Self-Management for Large Scale Distributed Systems

Ahmad Al-Shishtawy (ahmadas@kth.se)

Advisors:
Vladimir Vlassov (vladv@kth.se)

Seif Haridi (seif@sics.se)

KTH Royal Institute of Technology
Stockholm, Sweden

CONTRAIL Summer school 2011

June 28, 2011

Swedish Institute of Computer Science

Outline

Outline

Dealing with Complexity

Problem

All computing systems need to be managed

Dealing with Complexity

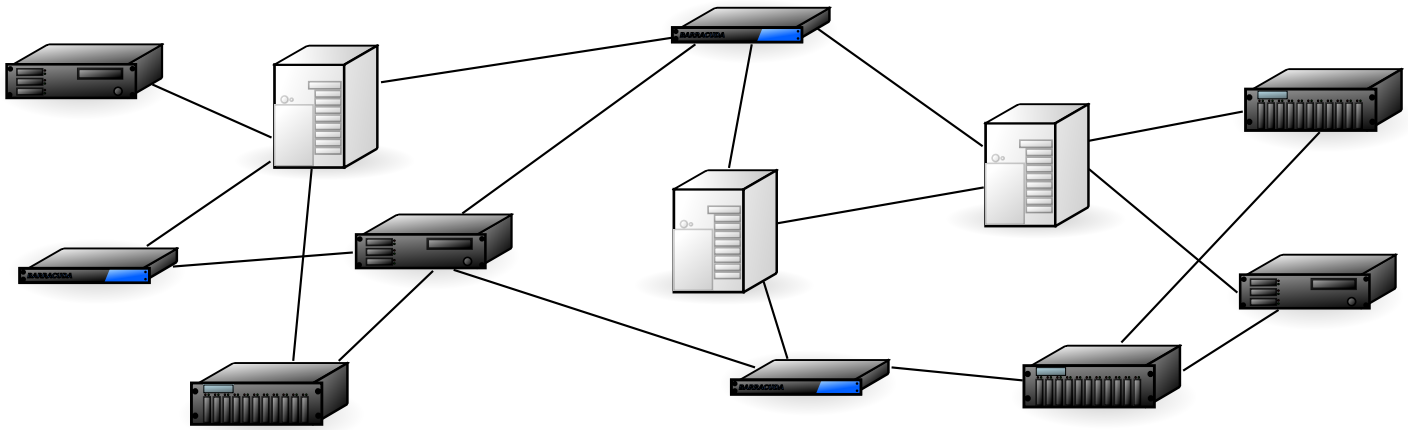
Problem

All computing systems need to be managed

Dealing with Complexity

Problem

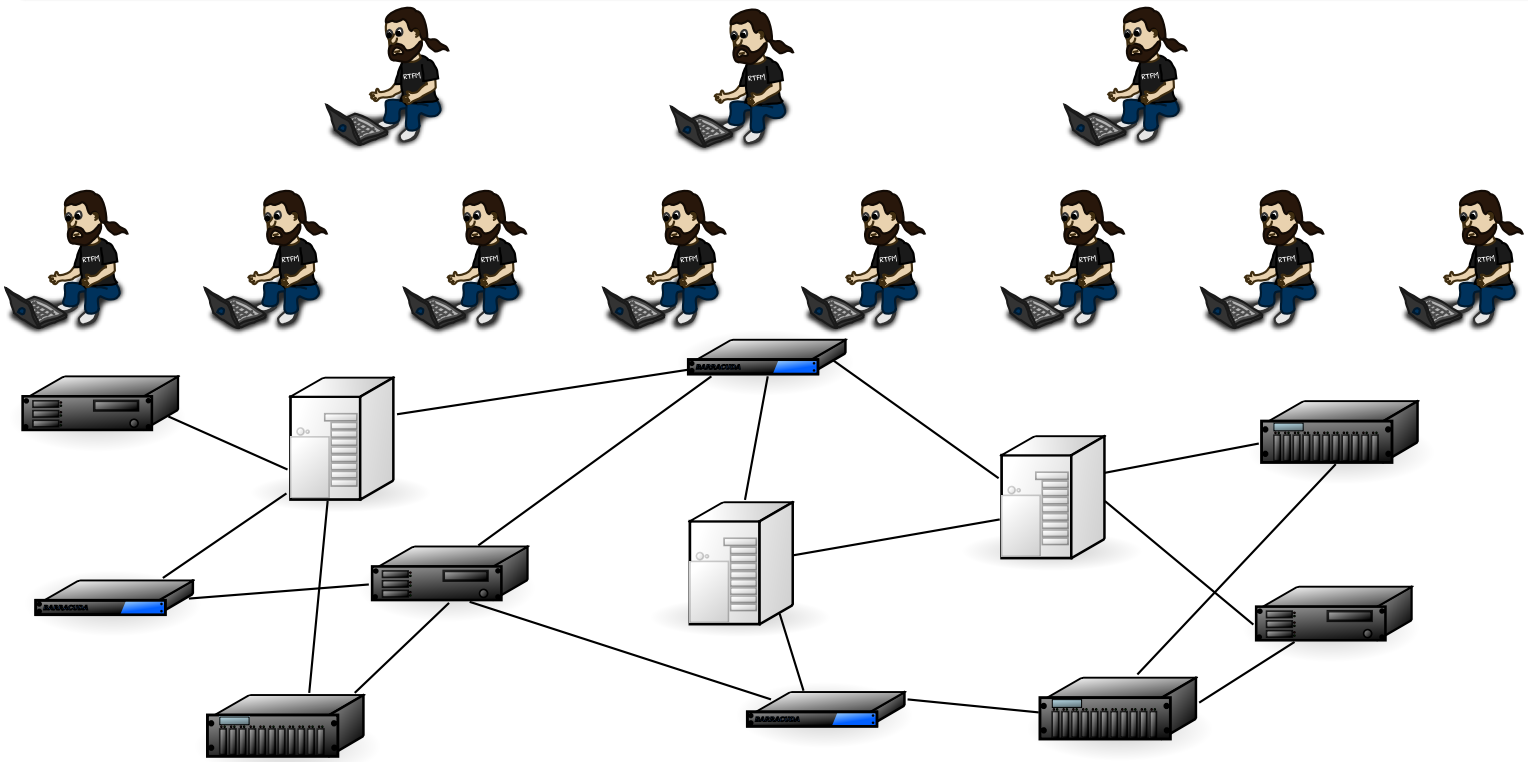
Computing systems are getting more and more complex



Dealing with Complexity

Problem

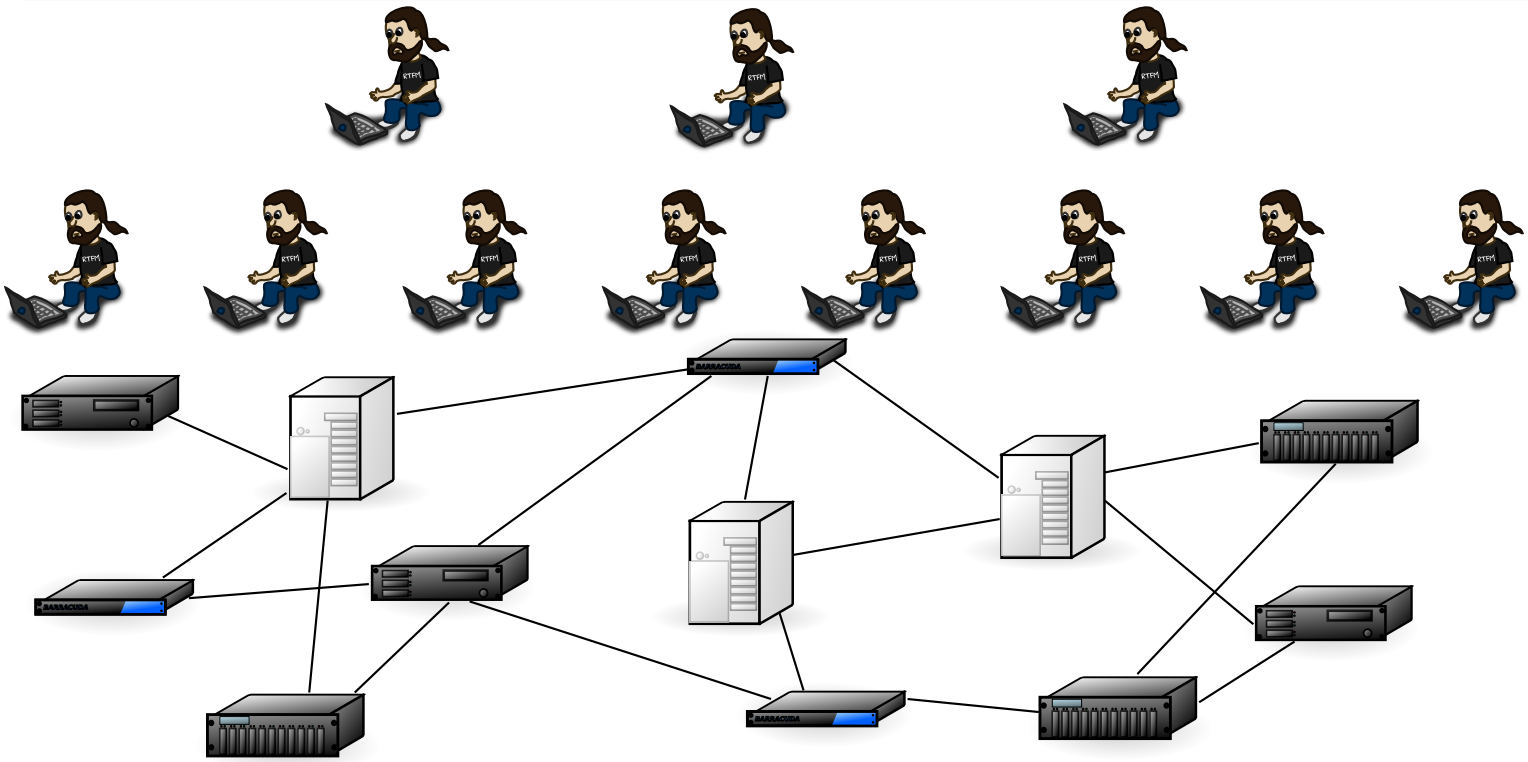
Complexity means higher administration overheads



Dealing with Complexity

Problem

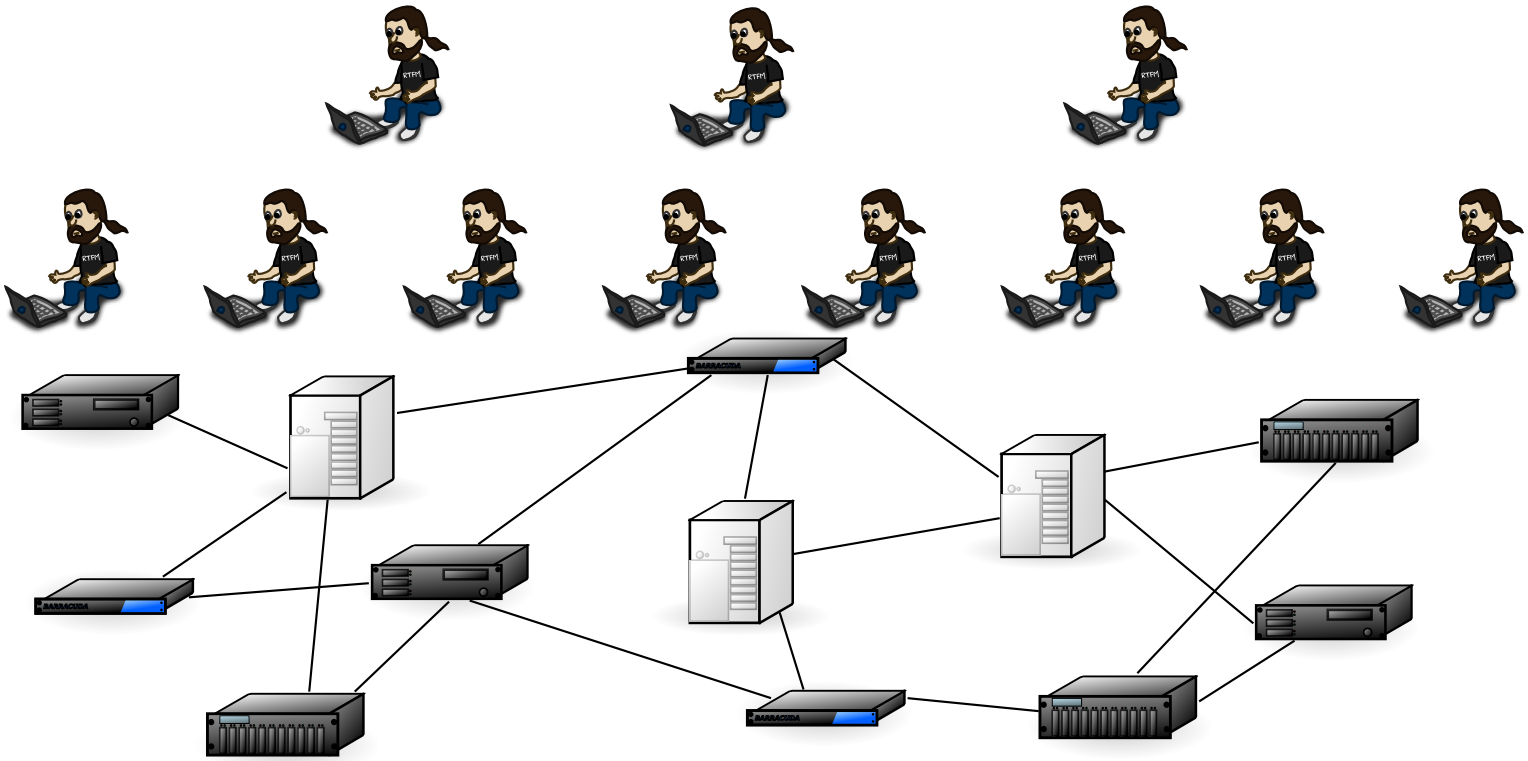
Complexity poses a barrier on further development



Dealing with Complexity

Solution

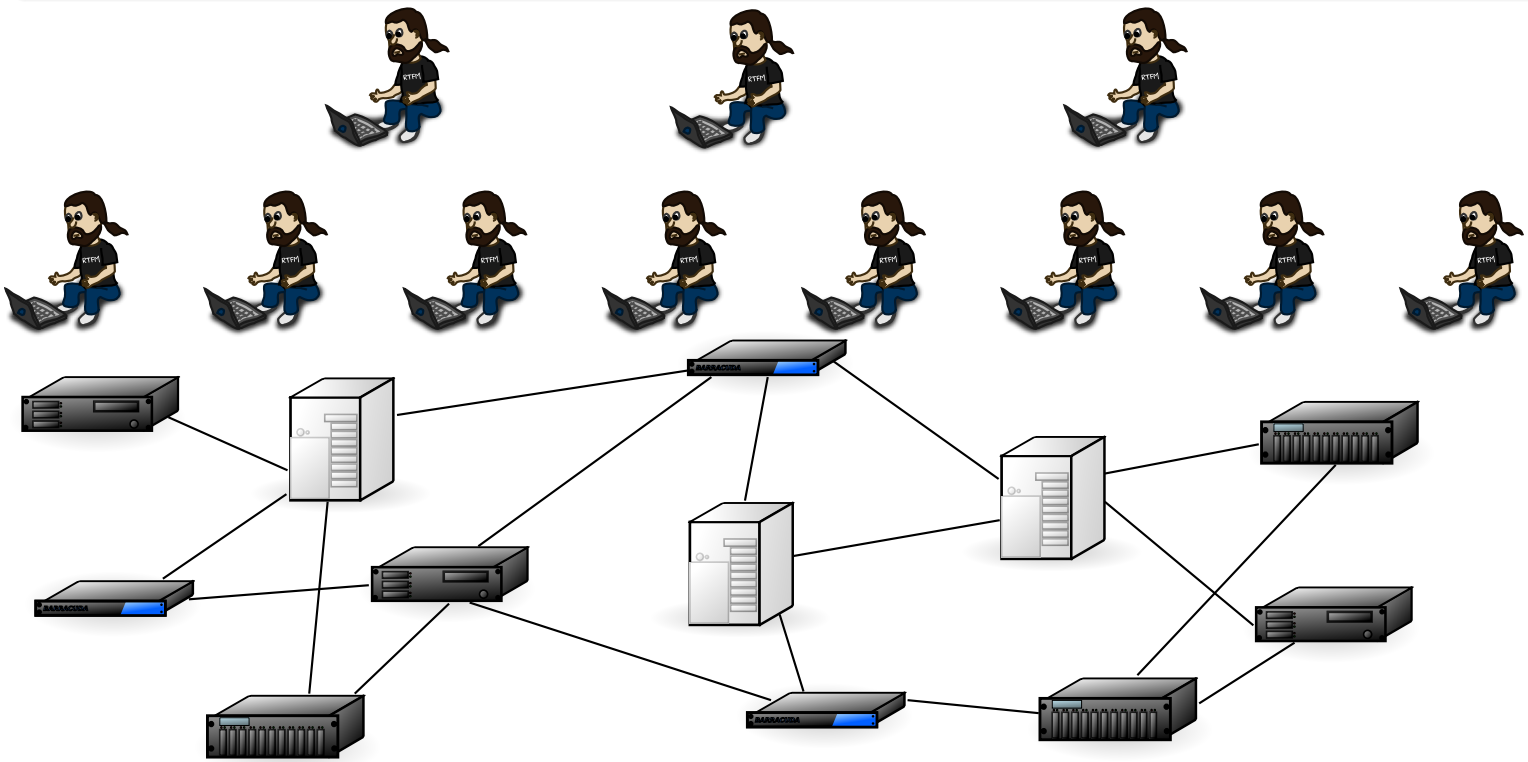
The Autonomic Computing initiative by IBM



Dealing with Complexity

Solution

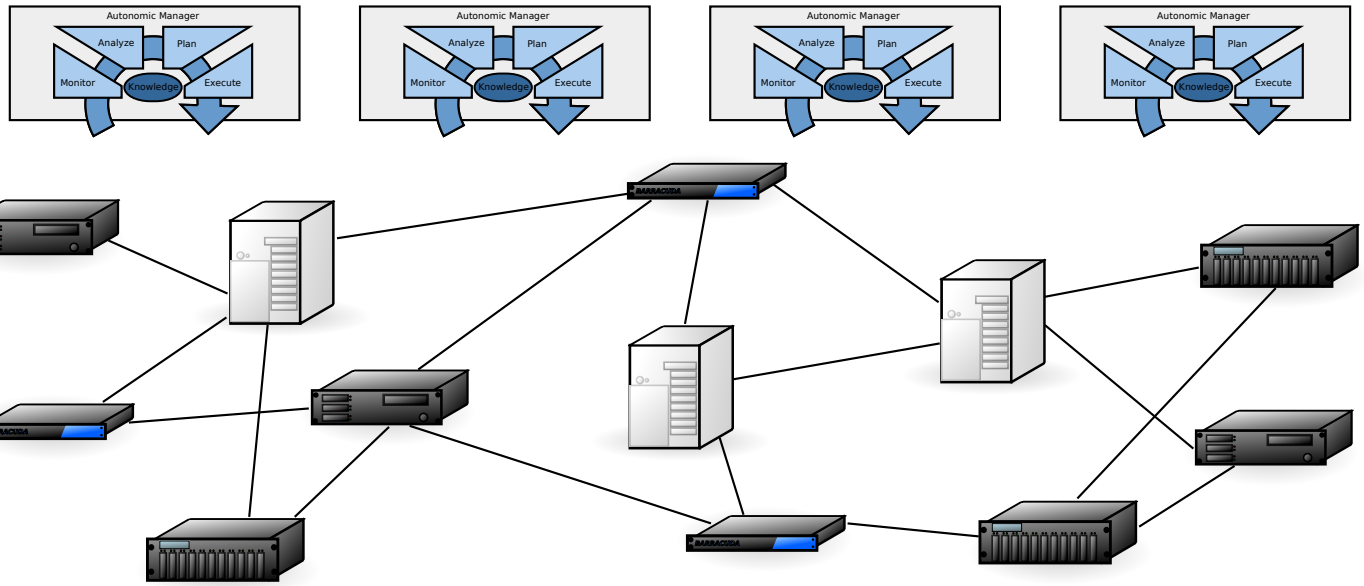
Self-Management: Systems capable of managing themselves



Dealing with Complexity

Solution

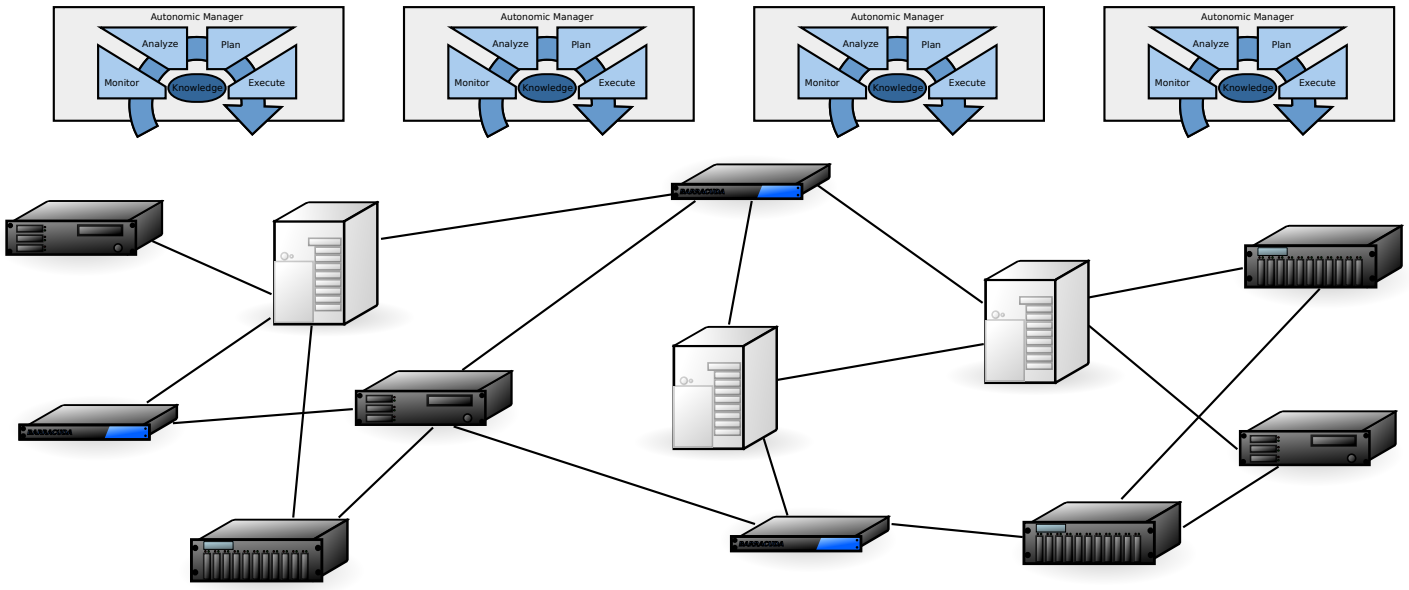
Use Autonomic Managers



Dealing with Complexity

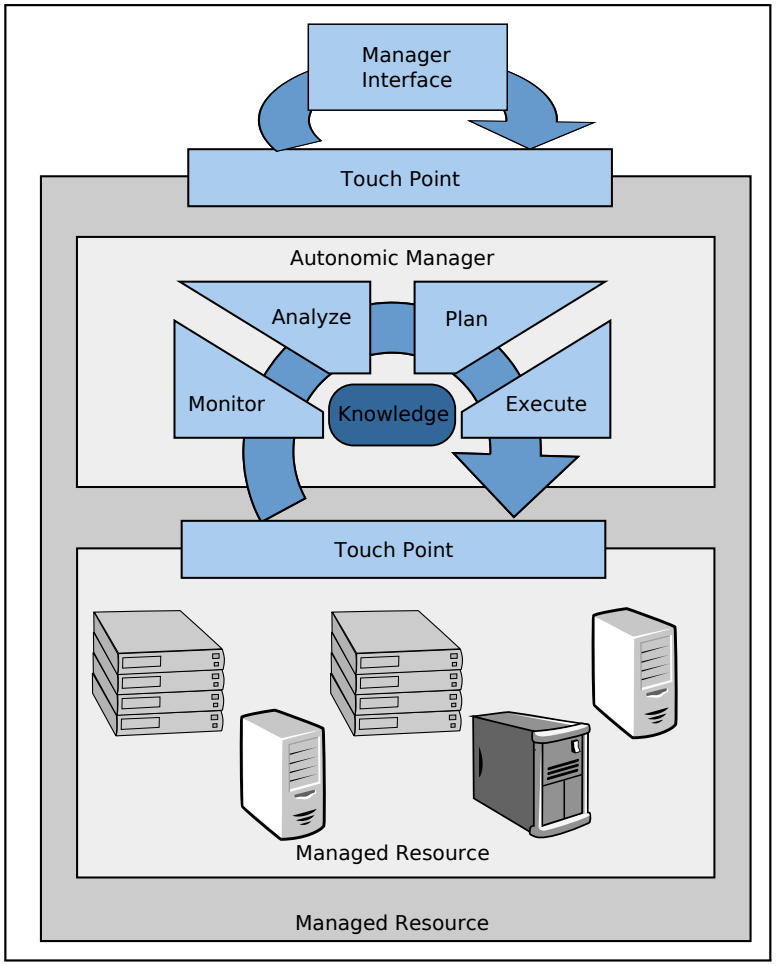
Open Question

How to achieve Self-Management?



The Autonomic Computing Architecture

- Managed Resource
- Touchpoint (Sensors & Actuators)
- Autonomic Manager
  - Monitor
  - Analyze
  - Plan
  - Execute
- Knowledge Source
- Communication
- Manager Interface



The Goal

Large-scale distributed systems

- Complex and require self-management
- May run on unreliable resources
- Major sources of complexity:
  - Scale (resources, events, users, . . .)
  - Dynamism (resource churn, load changes, . . .)

Goal

- A platform (concepts, abstractions, algorithms. . .) that facilitates development of self-managing applications in large-scale and/or dynamic distributed environment.
- A methodology that help us to achieve self-management.

The Goal

Large-scale distributed systems

- Complex and require self-management
- May run on unreliable resources
- Major sources of complexity:
  - Scale (resources, events, users, . . .)
  - Dynamism (resource churn, load changes, . . .)

Goal

- A platform (concepts, abstractions, algorithms. . .) that facilitates development of self-managing applications in large-scale and/or dynamic distributed environment.
- A methodology that help us to achieve self-management.

Research Plan

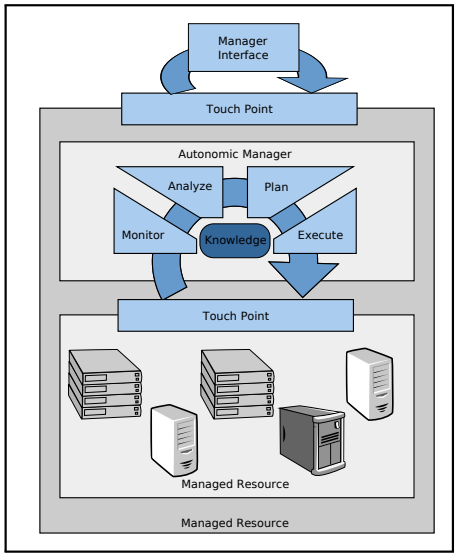
Self-Management in large-scale distributed systems. Consists of four main parts:

- Part 1: Touchpoints and feedback loops in distributed systems
- Part 2: Robust Management
- Part 3: Improve management logic
- Part 4: Integrate previous parts in a self-managing system.

Outline

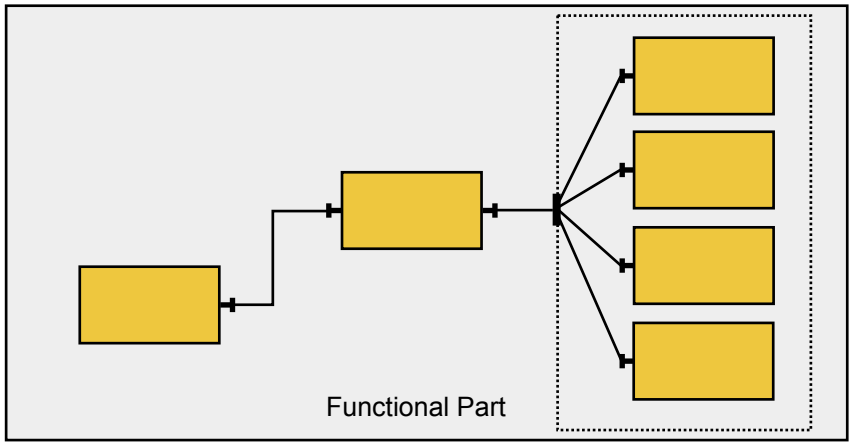
Niche

- Niche is a Distributed Component Management System
- Niche implements the Autonomic Computing Architecture for large-scale distributed environment
- Niche leverages Structured Overlay Networks for communication and for provisioning of basic services (DHT, Publish/Subscribe, Groups, etc.)



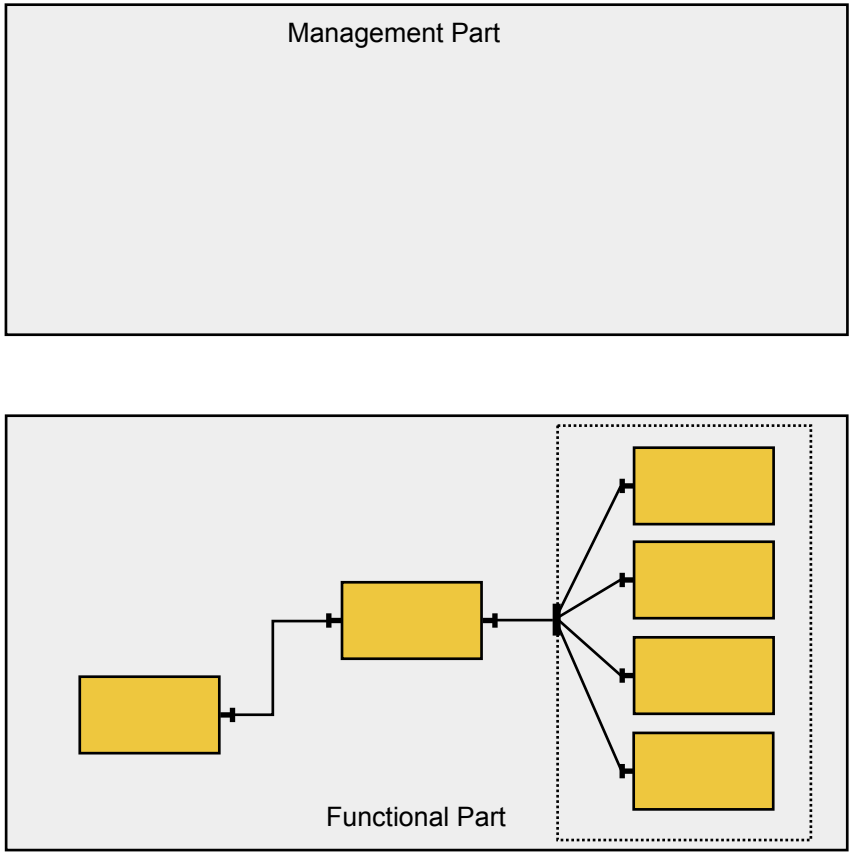
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API



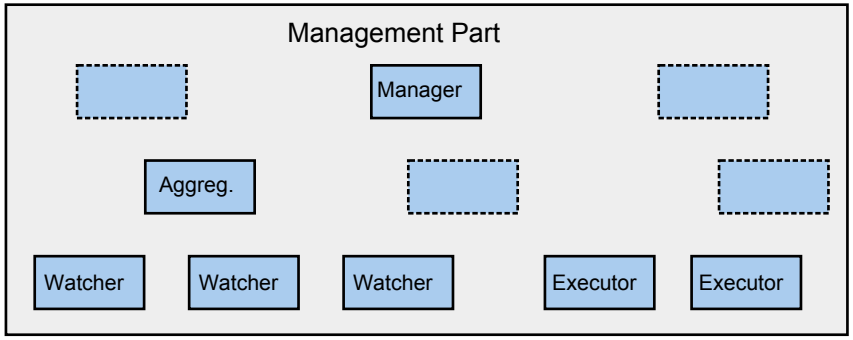
Management Part

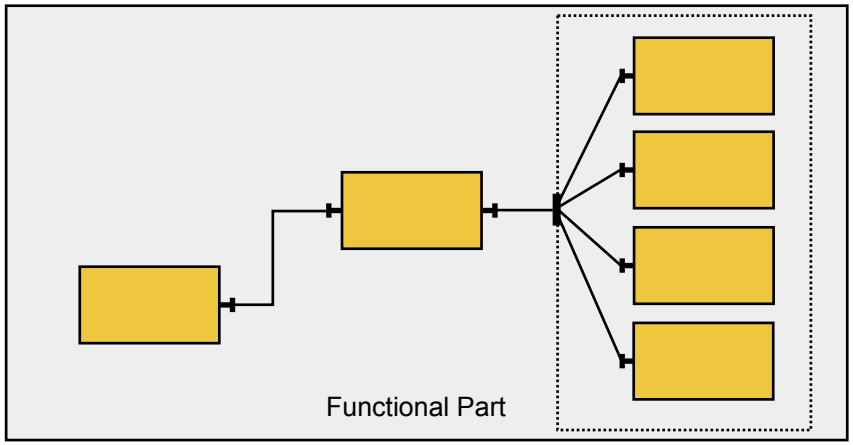
- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API



Management Part

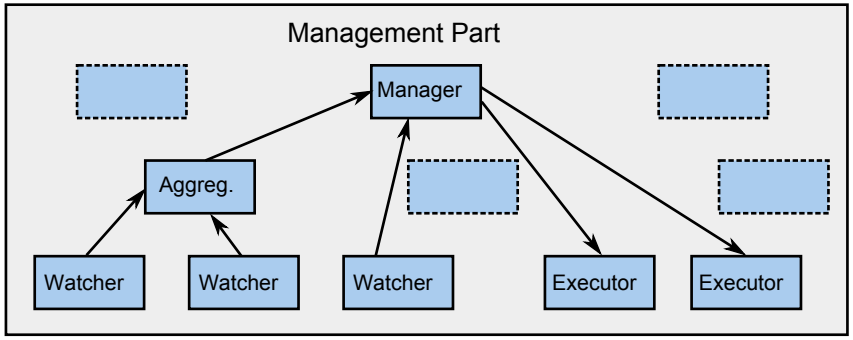
- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API

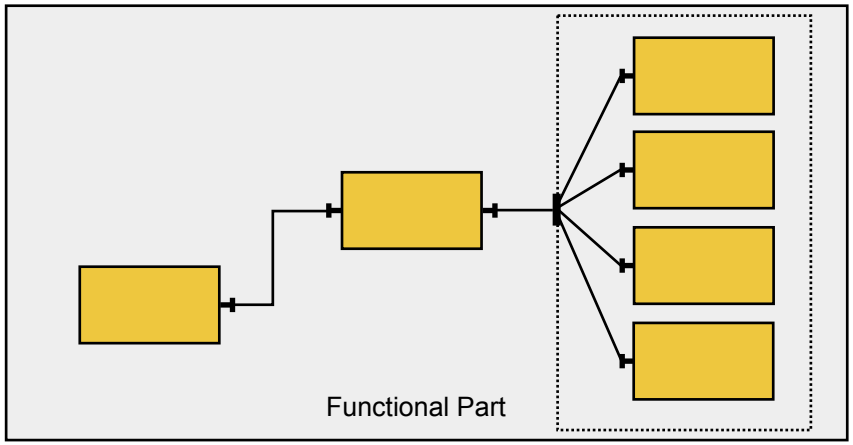




Management Part

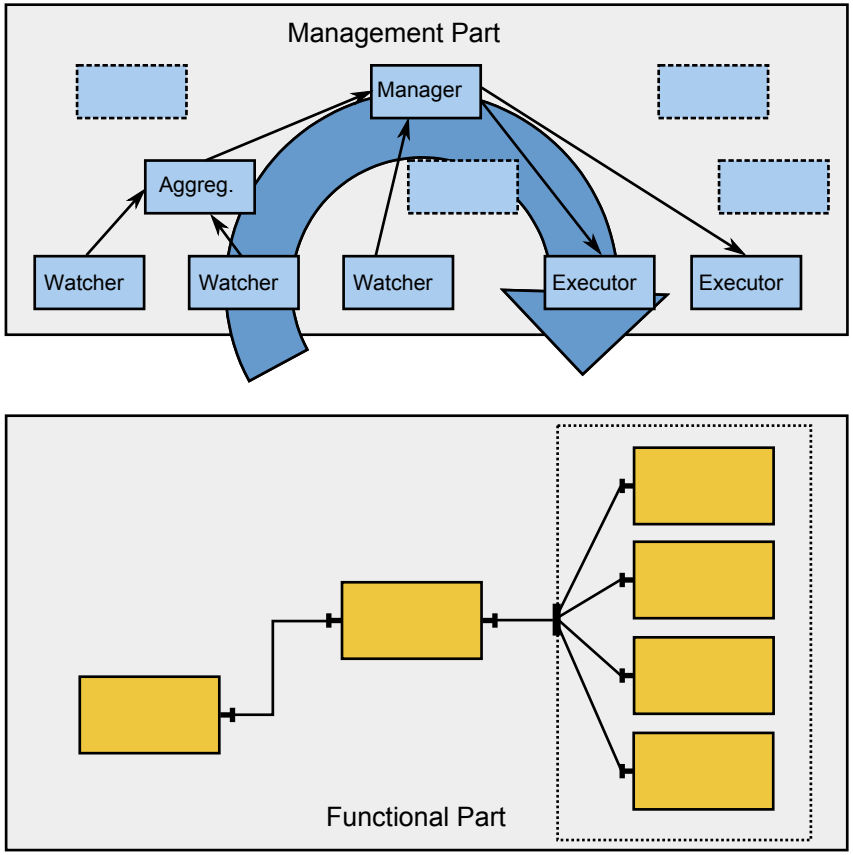
- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API





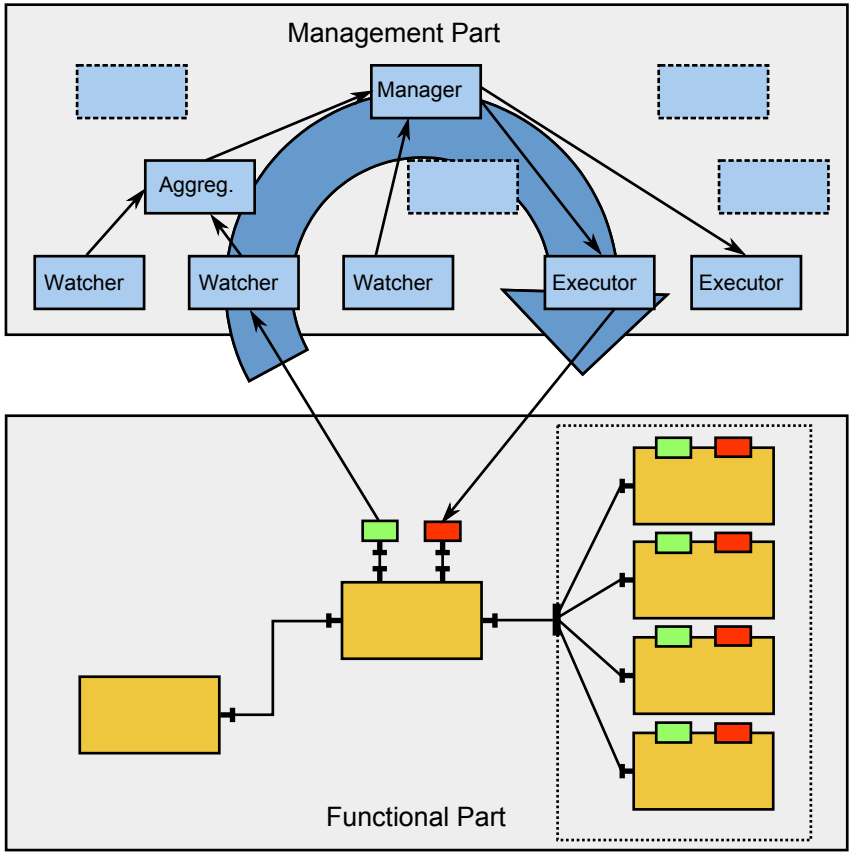
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API



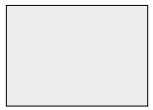
Management Part

- Management Elements
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Communicate through events
- Publish/Subscribe
- Autonomic Managers (control loops) built as network of MEs
- Sensors and Actuators for components and groups
- Actuation API



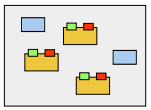
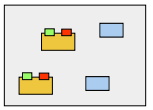
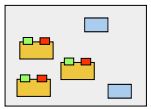
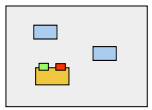
Runtime Environment

- Containers that host components and MEs
- Use a Structured Overlay Network for communication
- Provide overlay services



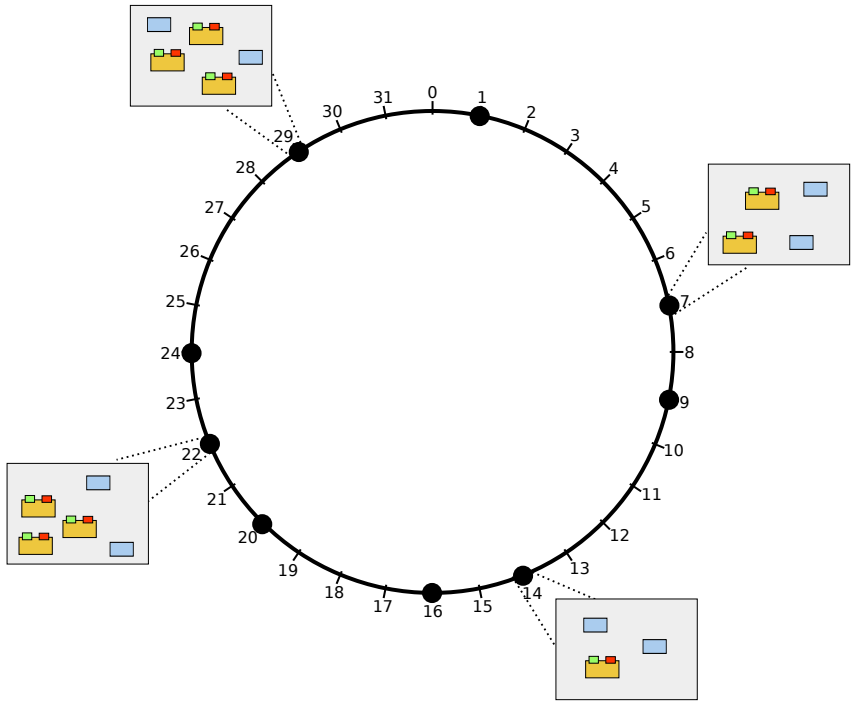

Runtime Environment

- Containers that host components and MEs
- Use a Structured Overlay Network for communication
- Provide overlay services

Runtime Environment

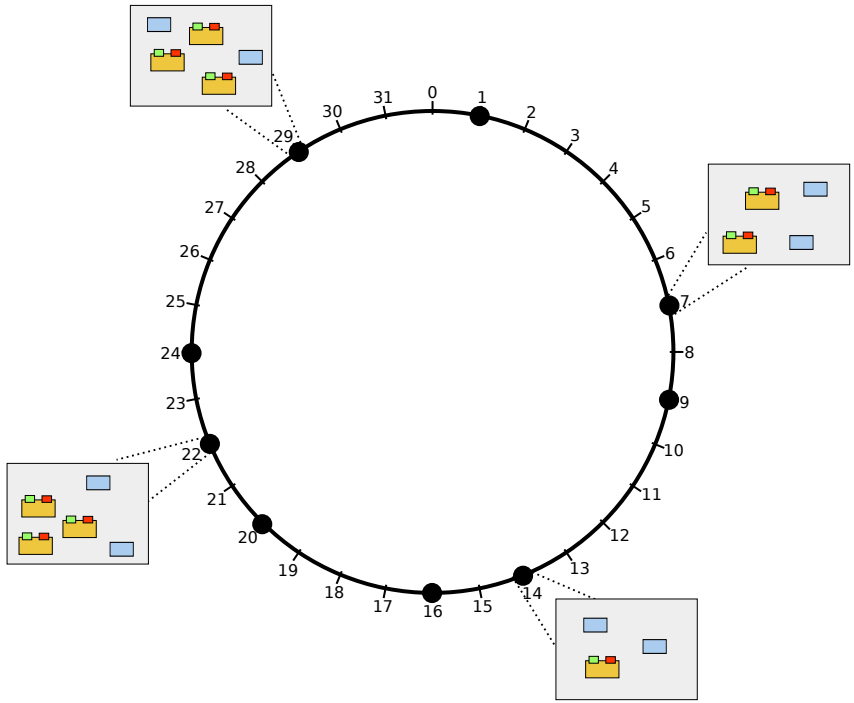
- Containers that host components and MEs
- Use a Structured Overlay Network for communication
- Provide overlay services



Dealing with Resource Churn

How to deal with failures?

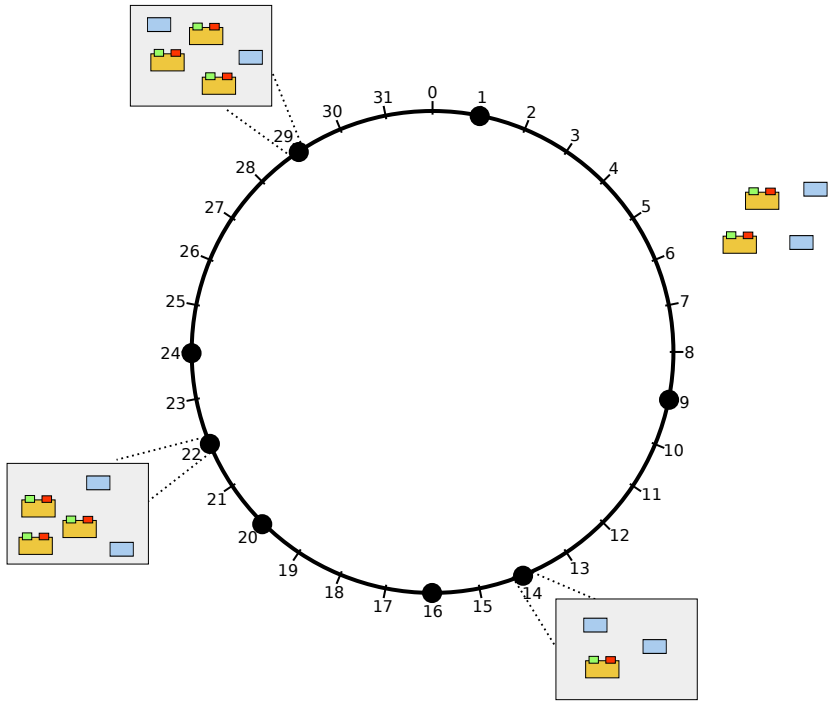
- MEs heal the functional part
- How to heal failed MEs?
  - Programmatically in the management logic
  - Transparently by the platform



Dealing with Resource Churn

How to deal with failures?

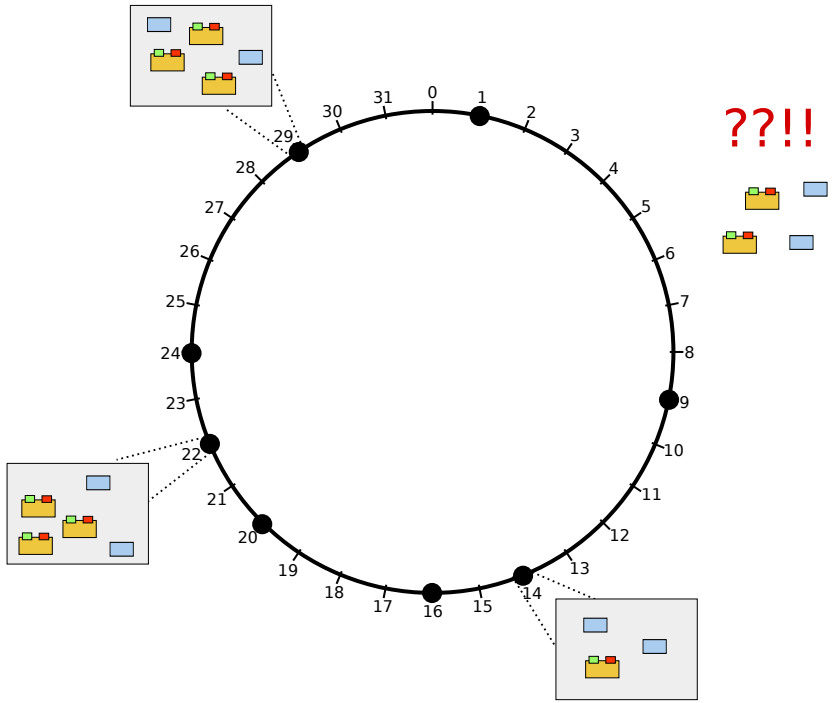
- MEs heal the functional part
- How to heal failed MEs?
  - Programmatically in the management logic
  - Transparently by the platform



Dealing with Resource Churn

How to deal with failures?

- MEs heal the functional part
- How to heal failed MEs?
  - Programmatically in the management logic
  - Transparently by the platform



Outline

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Robust Management Elements

A Robust Management Element (RME):

- is replicated to ensure fault-tolerance
- tolerates continuous churn by automatically restoring failed replicas on other nodes
- maintains its state consistent among replicas
- provides its service with minimal disruption in spite of resource churn (high availability)
- is location transparent, i.e., RME clients communicate with it regardless of current location of its replicas

Solution Outline

- Replicated state machine
- An algorithm to reconfigure the replicated state machine. (We used the SMART algorithm)
- Our decentralized algorithm to automate reconfiguration

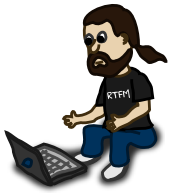
Solution Outline

- Replicated state machine
- An algorithm to reconfigure the replicated state machine. (We used the SMART algorithm)
- Our decentralized algorithm to automate reconfiguration

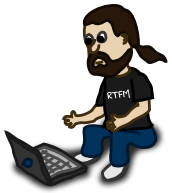
Solution Outline

- Replicated state machine
- An algorithm to reconfigure the replicated state machine. (We used the SMART algorithm)
- Our decentralized algorithm to automate reconfiguration

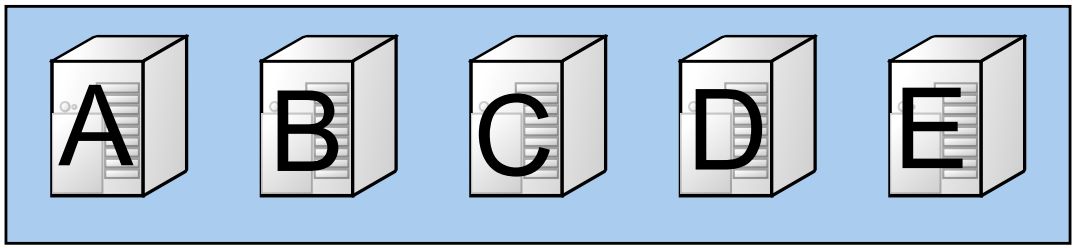
SMART



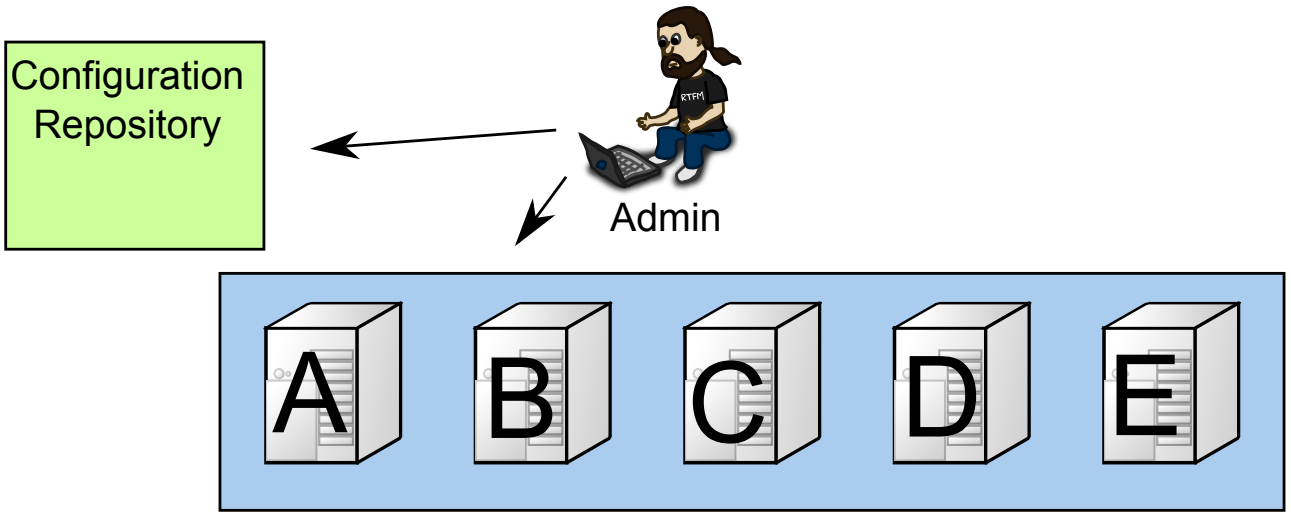
SMART



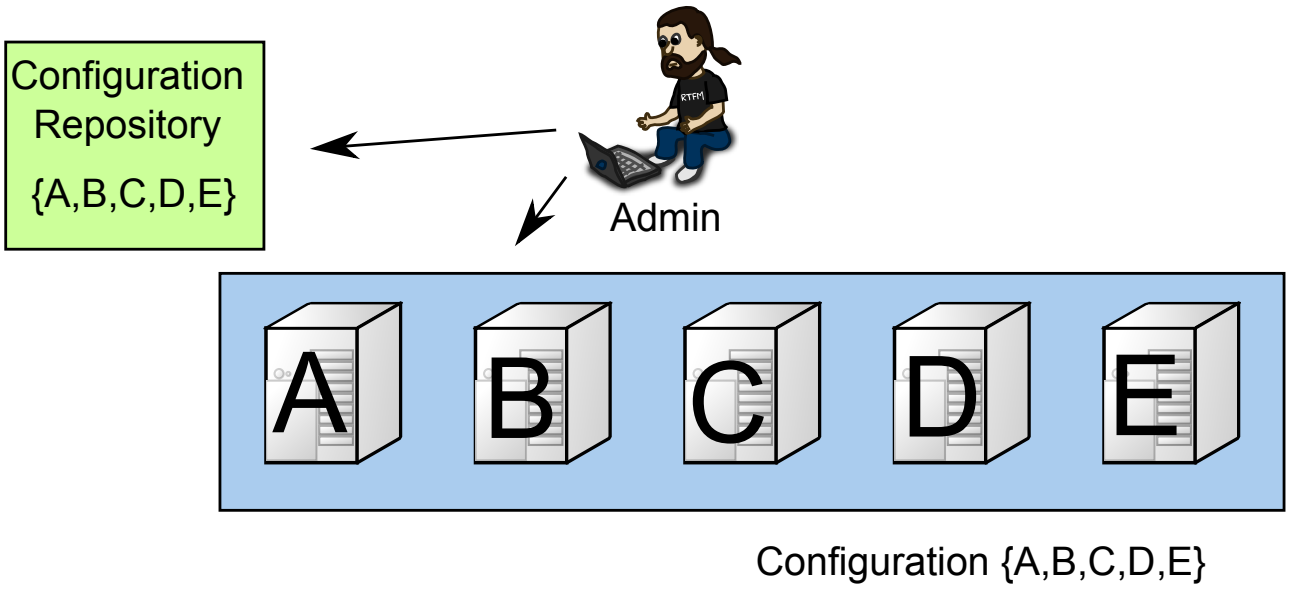
Admin



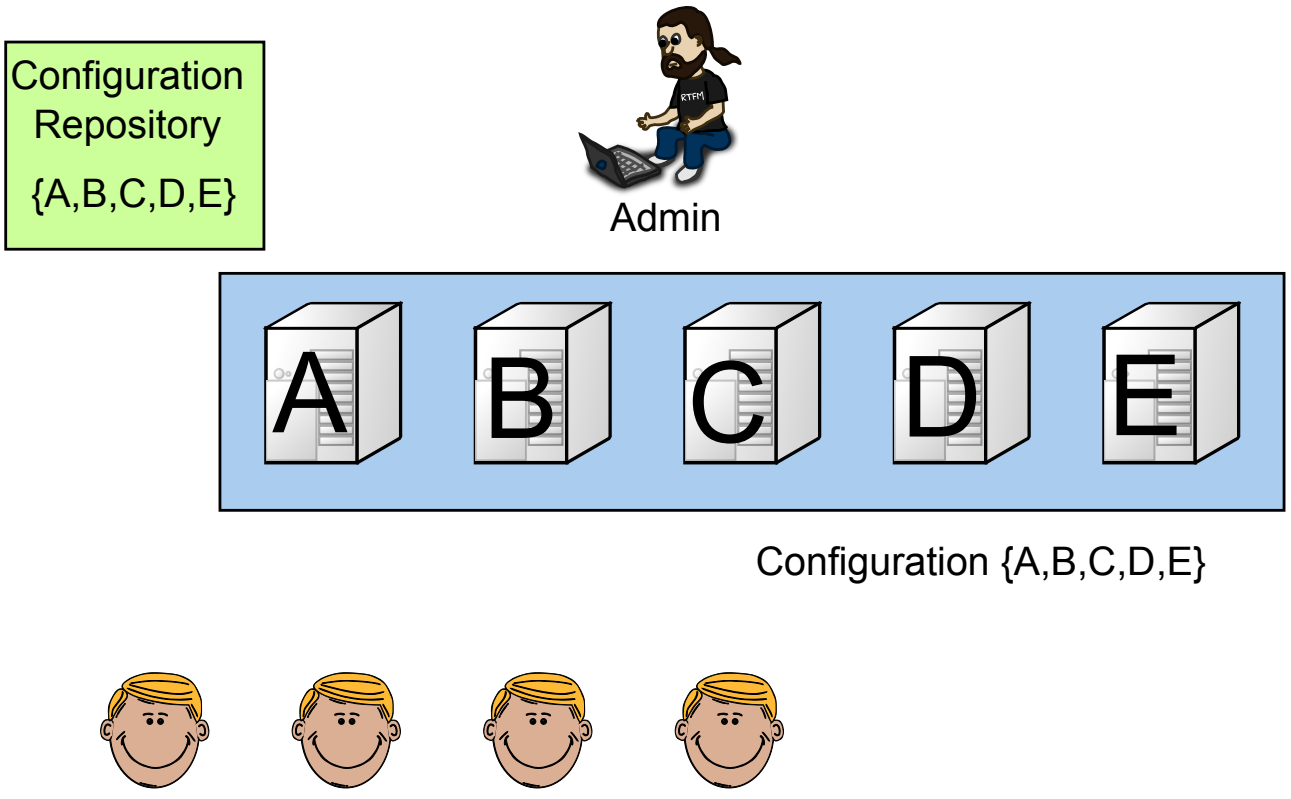
SMART



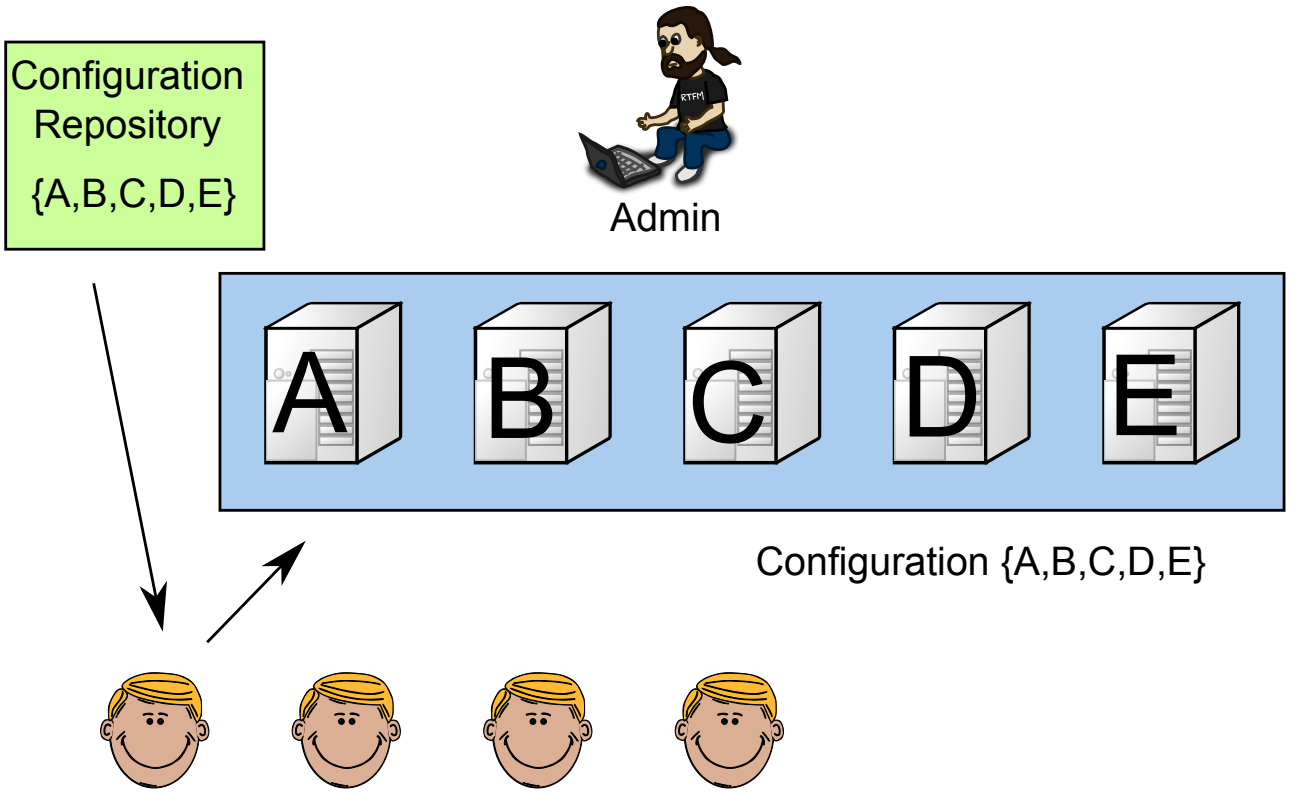
SMART


Configuration Repository
{A,B,C,D,E}
Admin
A
B
C
D
E
Configuration {A,B,C,D,E}

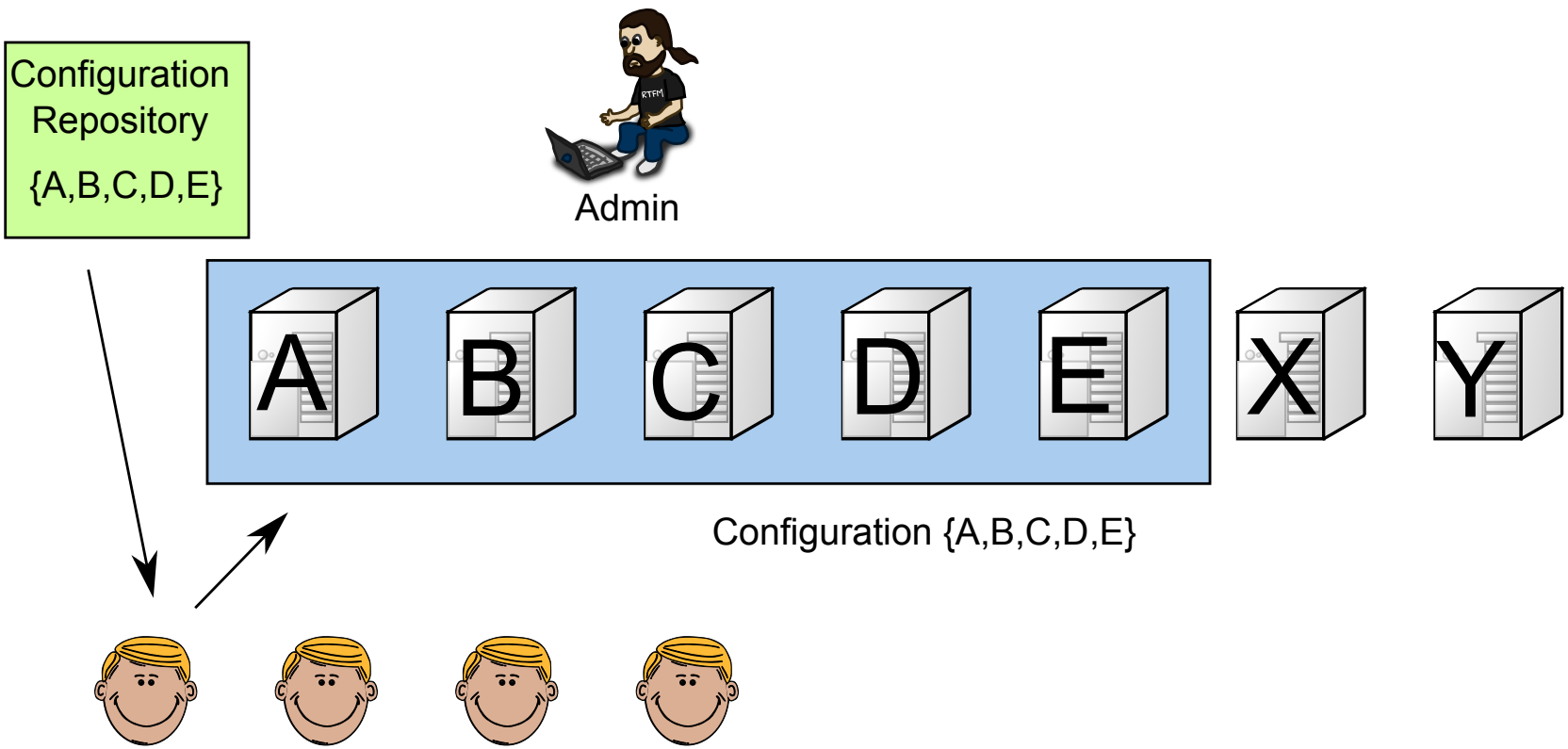
SMART


Configuration Repository
{A,B,C,D,E}
Admin
A
B
C
D
E
Configuration {A,B,C,D,E}

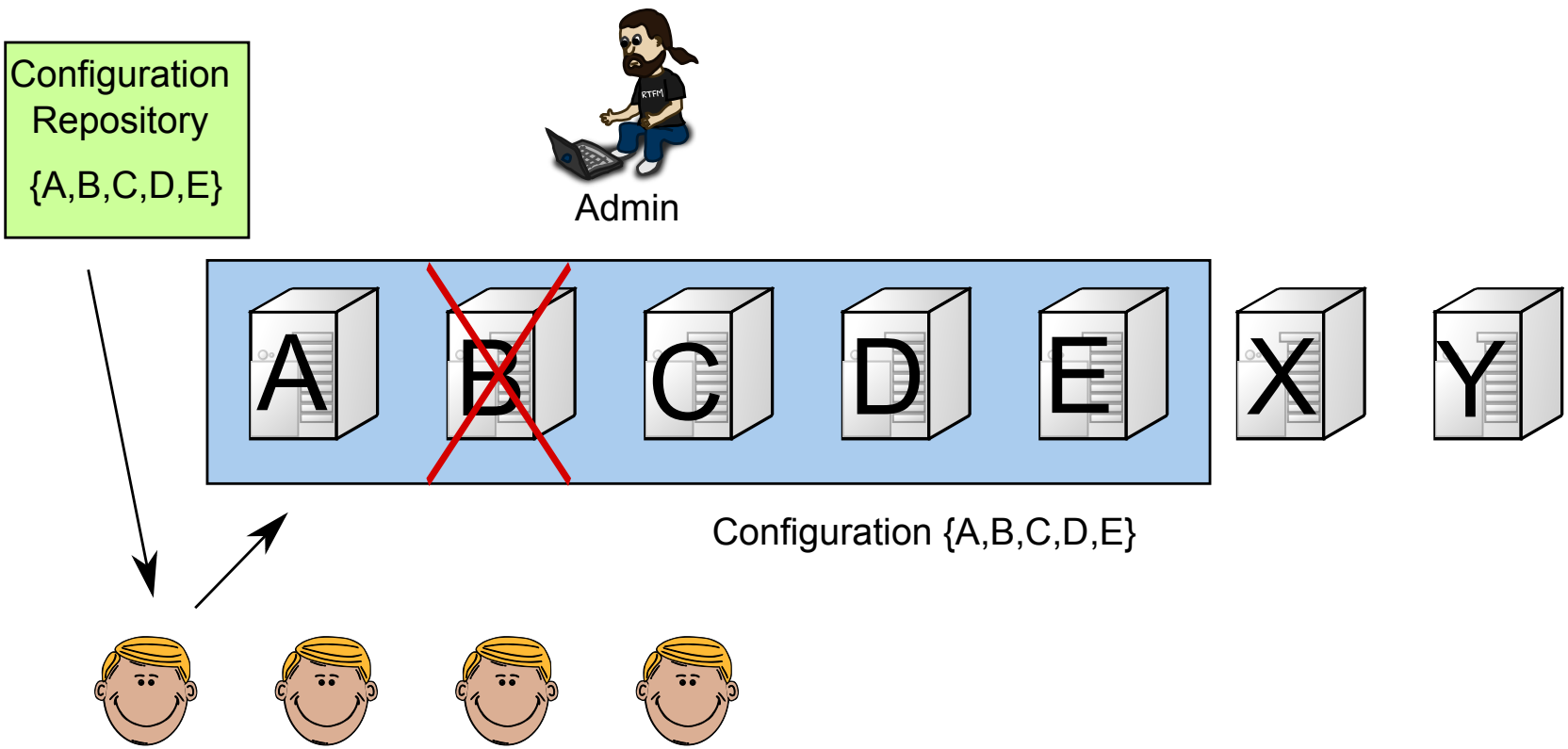
SMART


Configuration Repository
{A,B,C,D,E}
Admin
A
B
C
D
E
Configuration {A,B,C,D,E}

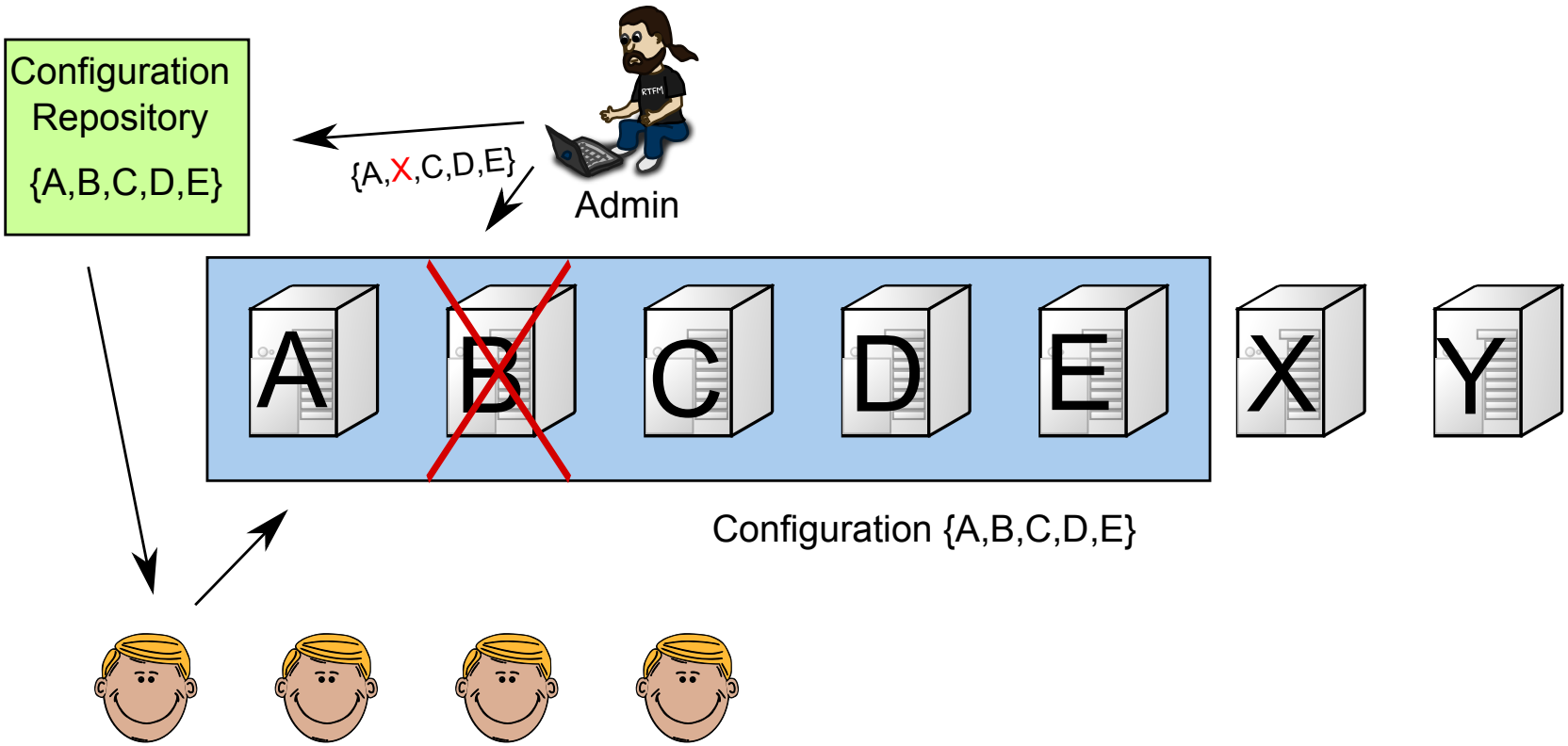
SMART


Configuration Repository
{A,B,C,D,E}
Admin
A
B
C
D
E
X
Y
Configuration {A,B,C,D,E}

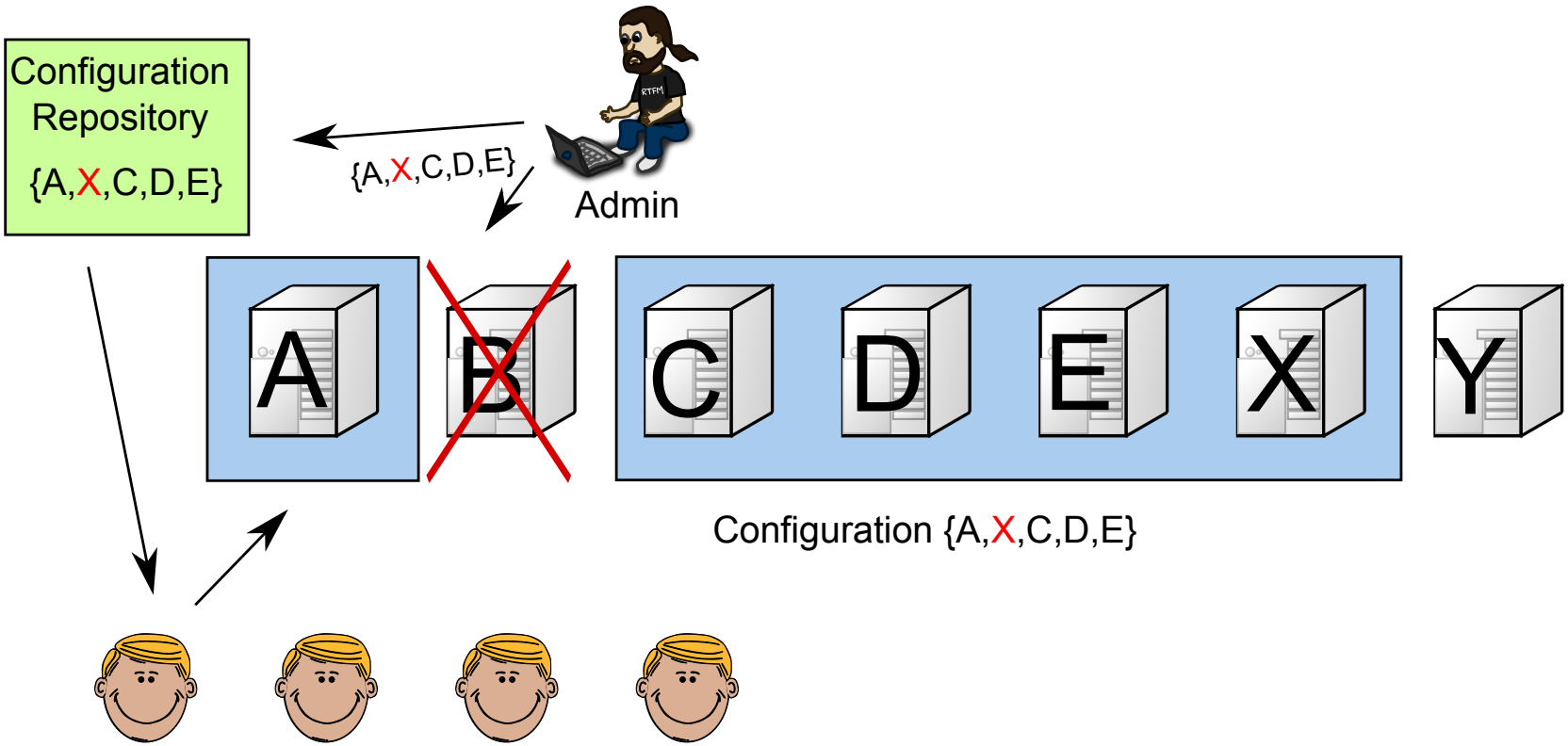
SMART


Configuration Repository
{A,B,C,D,E}
Admin
A
B
C
D
E
X
Y
Configuration {A,B,C,D,E}

SMART

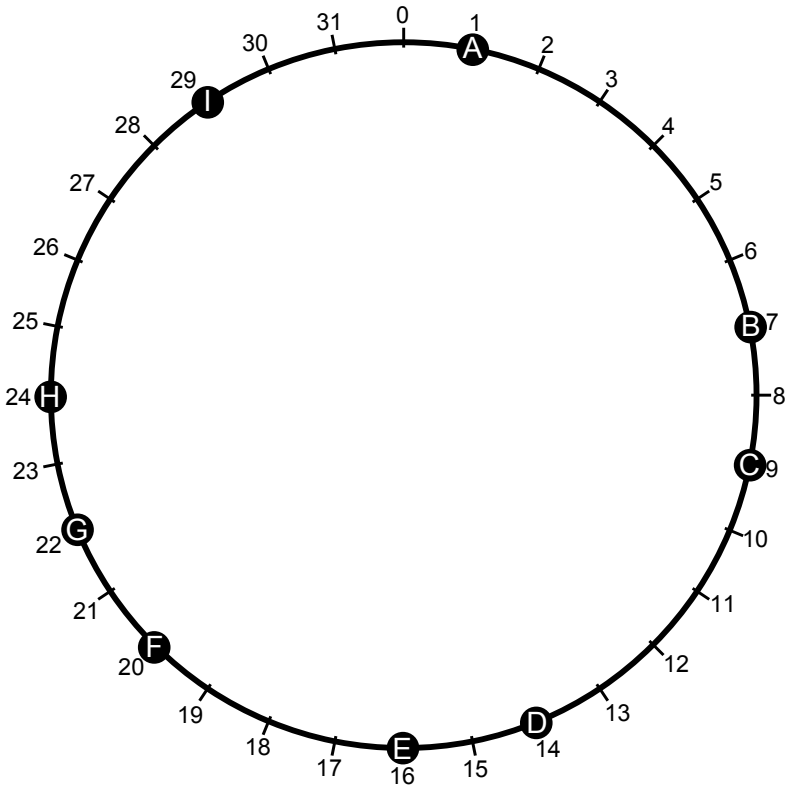

Configuration Repository
{A,B,C,D,E}
{A,X,C,D,E}
Admin
A
B
C
D
E
X
Y
Configuration {A,B,C,D,E}

SMART


Configuration Repository
{A,X,C,D,E}
{A,X,C,D,E}
Admin
A
B
C
D
E
X
Y
Configuration {A,X,C,D,E}

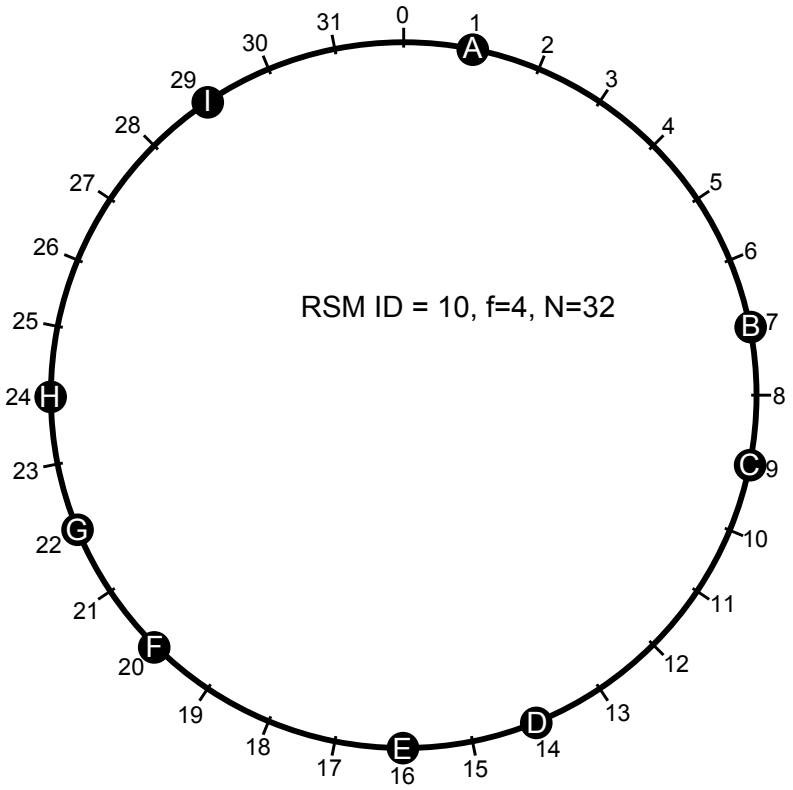
Creating a Replicated State Machine (RSM)

Any node can create a RSM. Select ID and replication degree



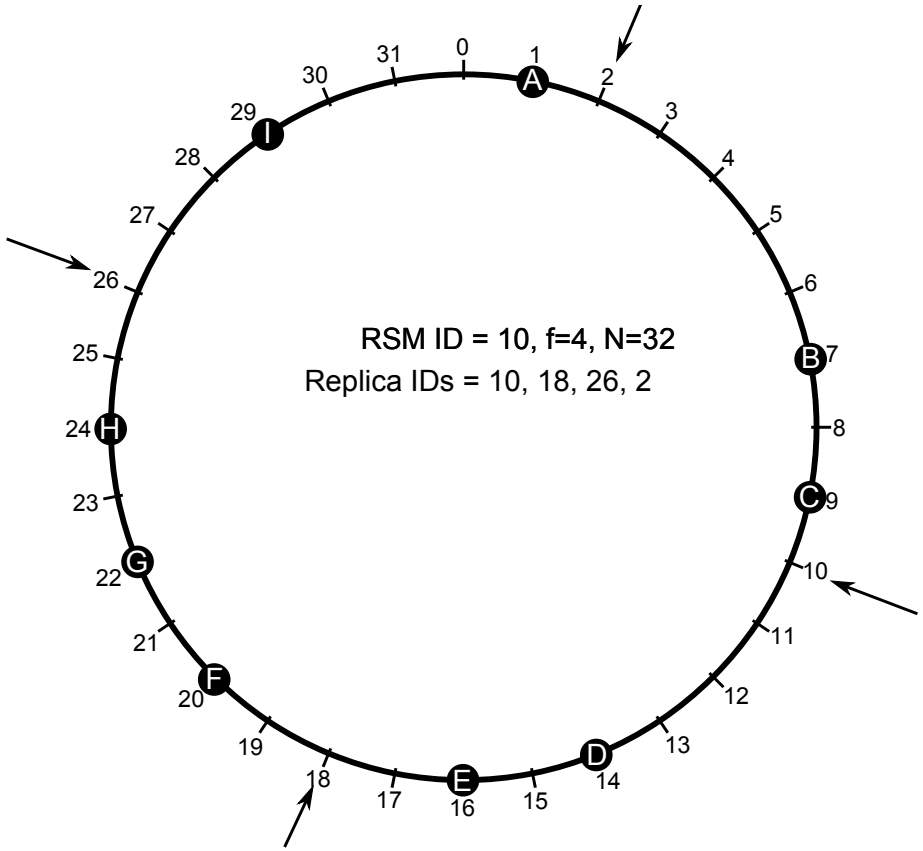
Creating a Replicated State Machine (RSM)

Any node can create a RSM. Select ID and replication degree



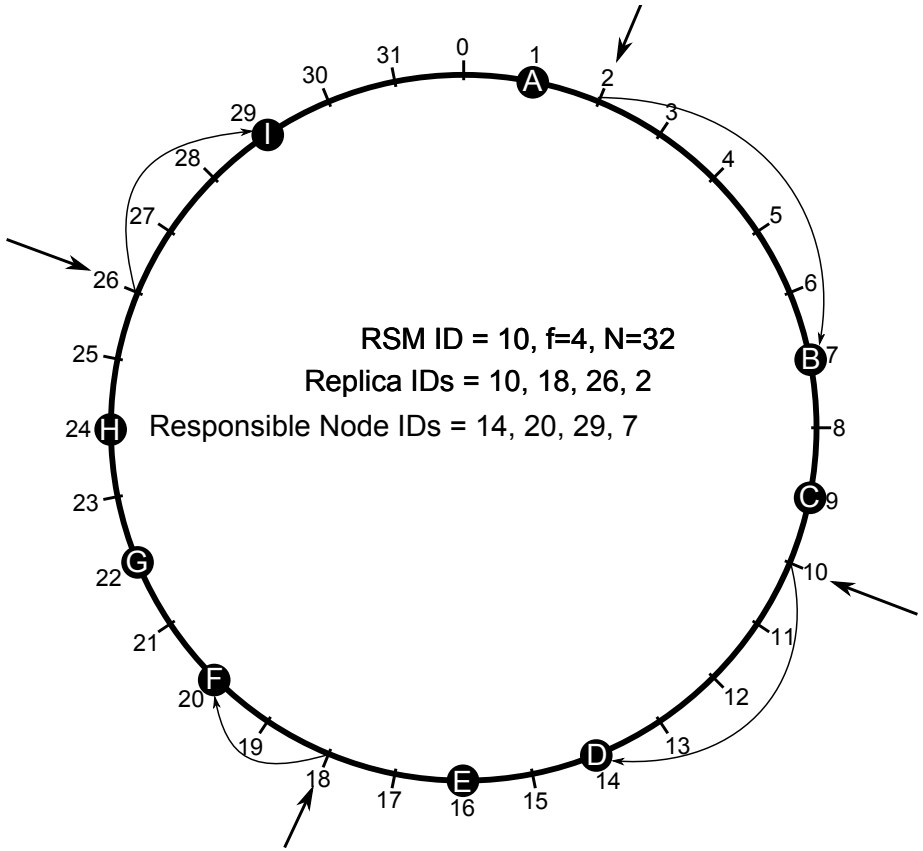
Creating a Replicated State Machine (RSM)

The node uses symmetric replication scheme to calculate replica IDs



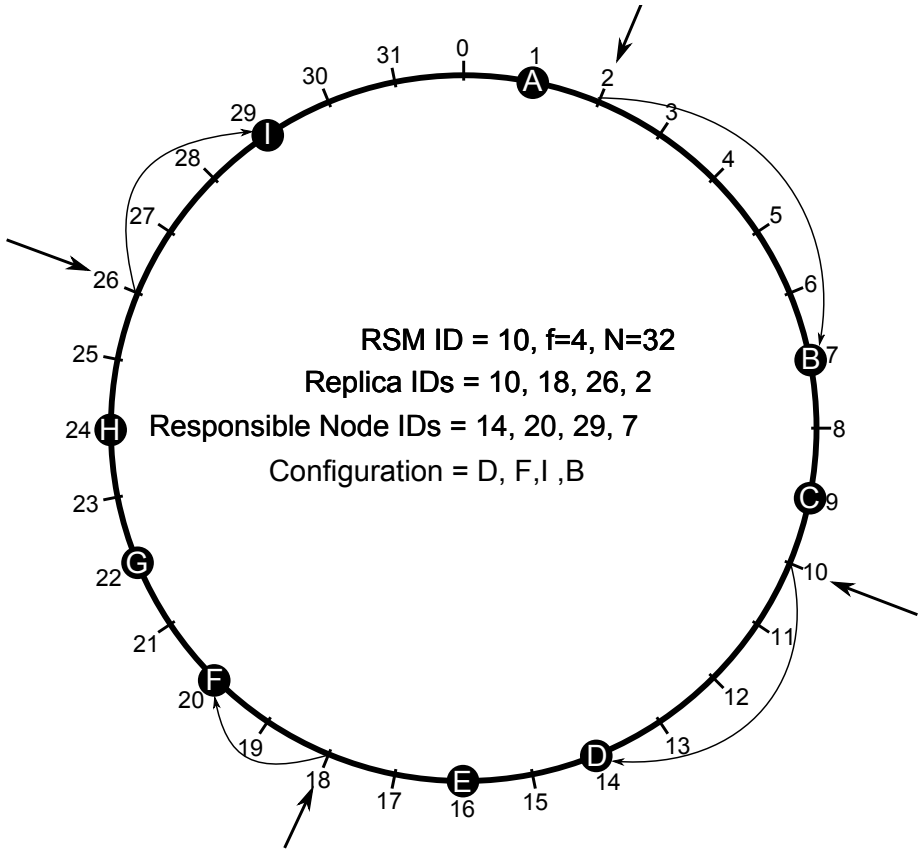
Creating a Replicated State Machine (RSM)

The node uses lookups to find responsible nodes . . .



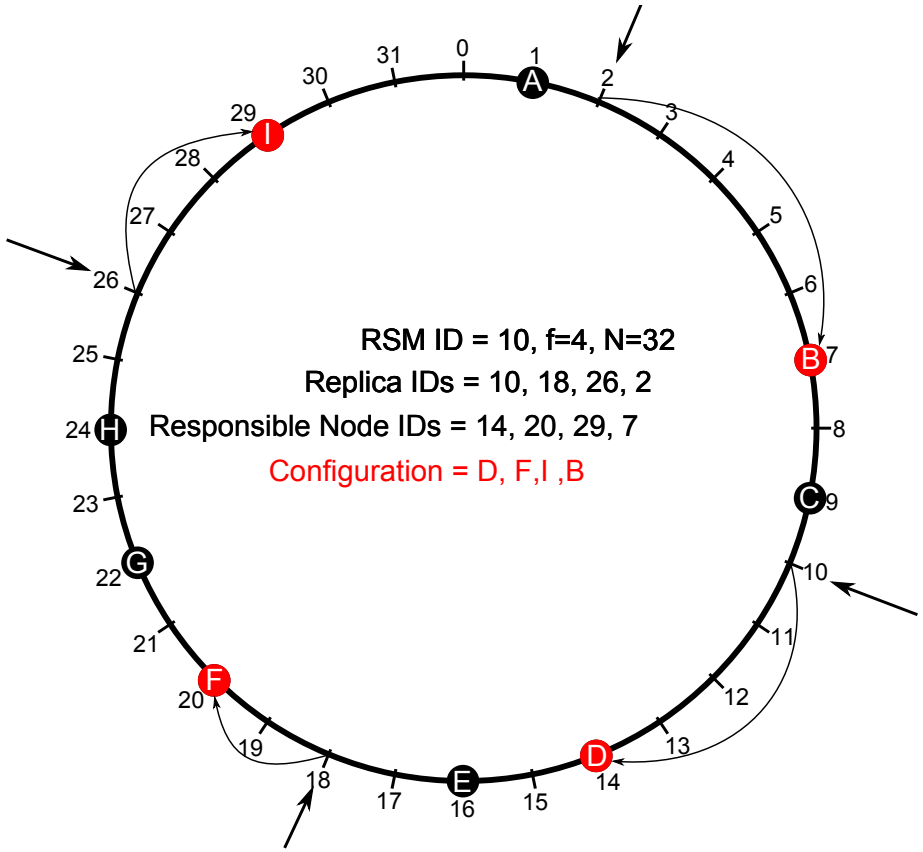
Creating a Replicated State Machine (RSM)

. . . and gets direct references to them



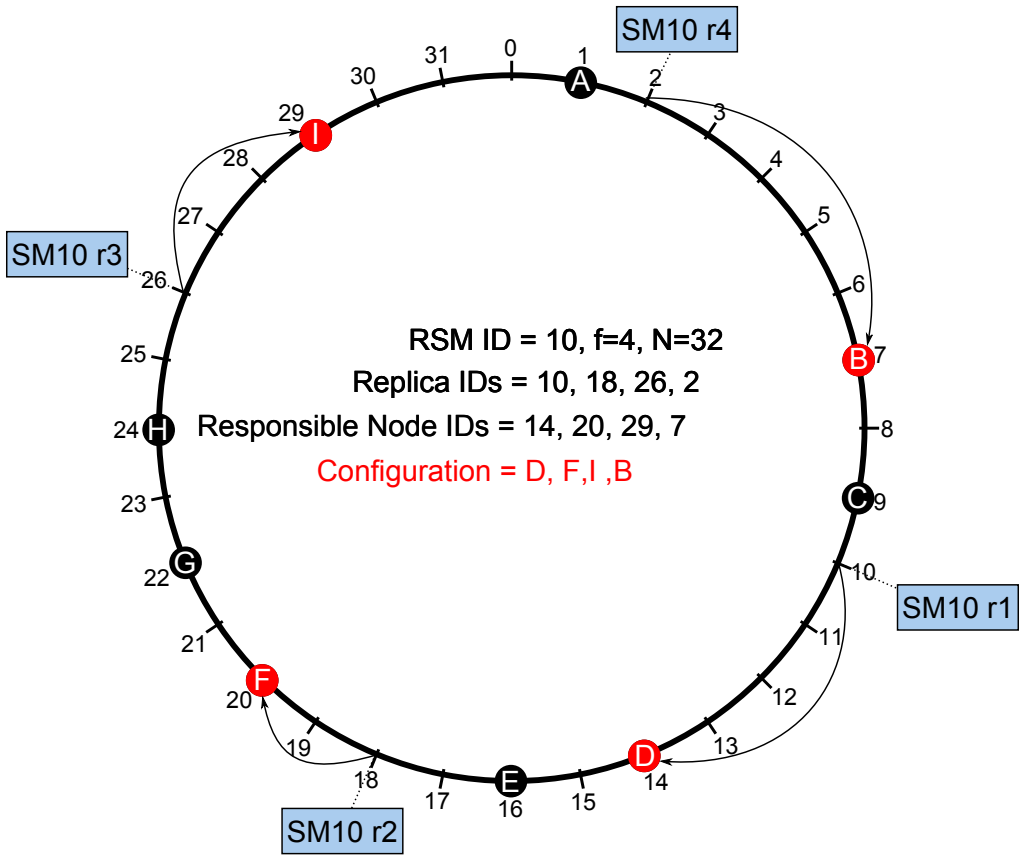
Creating a Replicated State Machine (RSM)

The set of direct references forms the configuration



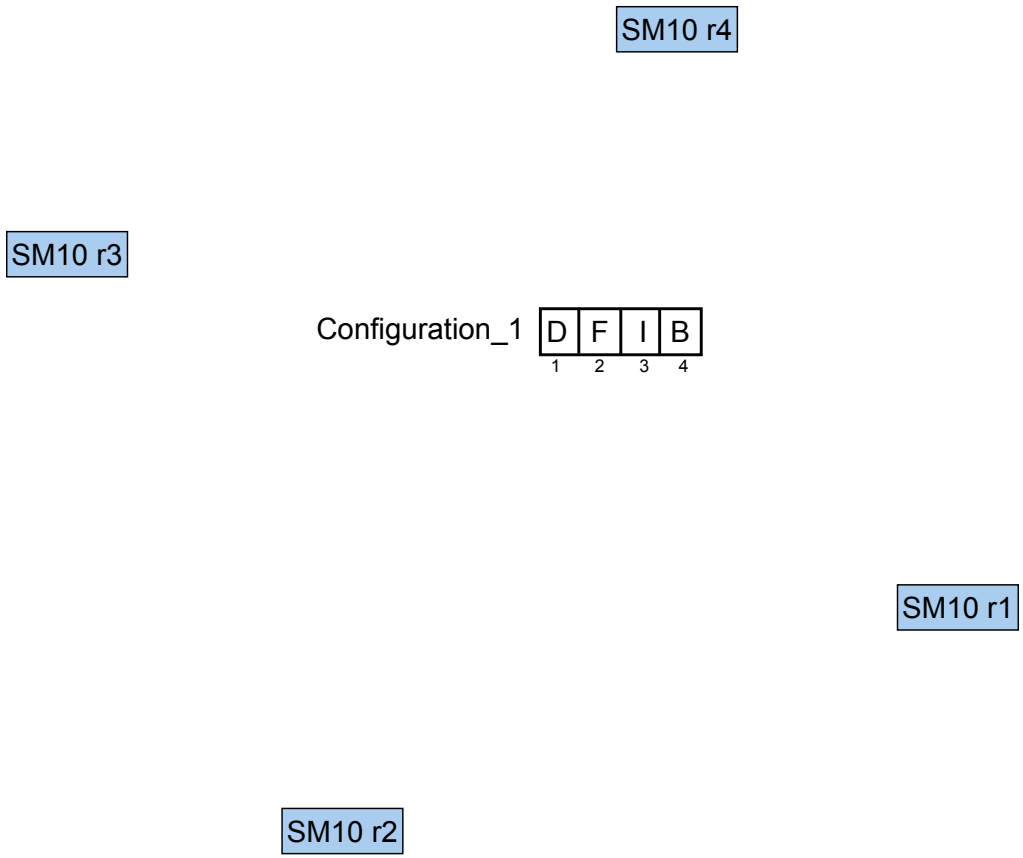
Creating a Replicated State Machine (RSM)

The node sends a *Create* message to the configuration

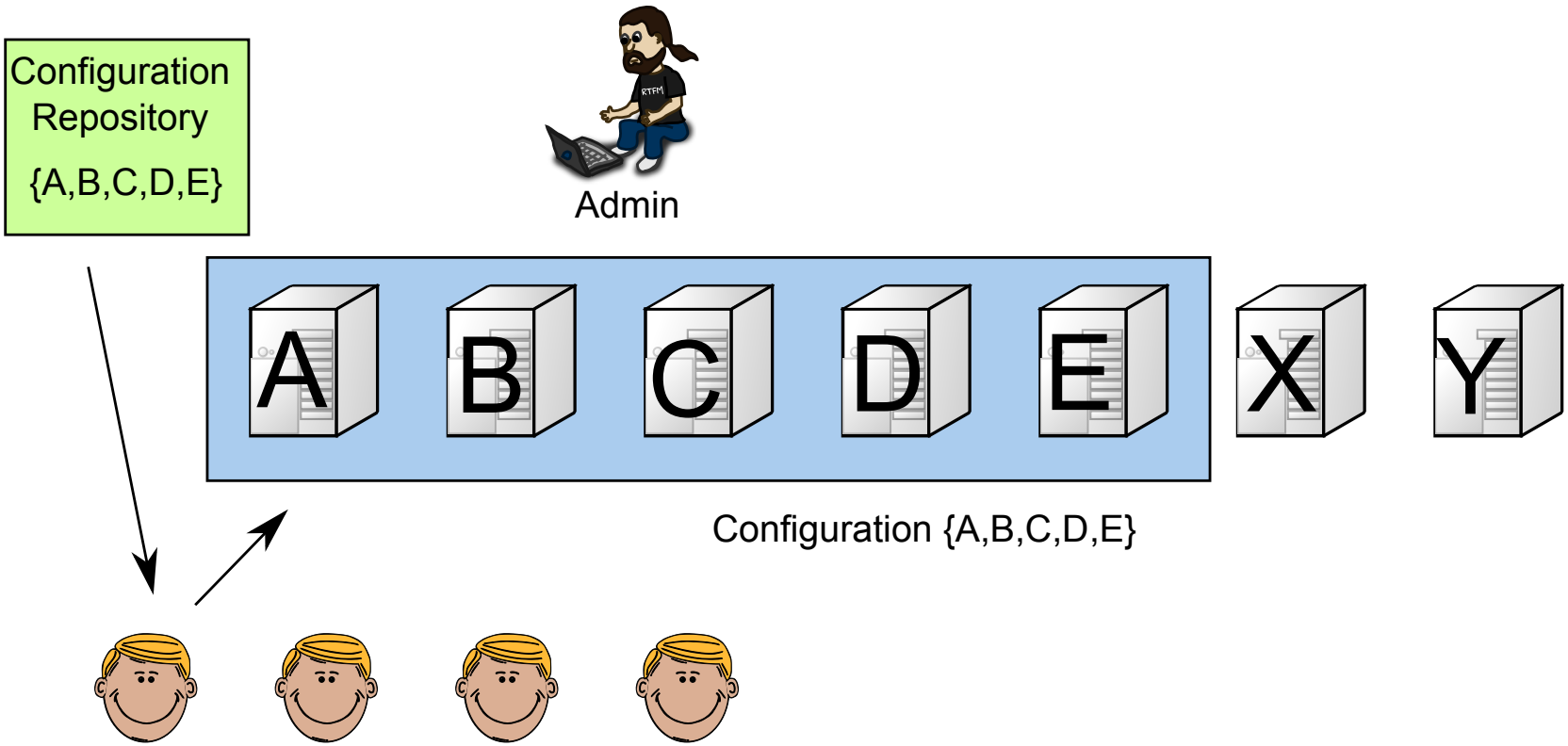


Creating a Replicated State Machine (RSM)

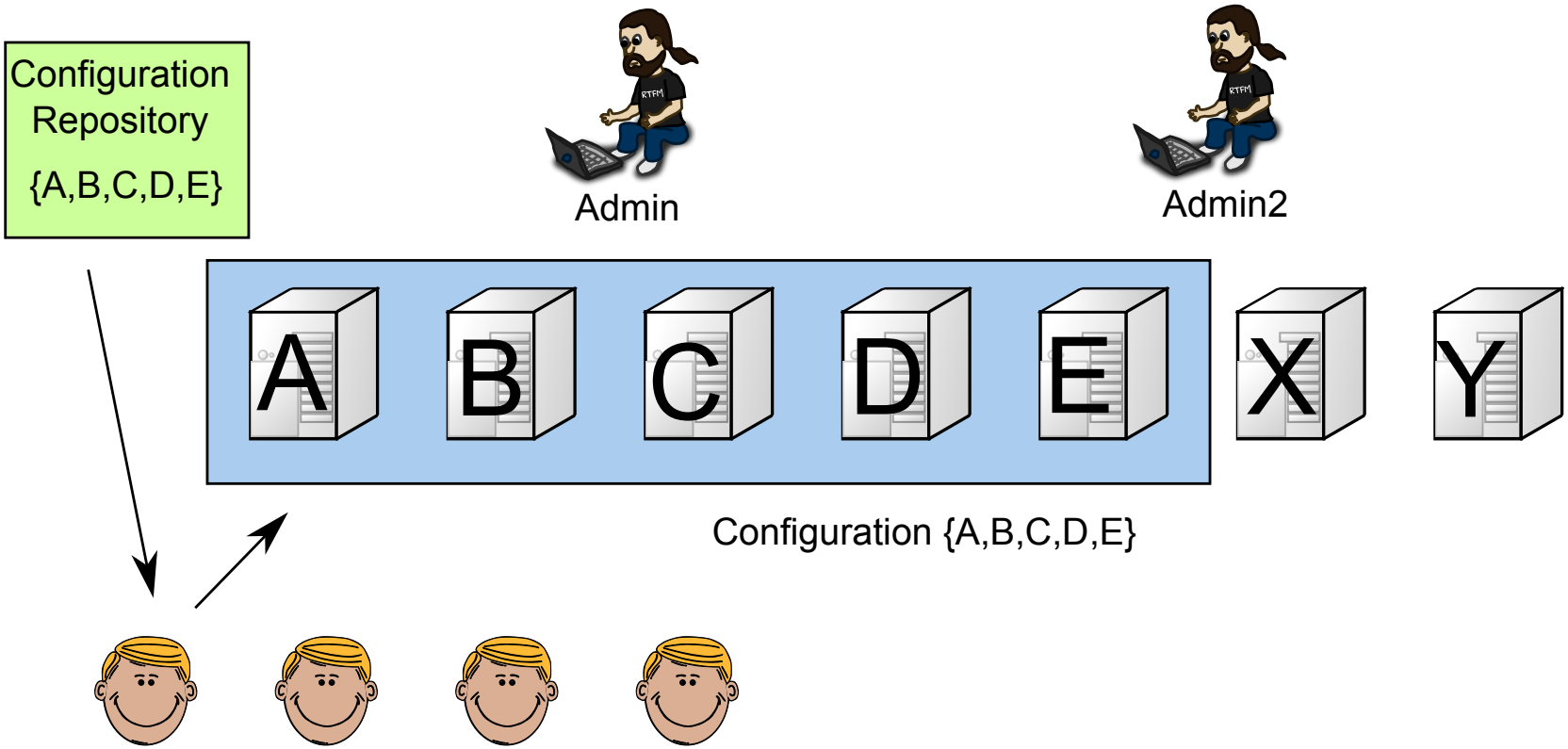
Now replicas communicate directly using the configuration



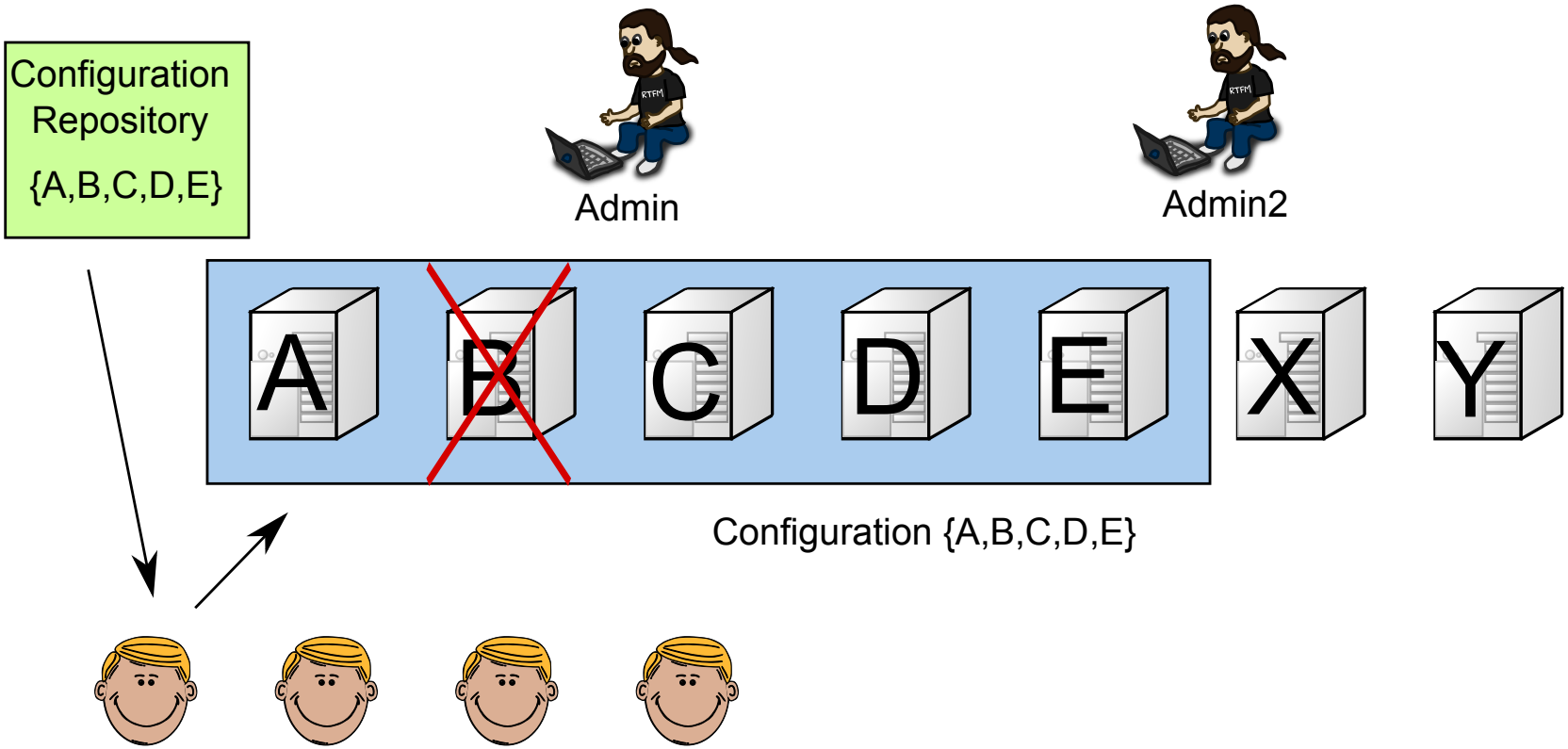
SMART with Multiple Admins



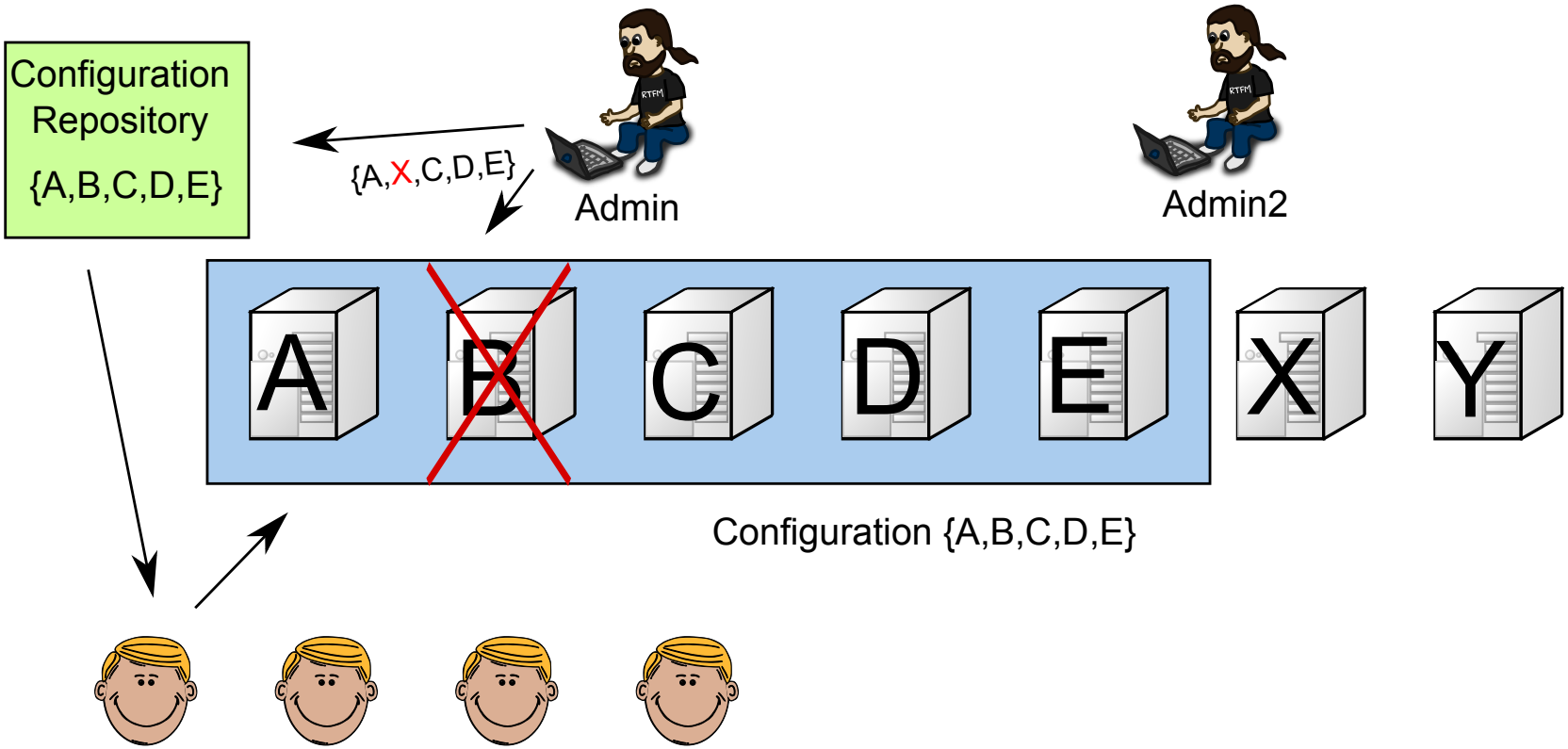
SMART with Multiple Admins



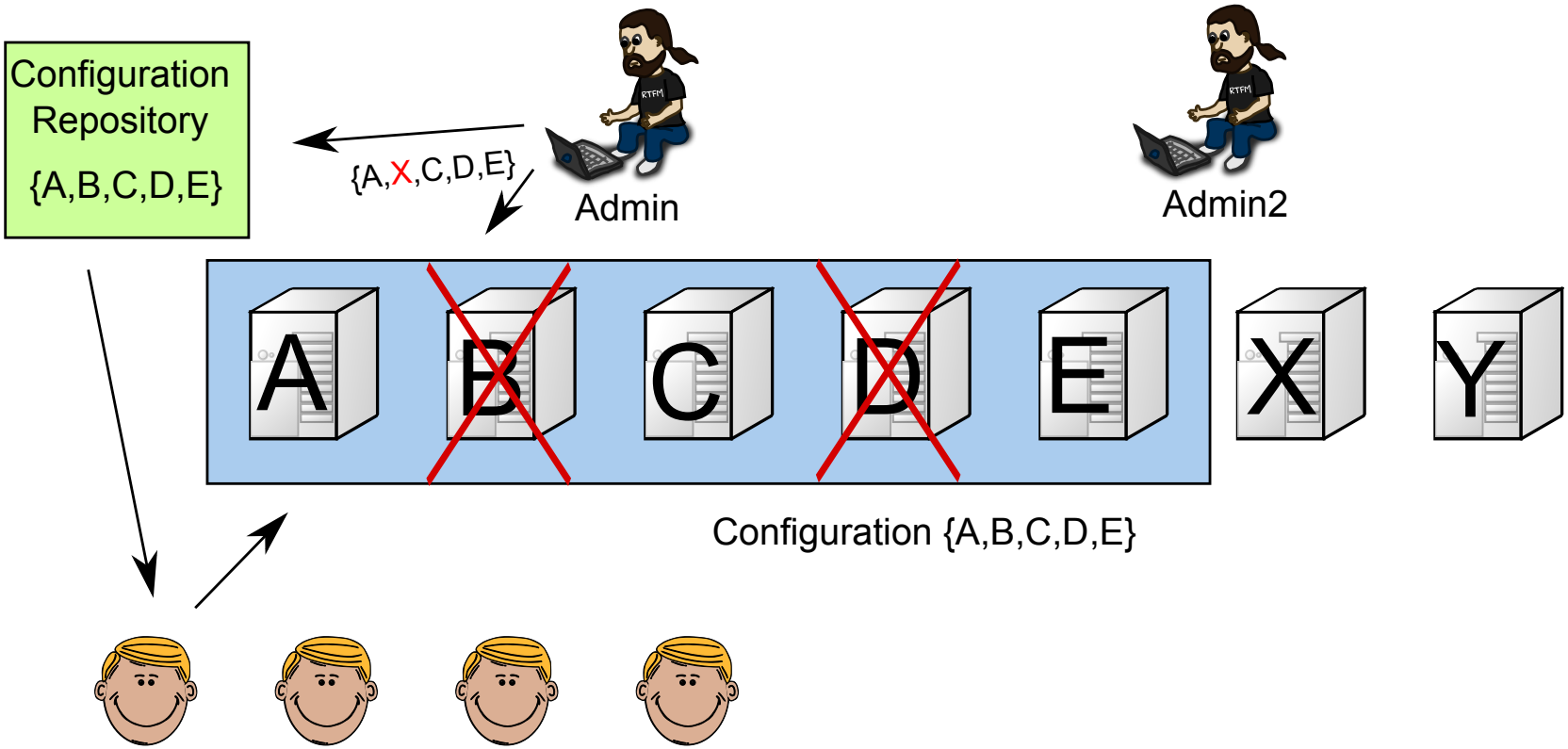
SMART with Multiple Admins



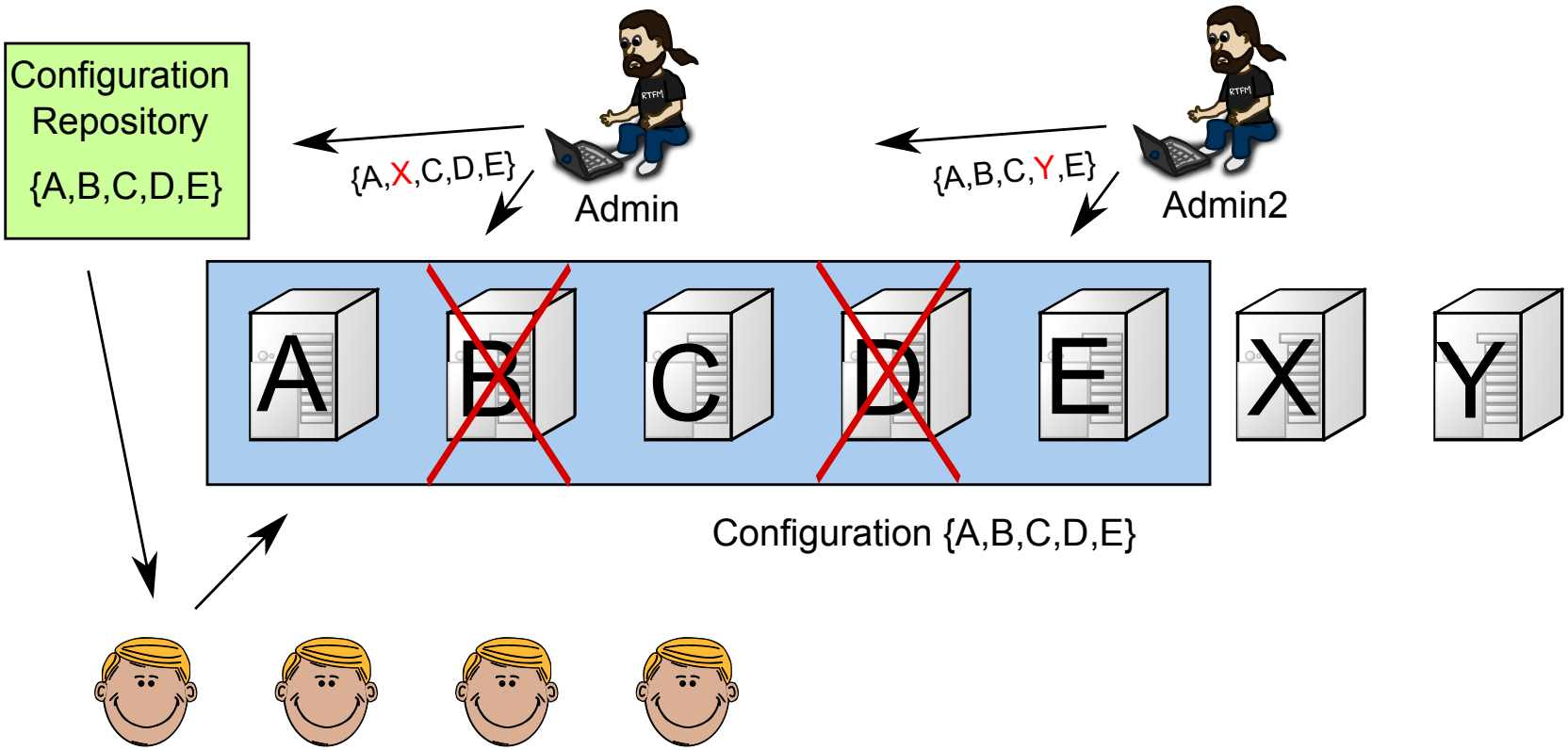
SMART with Multiple Admins



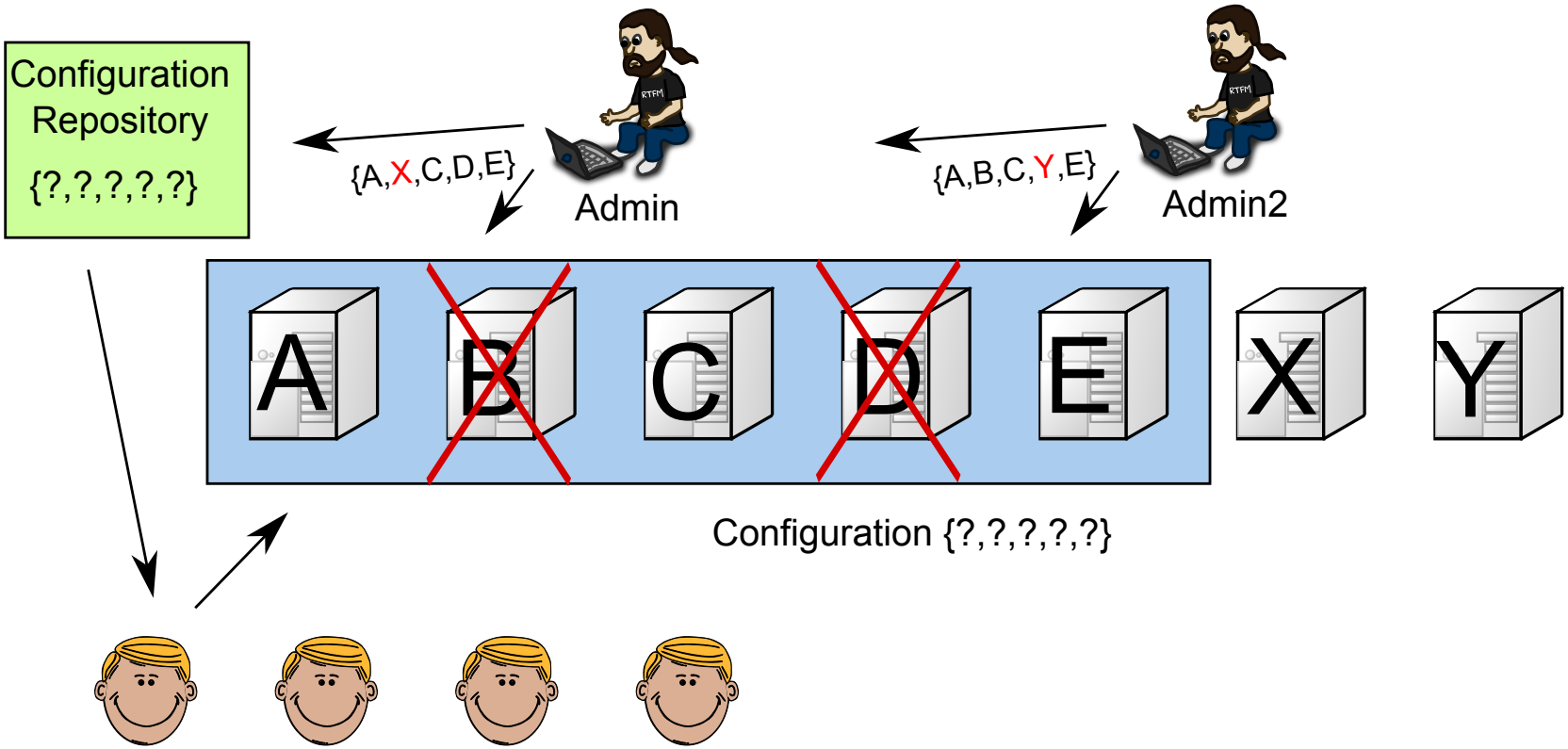
SMART with Multiple Admins



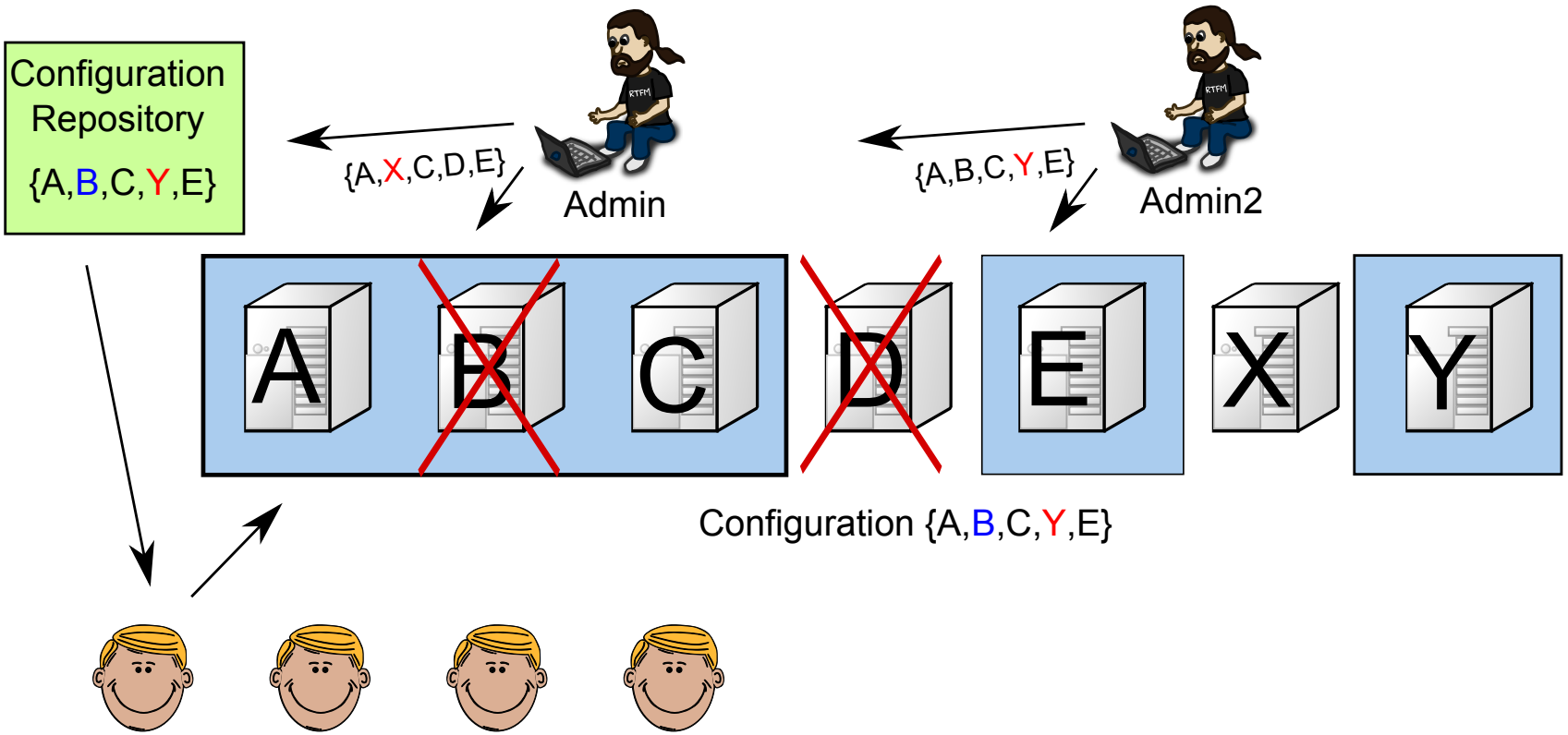
SMART with Multiple Admins



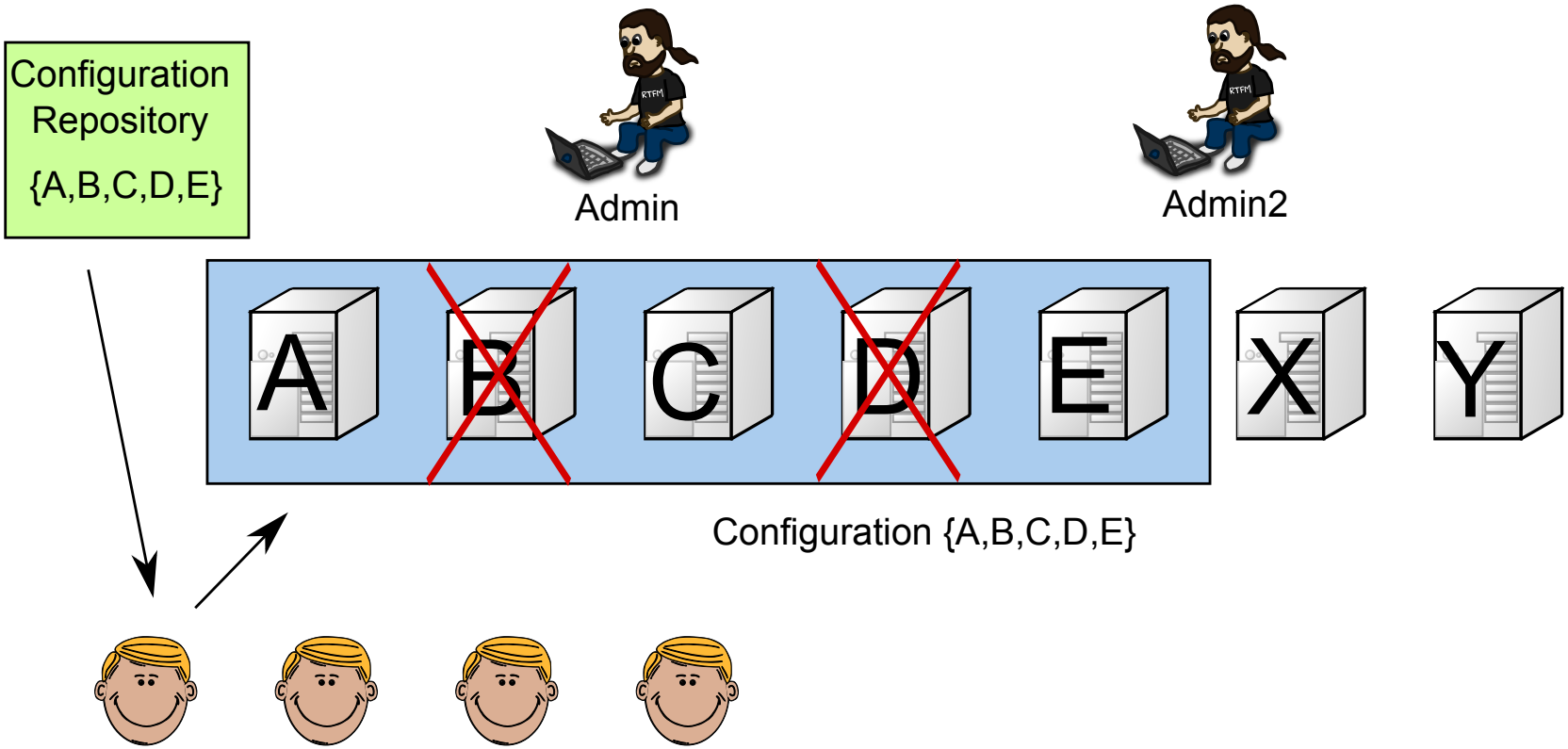
SMART with Multiple Admins



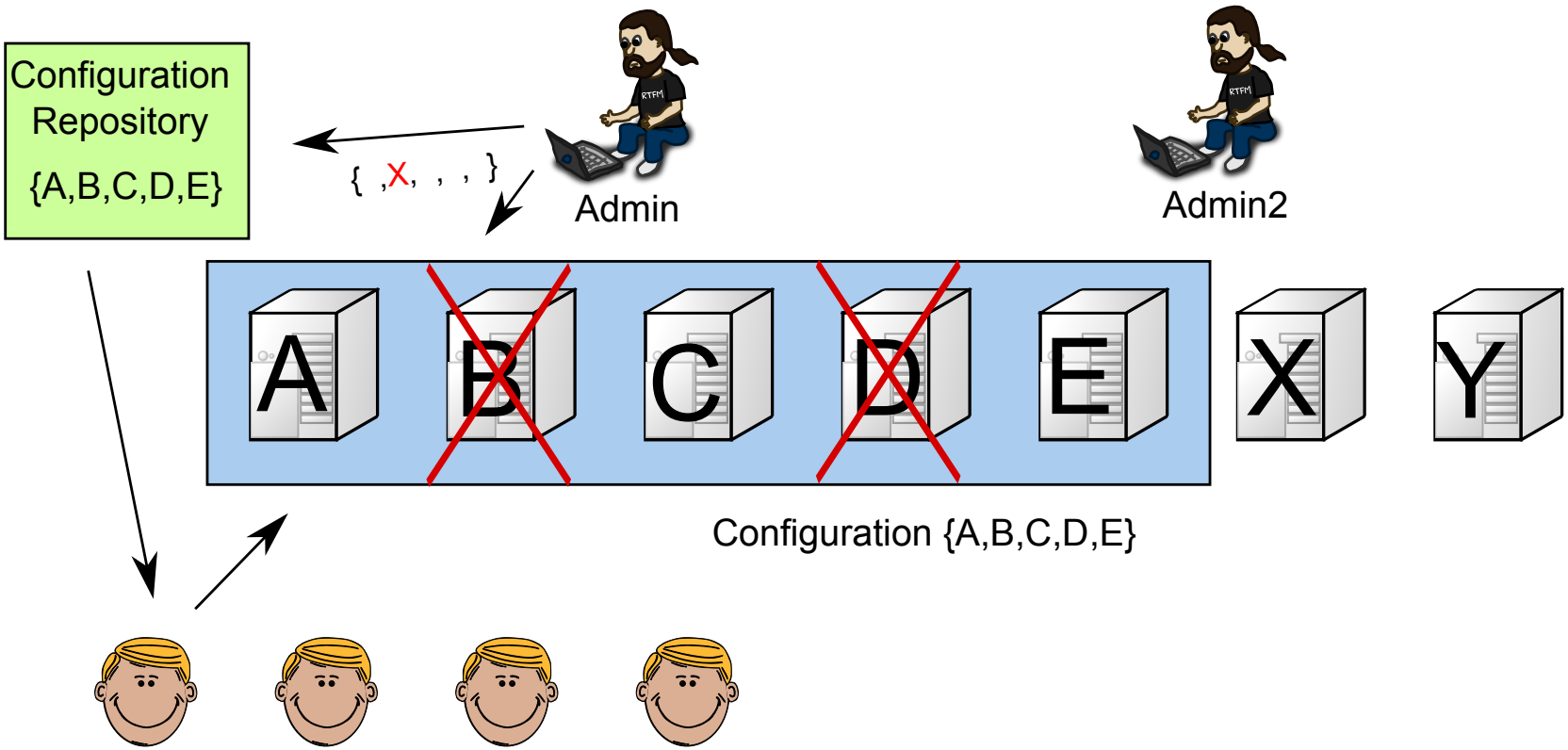
SMART with Multiple Admins



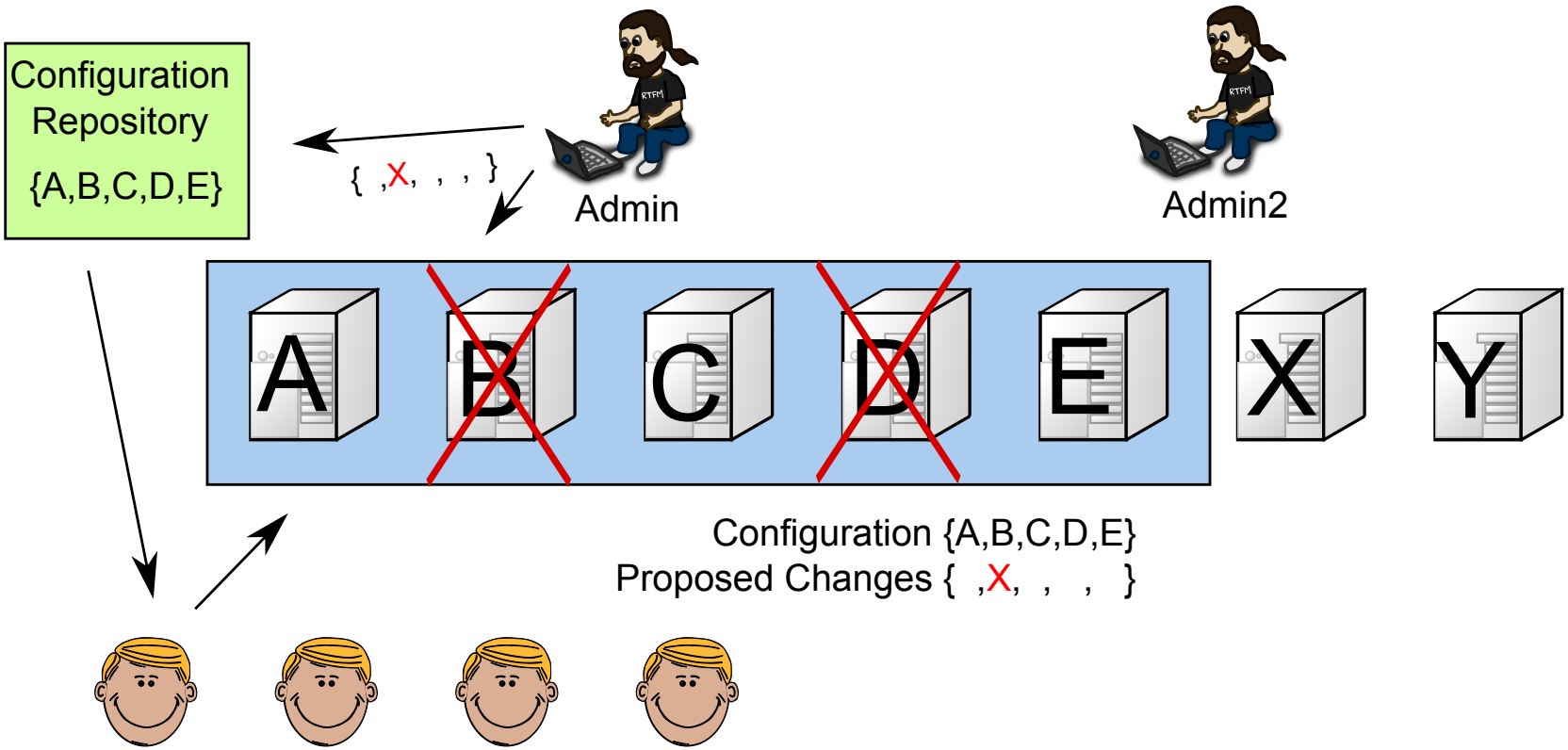
SMART with Multiple Admins



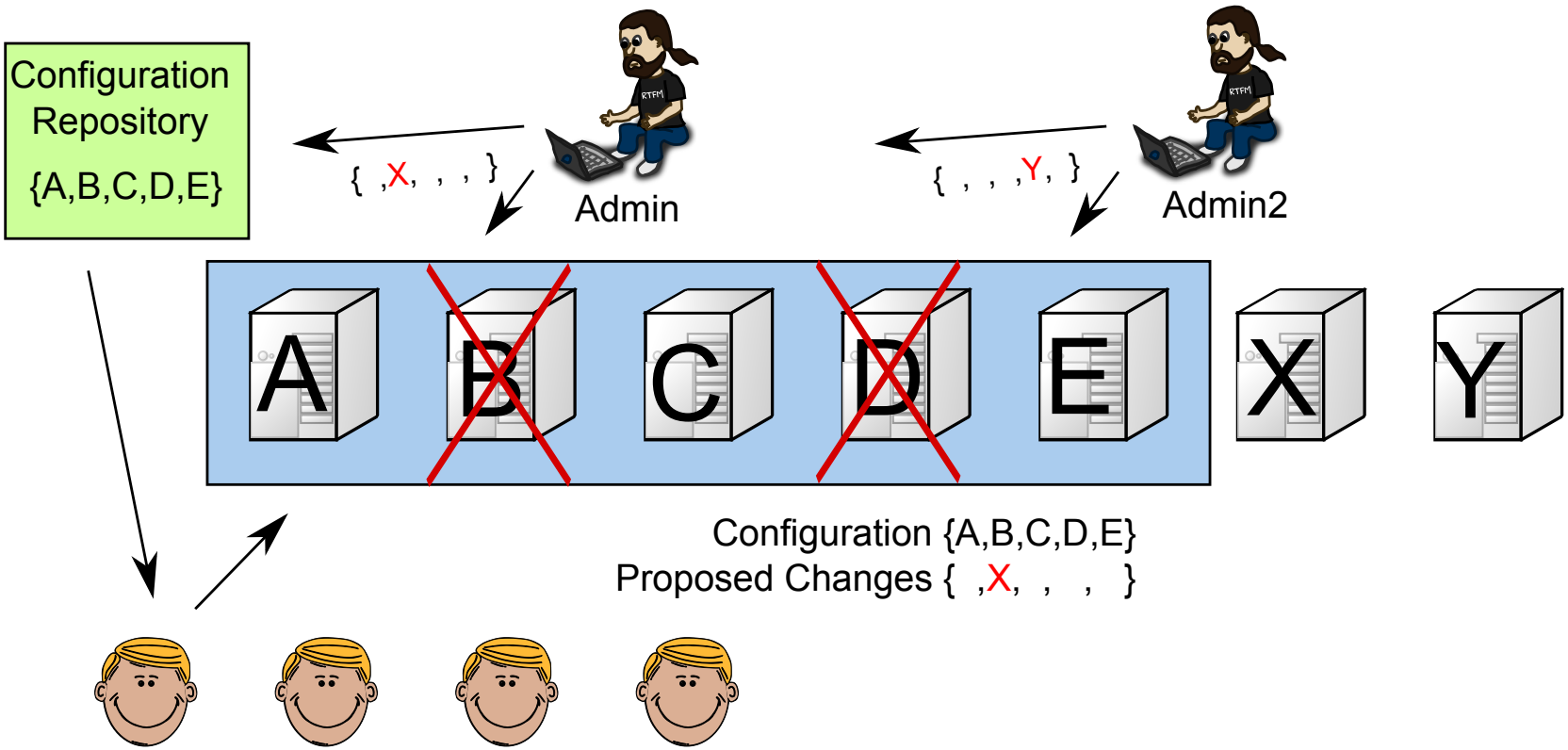
SMART with Multiple Admins



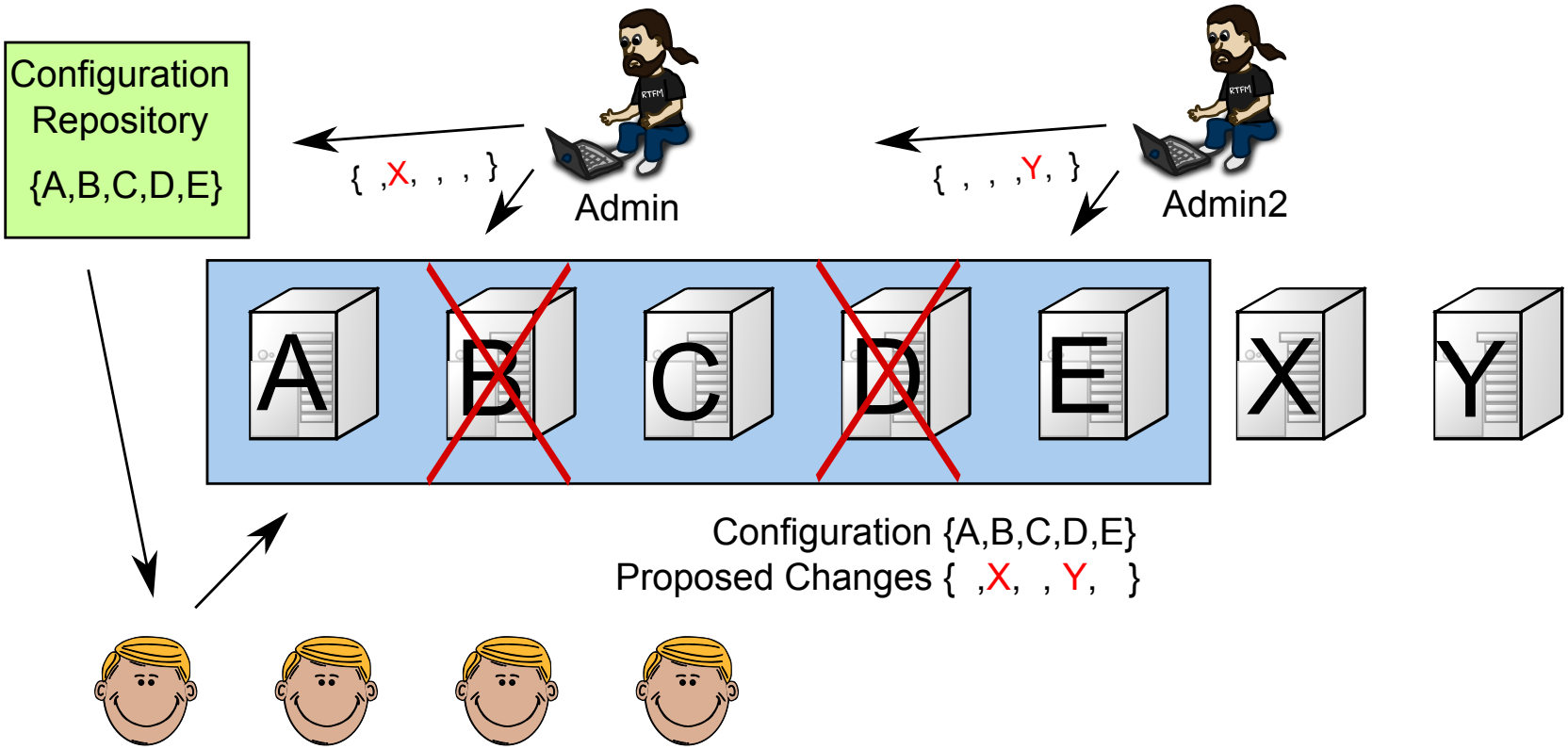
SMART with Multiple Admins



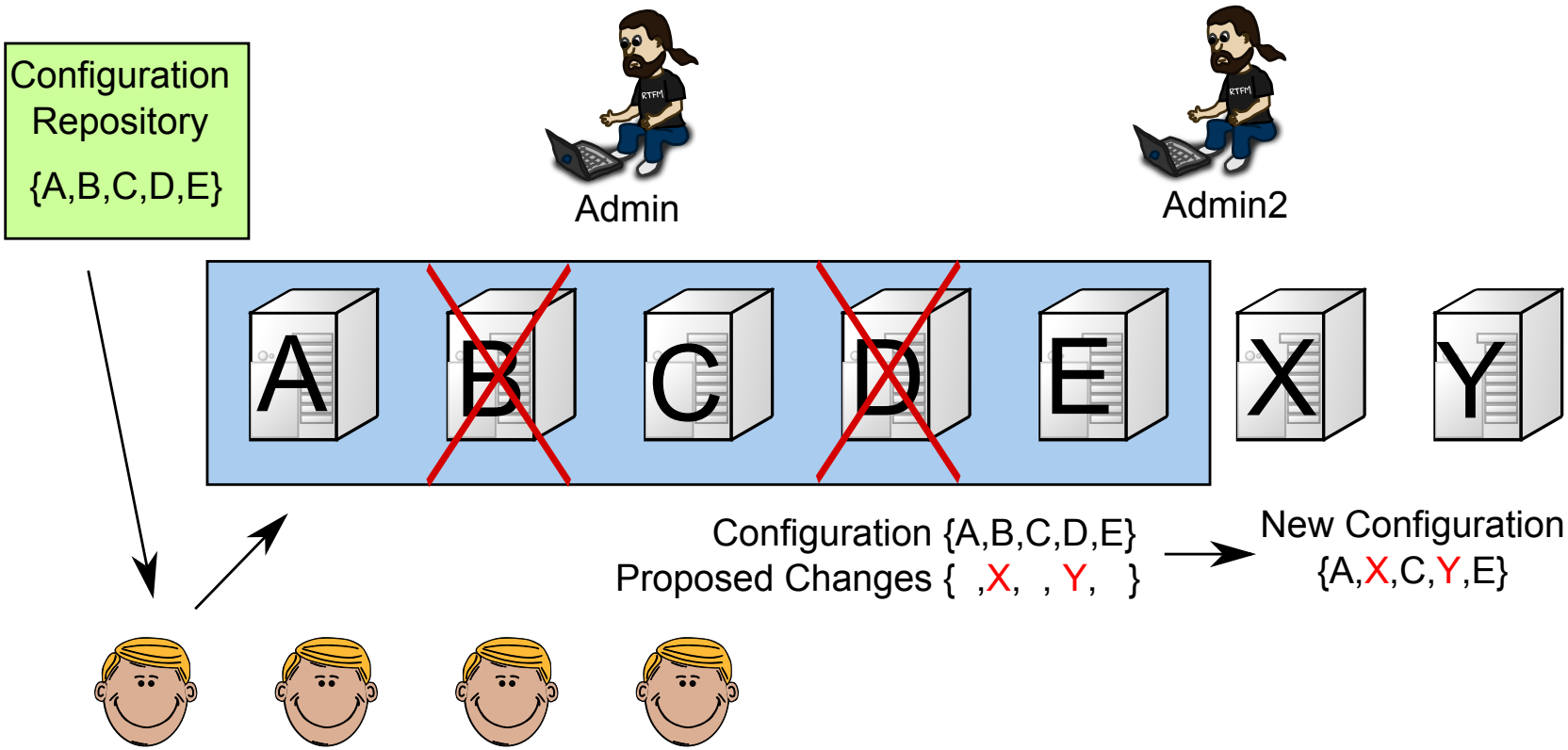
SMART with Multiple Admins



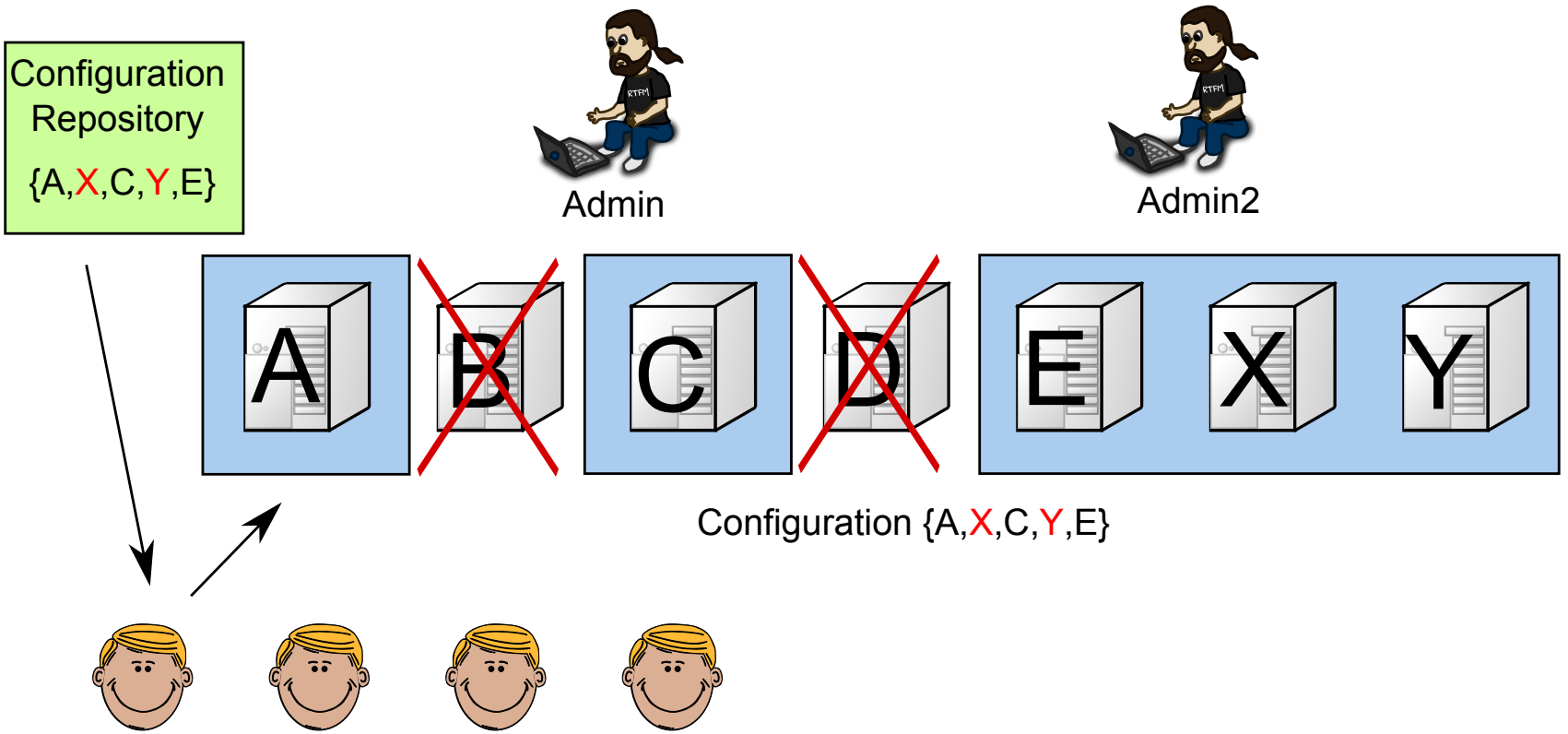
SMART with Multiple Admins



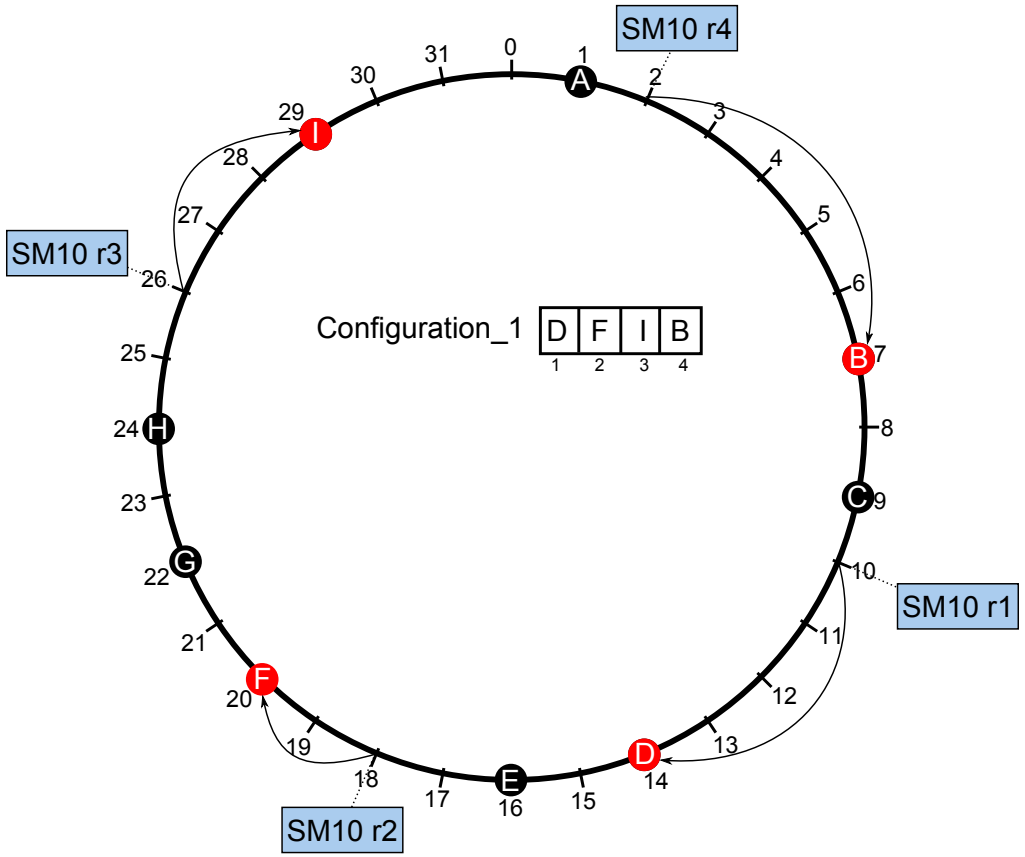
SMART with Multiple Admins



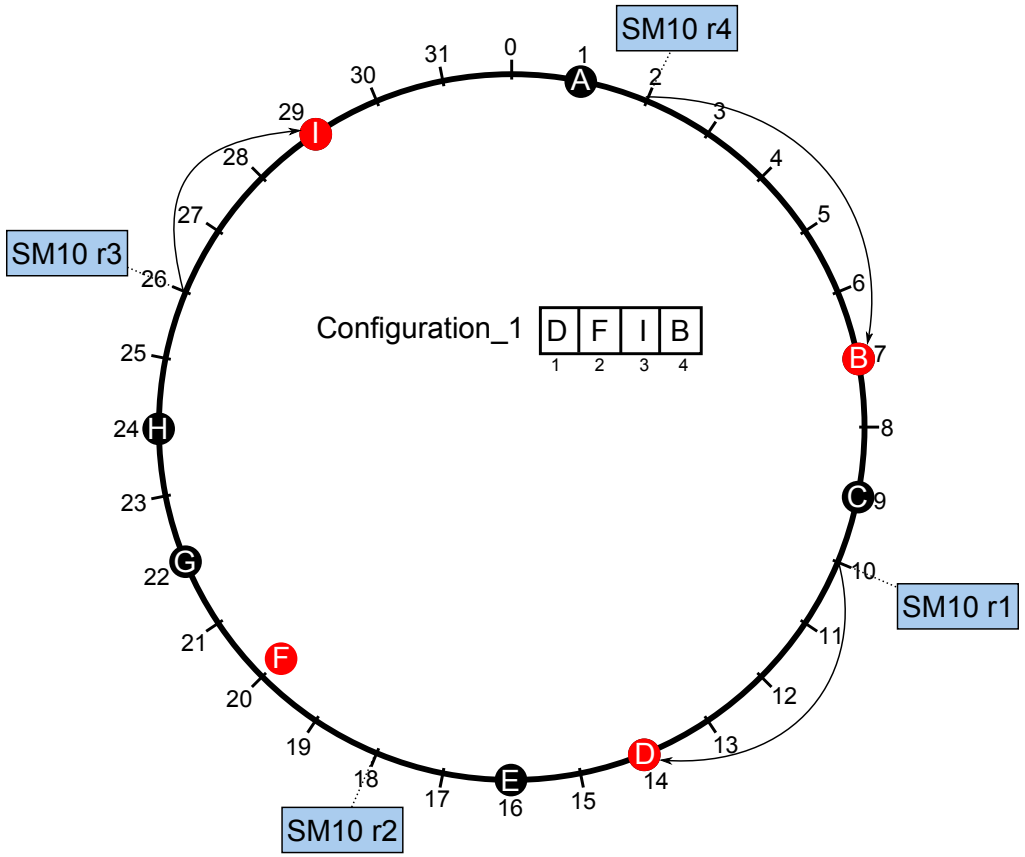
SMART with Multiple Admins



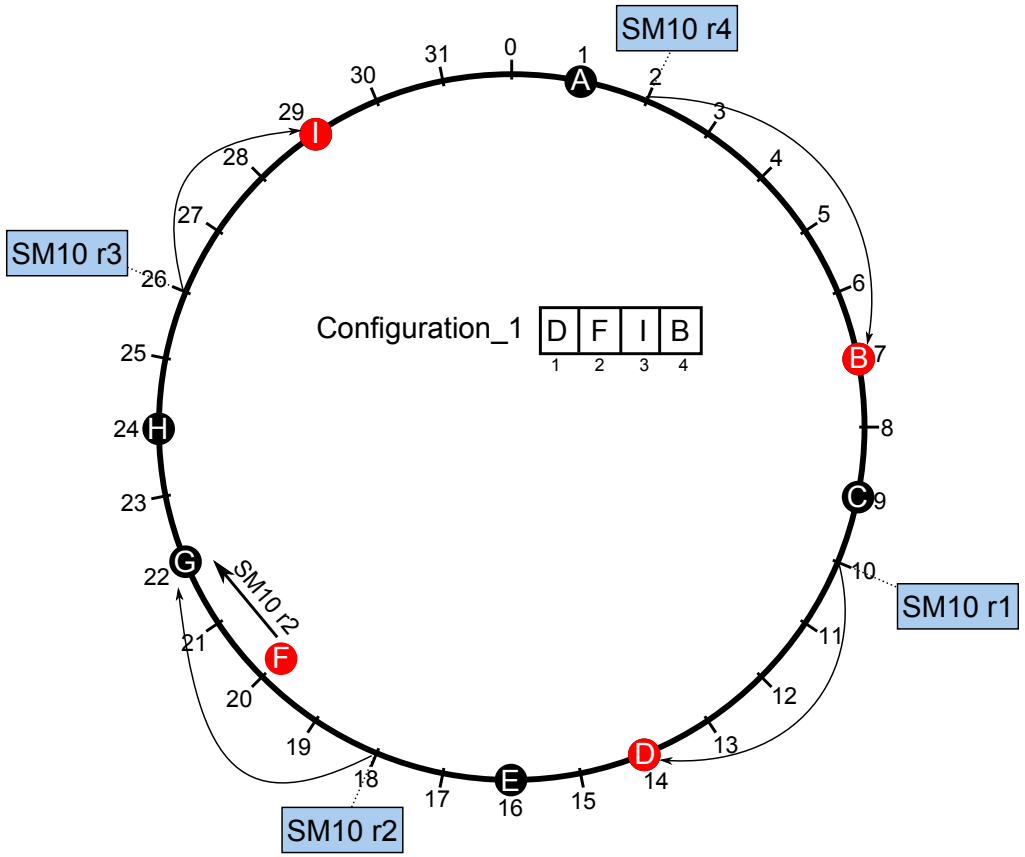
Handling Churn



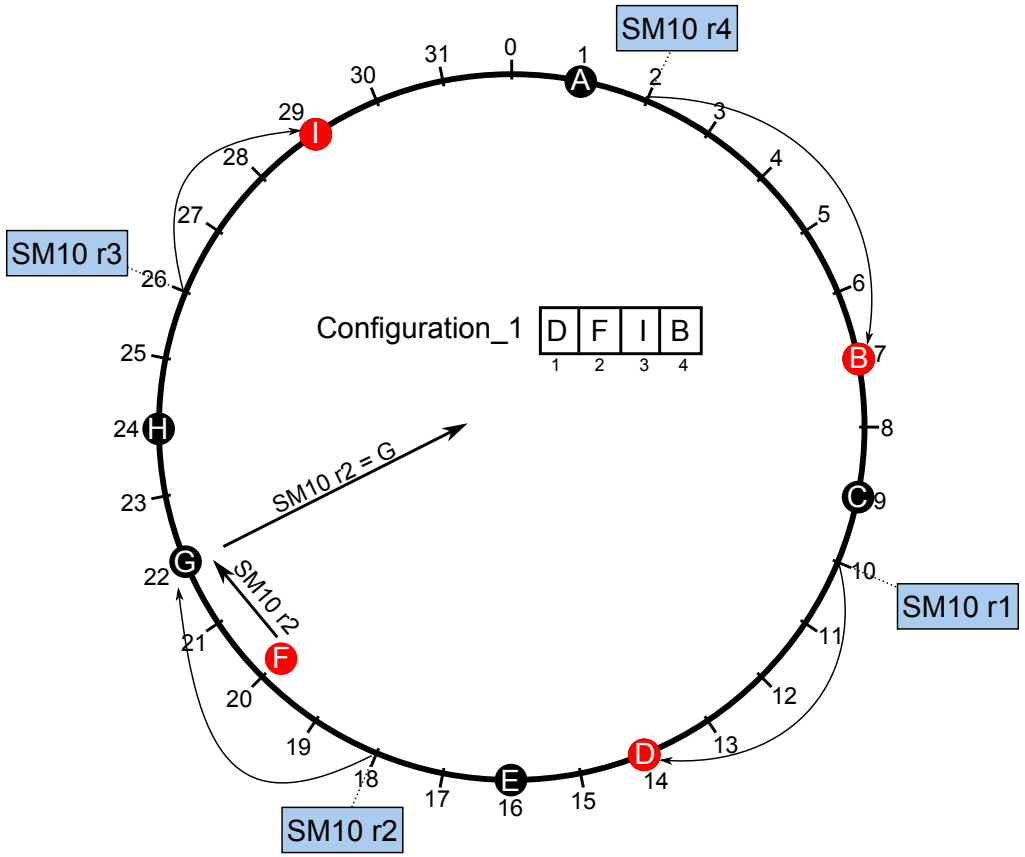
Handling Churn



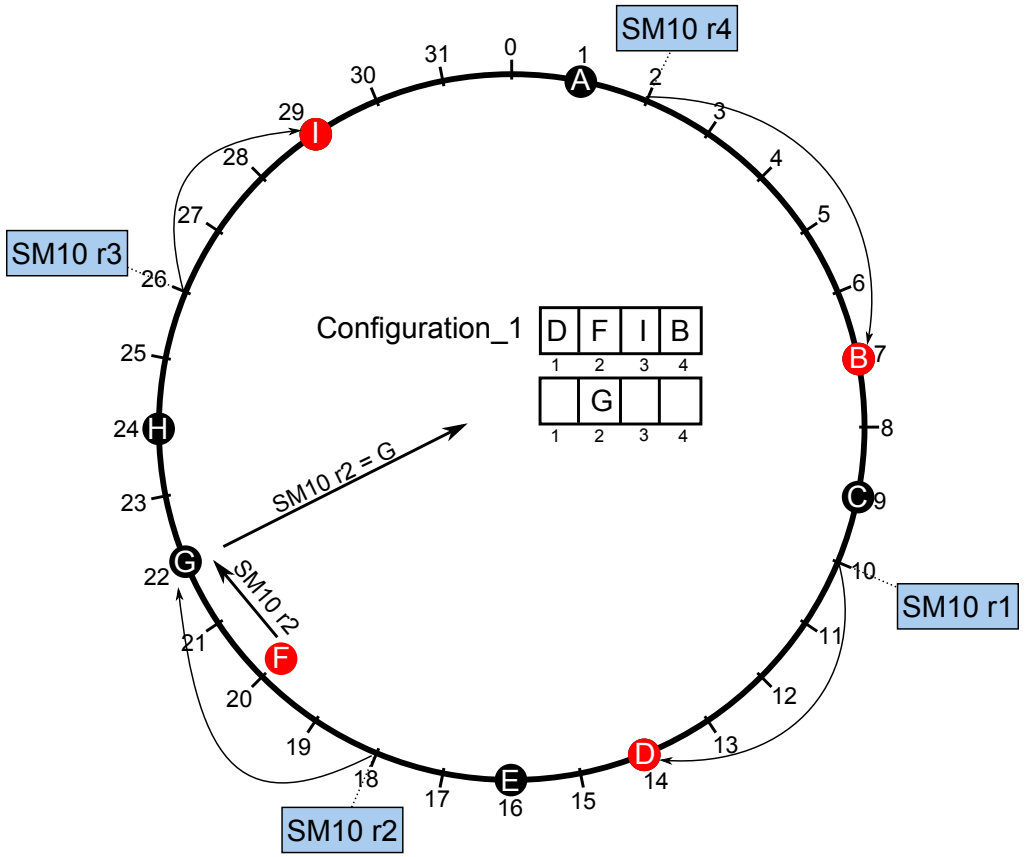
Handling Churn



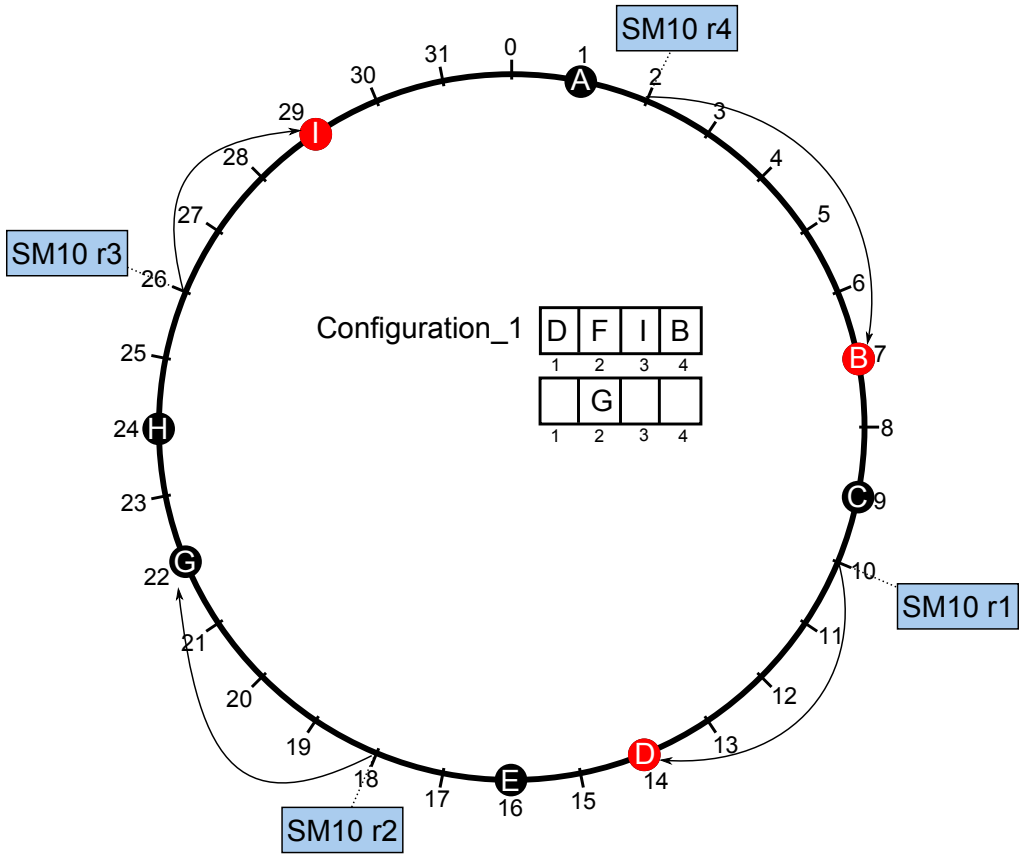
Handling Churn



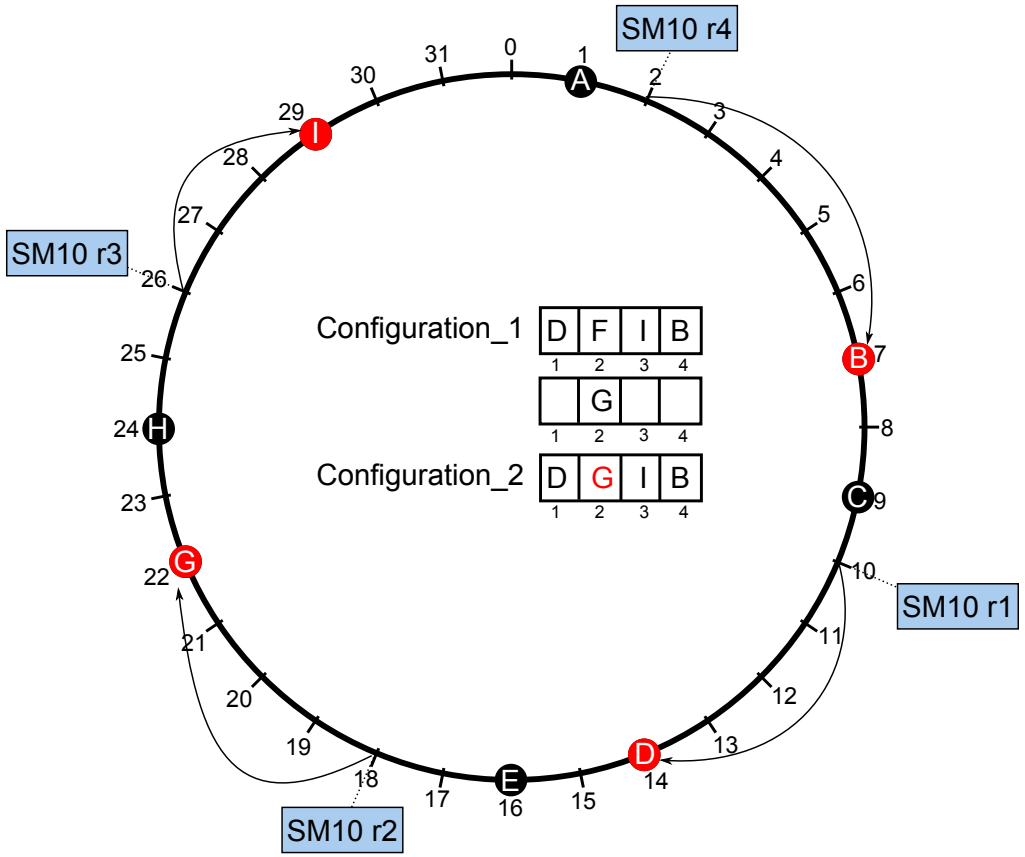
Handling Churn



Handling Churn



Handling Churn



Outline

Improve Management Logic

- Apply control theory to distributed systems
- Distributed optimization
- Reinforcement Learning

Self-Management in Cloud Applications

- Study elastic services in the Cloud
- Develop self-management techniques for Cloud applications
- Integrate all pieces into an elastic storage system