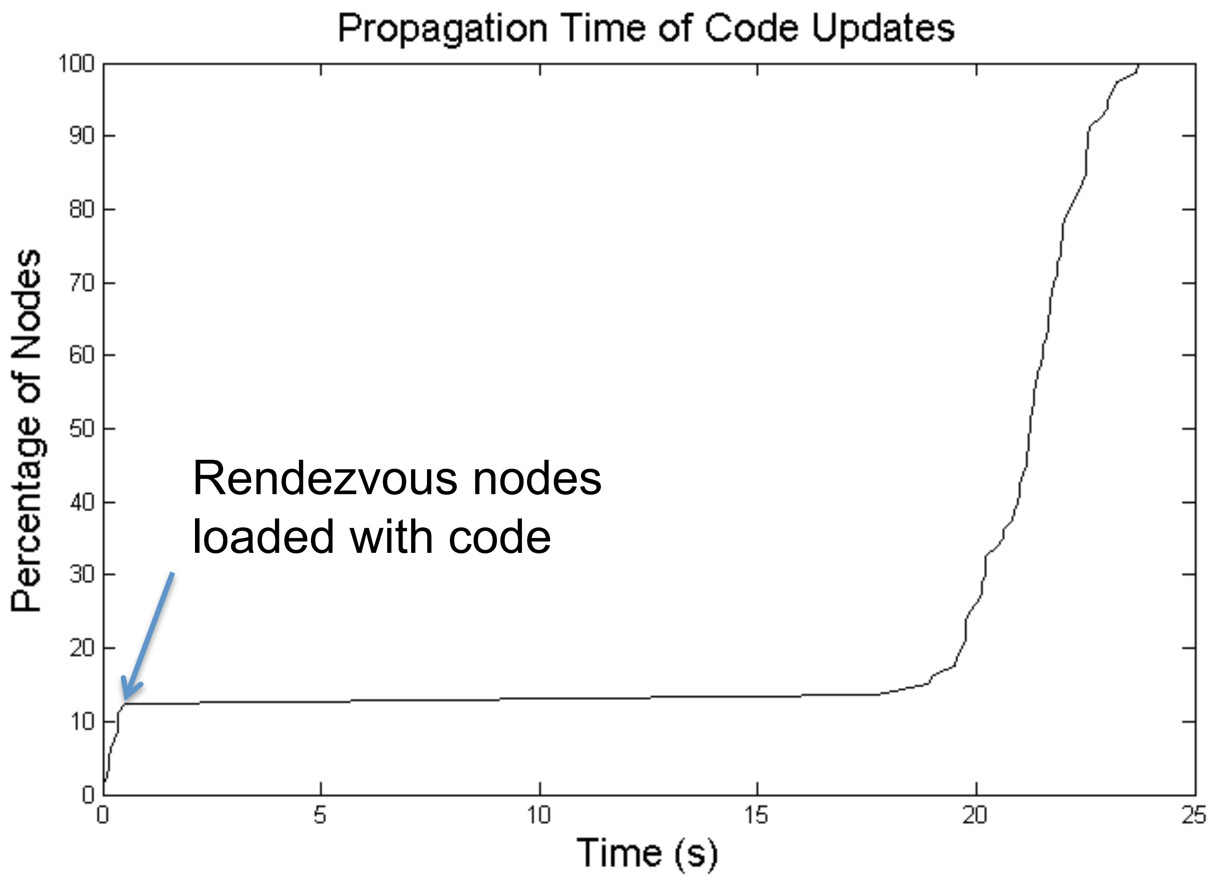
## Evaluation

- How long does it take to propagate code?



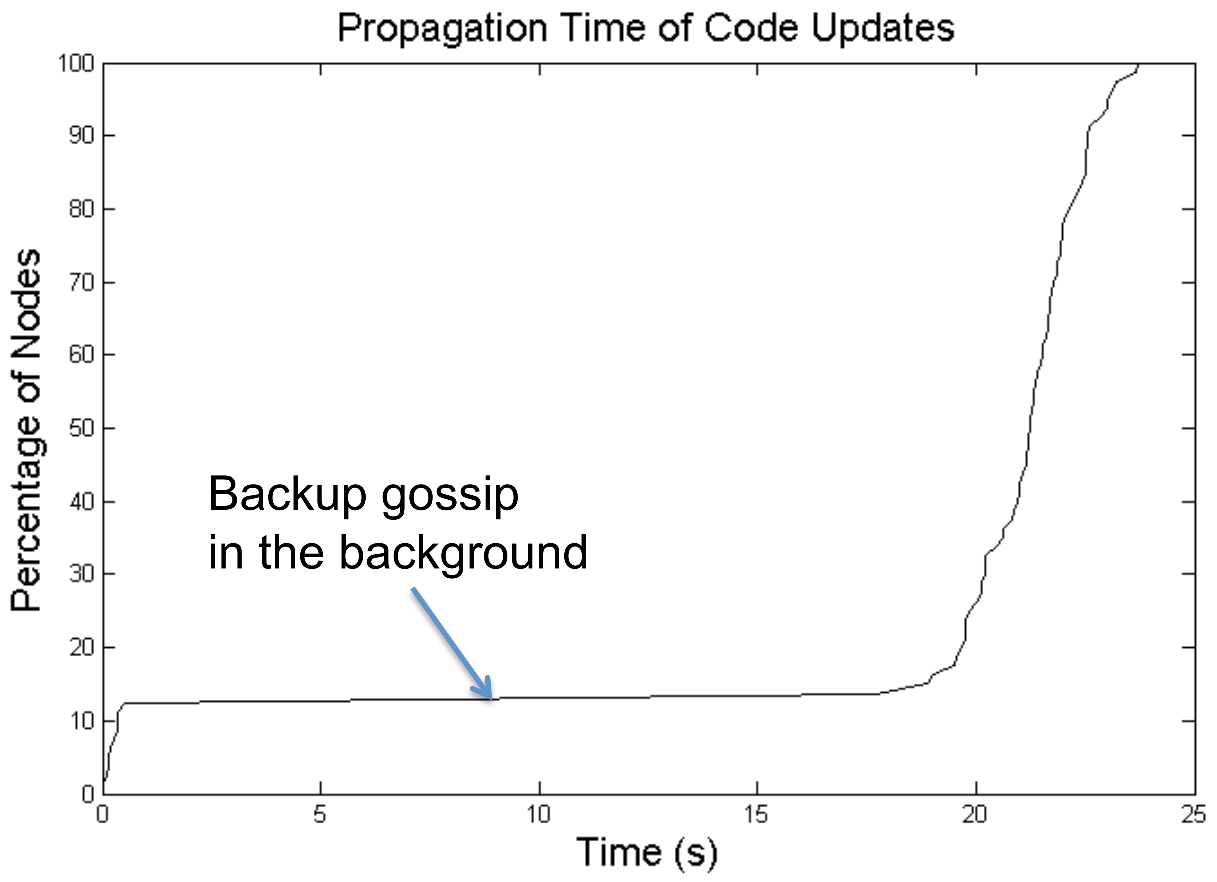
# Evaluation

- How long does it take to propagate code?



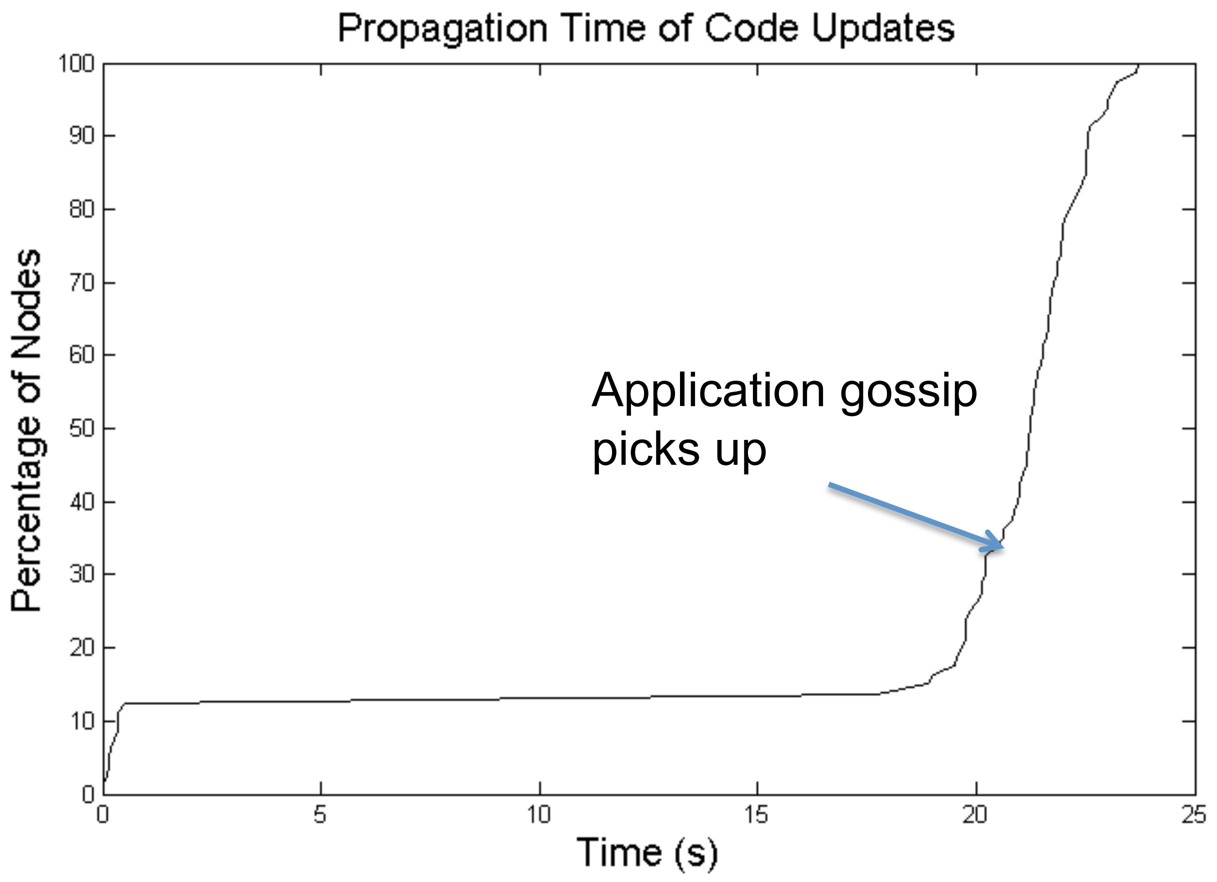
# Evaluation

- How long does it take to propagate code?



# Evaluation

- How long does it take to propagate code?



# Conclusion and Future Work

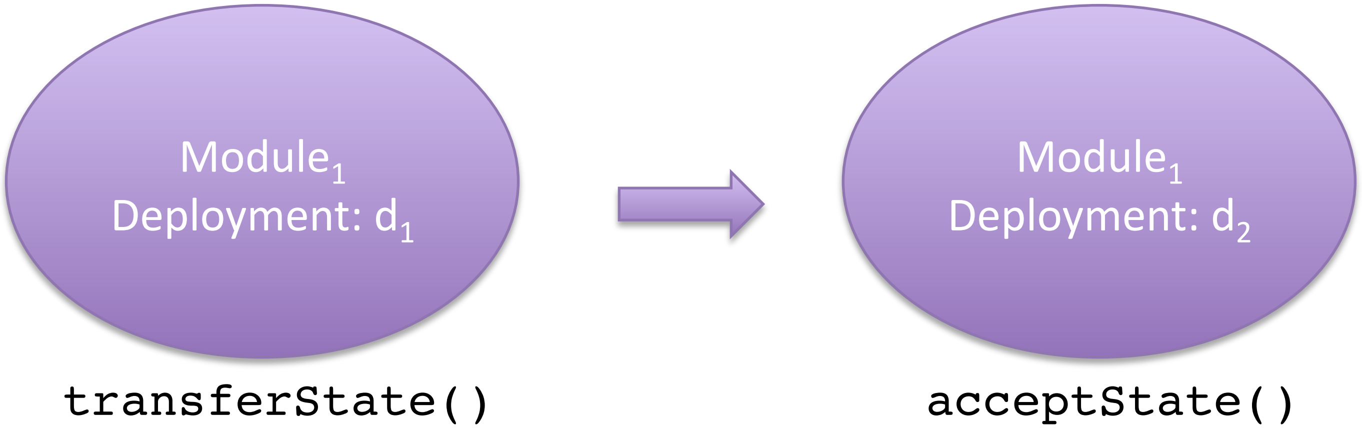
- Can we make the core smaller?
- Can the core be updated?
- Security
- NAT Traversal as a layered service

Questions?

# Module Management

- Core provides the following public methods for module updating:

```
public String transferState()
public void acceptState()
```


Module$_1$
Deployment: d$_1$
Module$_1$
Deployment: d$_2$
transferState()
acceptState()