CAS7318

A Geo Redundant Cloud VoIP Service

Based on Geo Clustering for SUSE Linux Enterprise Server High Availability Extension

Brett Buckingham
Managing Director, *silhouette* Research and Development
Broadview Networks
bbuckingham@broadviewnet.com

Abstract

Broadview Networks' "OfficeSuite Phone" is a cloud-based VoIP service used by over 120,000 business subscribers daily. The primary product underlying OfficeSuite Phone, *silhouette*, is a carrier-grade telecom product. We have recently extended silhouette's existing high availability architecture to support geographic redundancy. This presentation is a case study of the use of Geo Clustering for SUSE Linux Enterprise High Availability Extension. We will outline the challenges and solutions to several aspects of geo redundancy, including database replication, filesystem replication, geo cluster overlay, and the design of a dead man's switch to control geo failover.

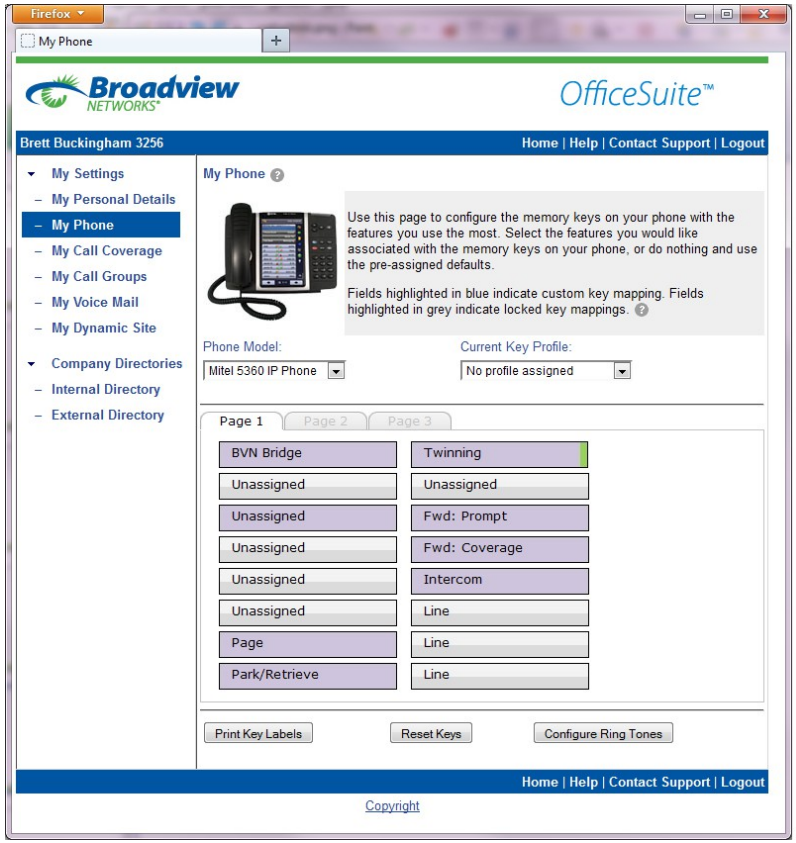
Table of Contents

Product Overview

Product Overview

- *silhouette* is sold to cloud services providers (CSPs)
- CSPs use *silhouette* to provide phone service to small to medium businesses
- 1 *silhouette* supports 20,000 subscribers (e.g. equivalent of 1000 PBXs with 20 subscribers each)
- Cloud VoIP: only phones and IP network at customer site
- Businesses manage their phone service entirely via a web interface
- Broadview Networks hosts the OfficeSuite service based on *silhouette (Broadview is a CSP),* and also licenses *silhouette* to other CSPs

Phone System Managed via Web



silhouette is Widely Deployed

- Broadview Networks has 14 *silhouette* systems in production underpinning the OfficeSuite Phone service, serving over 120,000 business users every day
- Broadview licenses *silhouette* to 17 other CSPs world wide, which combined serve an additional 60,000 business users every day

silhouette is Carrier-Grade

As a product intended to be hosted by cloud and telecom service providers, silhouette is subject to carrier-grade requirements, such as:

- Availability: 99.999%
- Reliability: 99.99%
- Scalability and throughput
- Real-time responsiveness
- Manageability and serviceability
- Security

Other Product Information

- Developed over past 13 years; in live production service for 10 years
- Software only

Comprised of several (25+) software components

Deployed on carrier-class X64 servers

SLES+HAE+Geo is embedded in the product

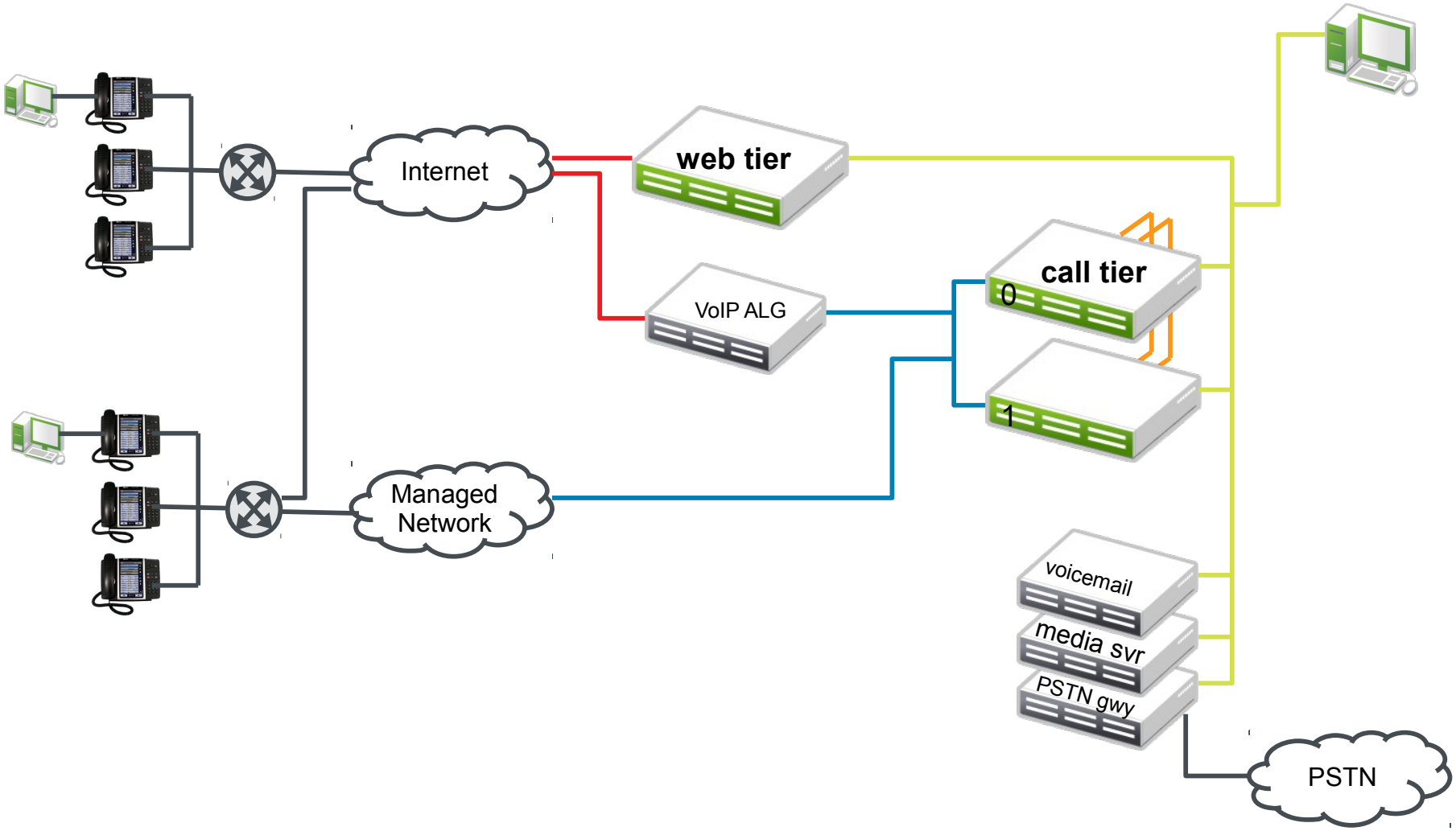
Interfaces with network peer components for some functions

- Deployed on 3 servers over 2 tiers:

Web-tier: single node HA “cluster”

Call-tier: 2 node HA cluster

Network Diagram



Rationale for Geo Redundancy

Why Geo Redundancy?

- Customers expect it / require it
- Business continuity safeguard

Expectations of a Cloud Service

- Using a cloud service means trusting a CSP to provide and manage the service. There can be an emotional barrier to trust a 3rd party vs. control the service in-house
- If the service is business critical (e.g. phone, email), the emotional barrier can be amplified
- Cloud services are presumed to be relocatable, distributed, resilient, not tied down to hardware or location; this can help to offset the angst
- Geo redundancy is at least an implicit expectation, and often is an explicit RFP check box, especially by customers who have recently experienced a disaster

Business Continuity / Disaster Recovery

- Business continuity refers to plans, policy, preparation, and procedure to safeguard a business and continue its operations despite serious incidents or disasters
- Some disasters are related to geography, e.g. flood zones, earthquake fault lines, common public utilities, etc.
- A geo redundant system intends to safeguard the system against geographic disasters, therefore redundant systems should be geographically diverse

Our Experience with Hurricane Sandy

- Destructive and deadly Atlantic hurricane in October 2012 which affected Caribbean and east cost of North America
- Well prepared (100% uptime), but learned a lot
- In one of Broadview's telecom central offices in NYC:

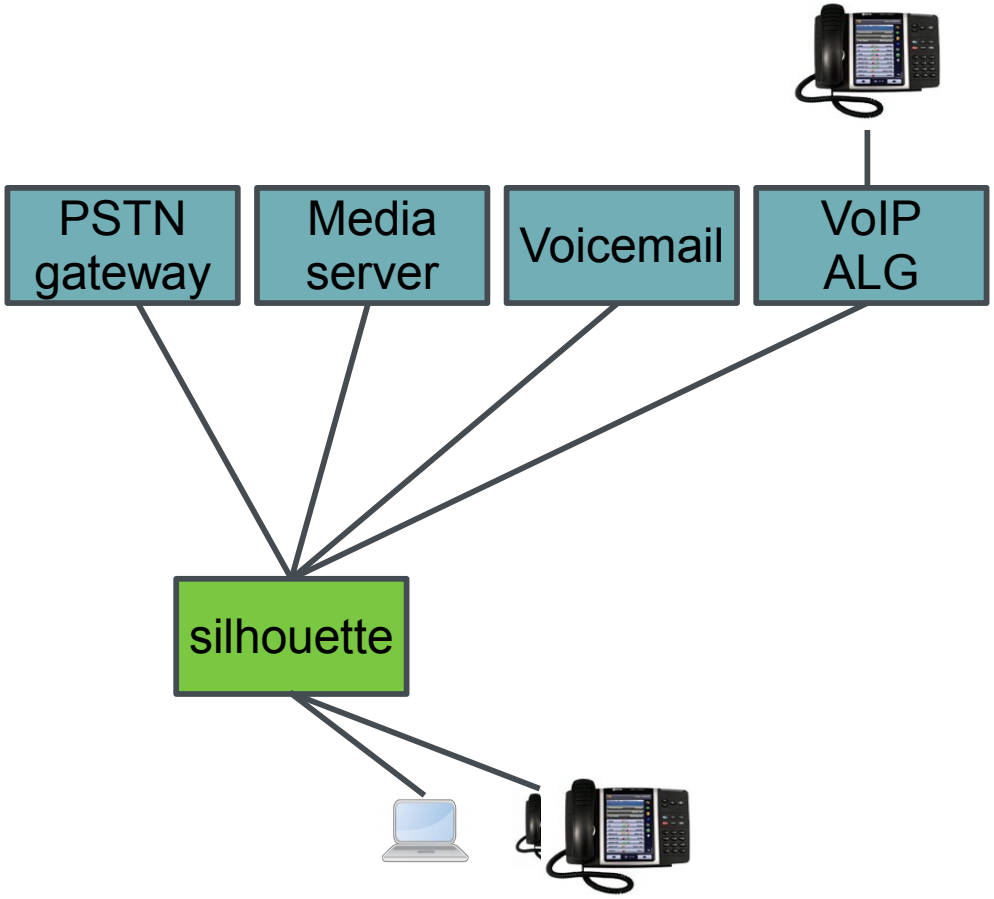
Commercial power down for 2 weeks, then unreliable for 2 additional months. We were on generator power throughout.

Basement and lobby flooded, travel in/out of Manhattan impossible. Operations personnel on-site continuously for several days

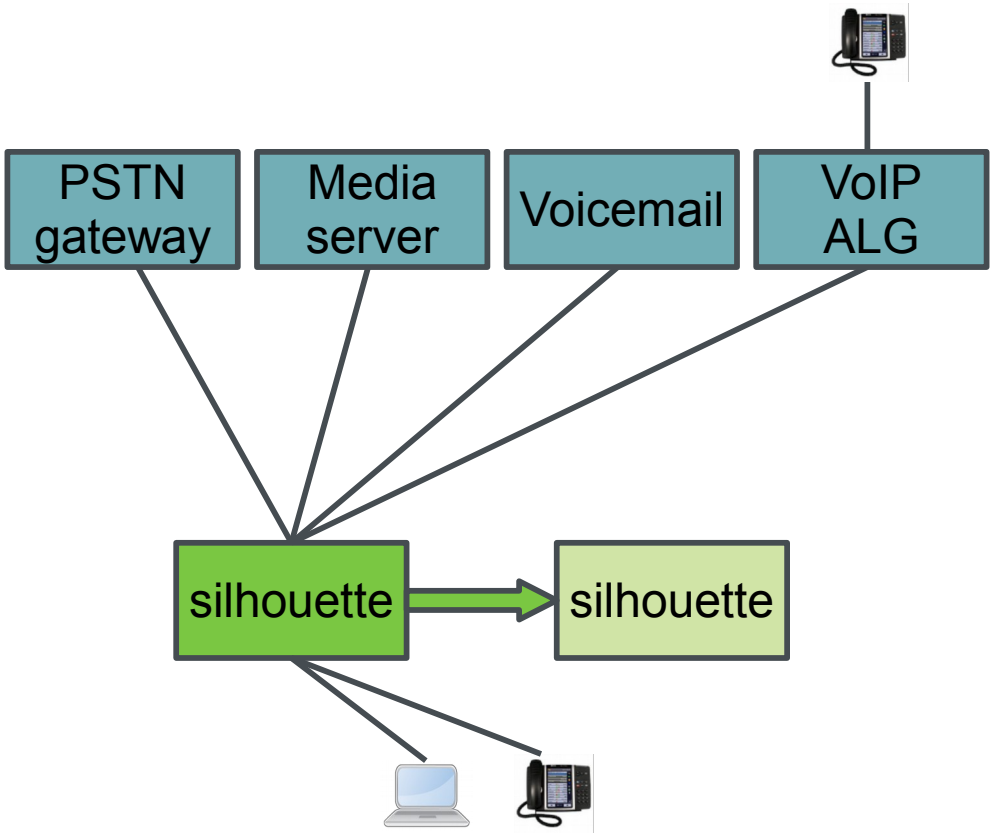
Some circuits from peering partners were down, and ultimately some partner COs were unrecoverable due to salt water damage

Geo Redundancy Overview

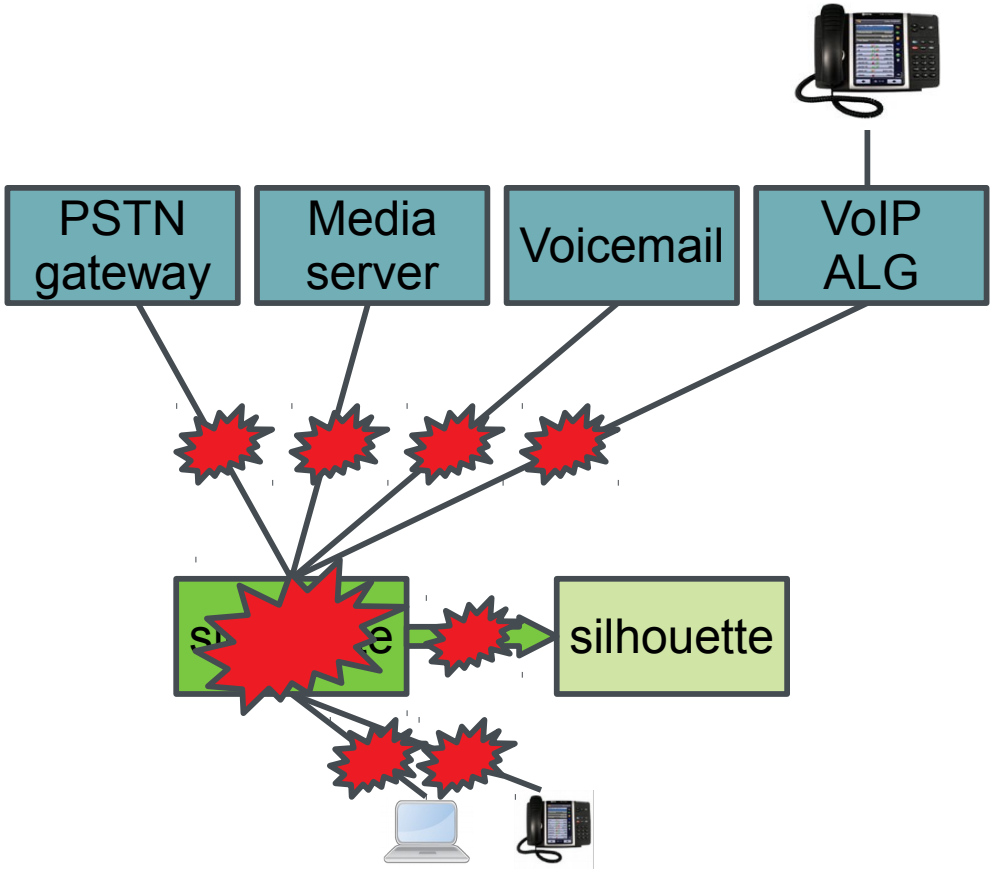
System and Network Peers



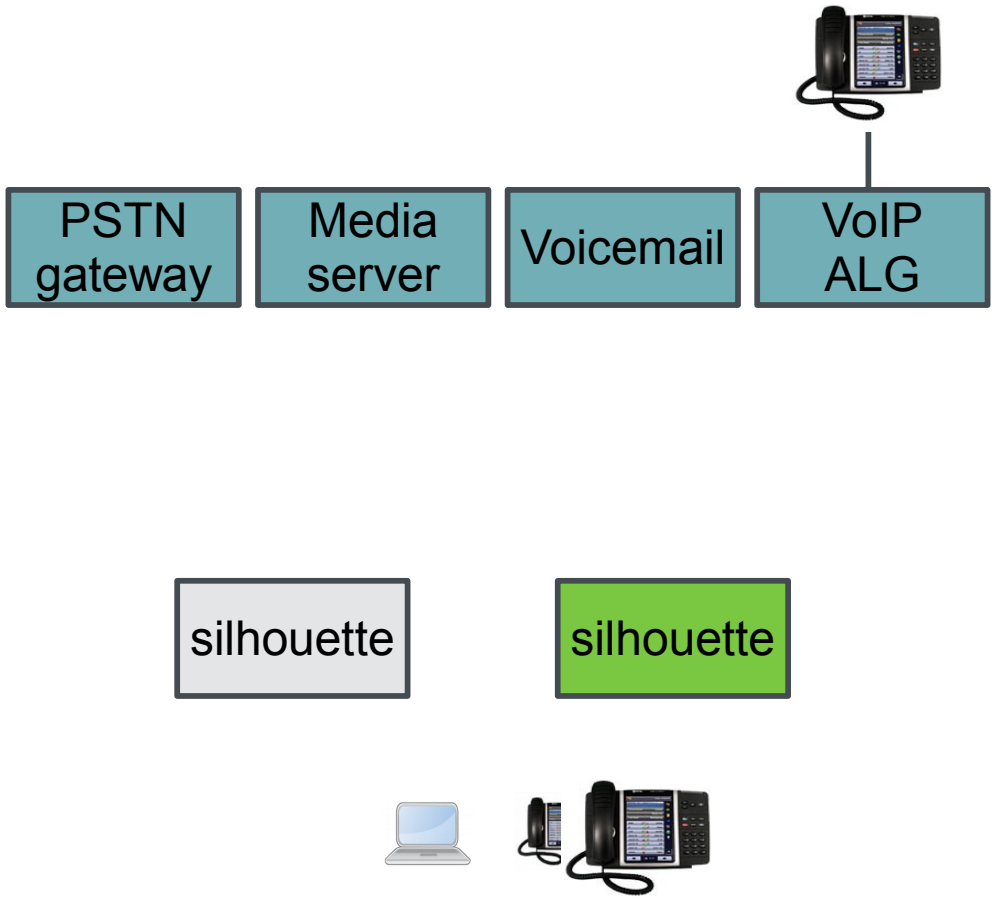
Replication to Backup System



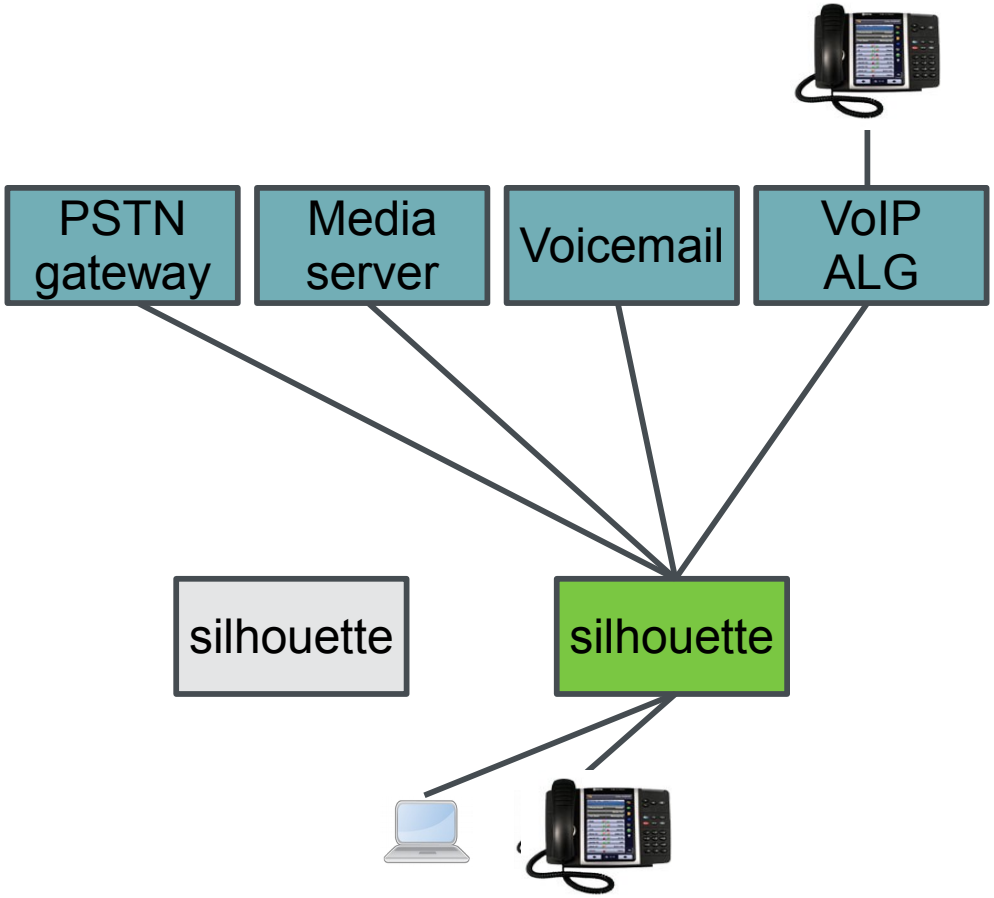
Primary System Failure Detected



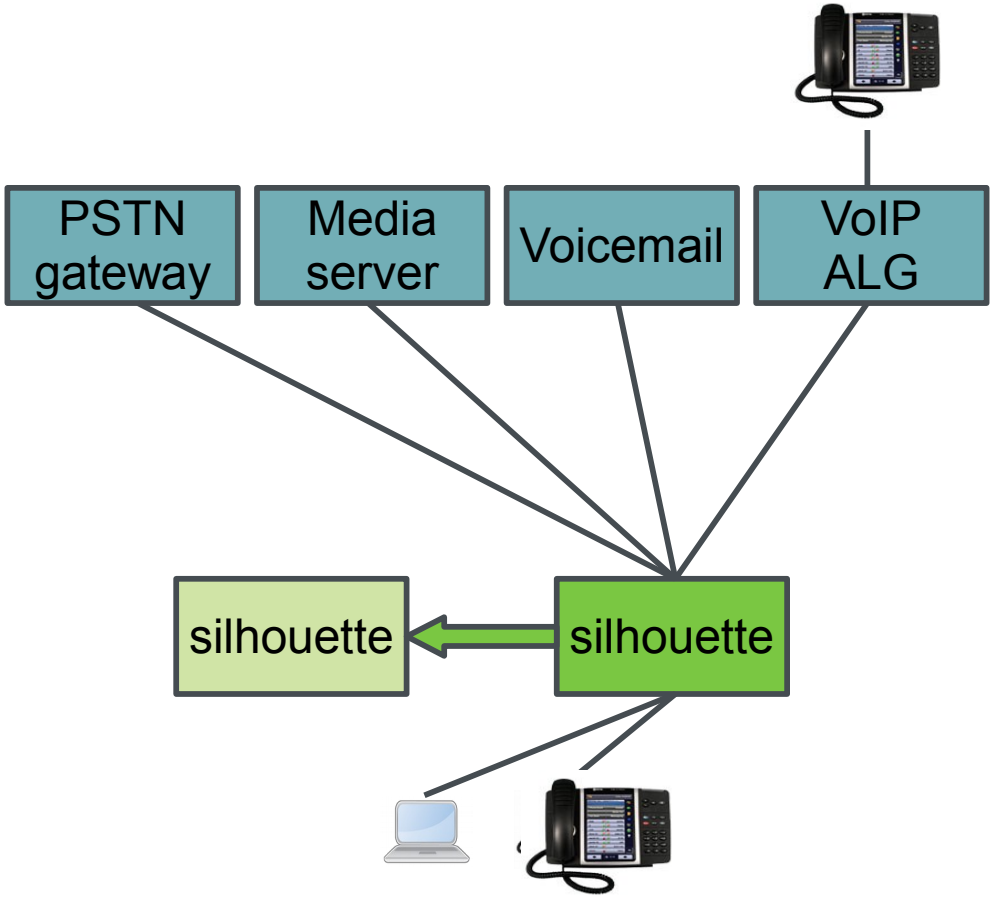
Backup System Promoted



Network Peers Connect to New Primary



Recovered System Becomes New Backup



Basic Concepts

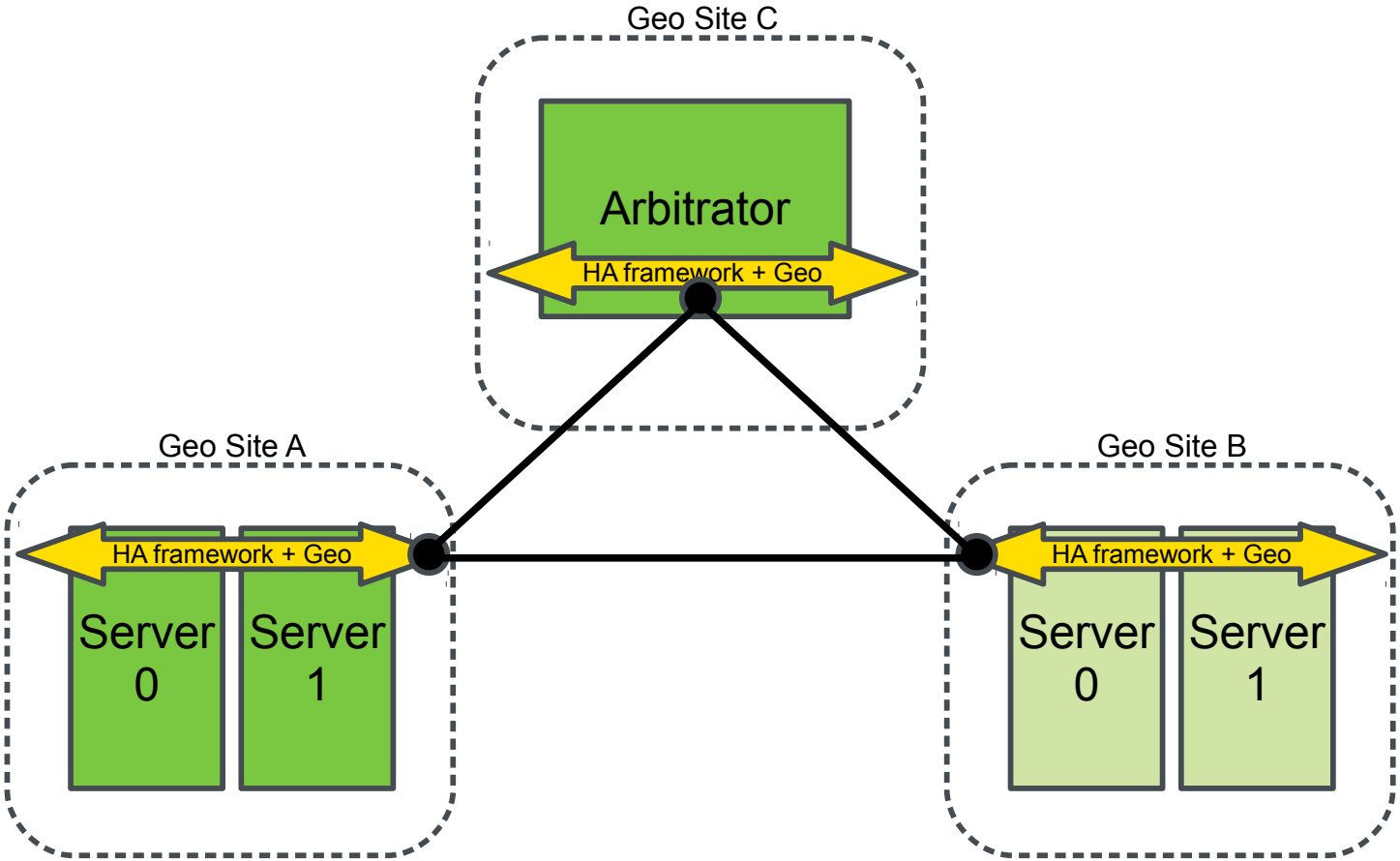
- Continuously replicate *silhouette* configuration and operational data to a backup geo site
- Only the primary geo site (typically) provides service at any one time
- Detect failure of primary site, promote backup to primary
- Implement mechanisms for network peers and phones (i.e. client systems) to recognize and tolerate *silhouette* changing location (IP)

Geo Cluster Architecture

Geo Cluster

- Employs Geo Clustering for SUSE Linux Enterprise High Availability Extension
- *Silhouette* primary and backup geo sites are linked in a geo cluster to an arbitrator node in a 3rd geo site
- Only call tier nodes participate in the geo cluster; web tier nodes are subordinate to and controlled by the call tier
- Typically only one *silhouette* provides service at any one time, as directed by a ticket scheme in the geo cluster

Geo Cluster


Geo Site C
Arbitrator
HA framework + Geo
Geo Site A
HA framework + Geo
Server 0
Server 1
Geo Site B
HA framework + Geo
Server 0
Server 1

Database Replication

High Availability Databases

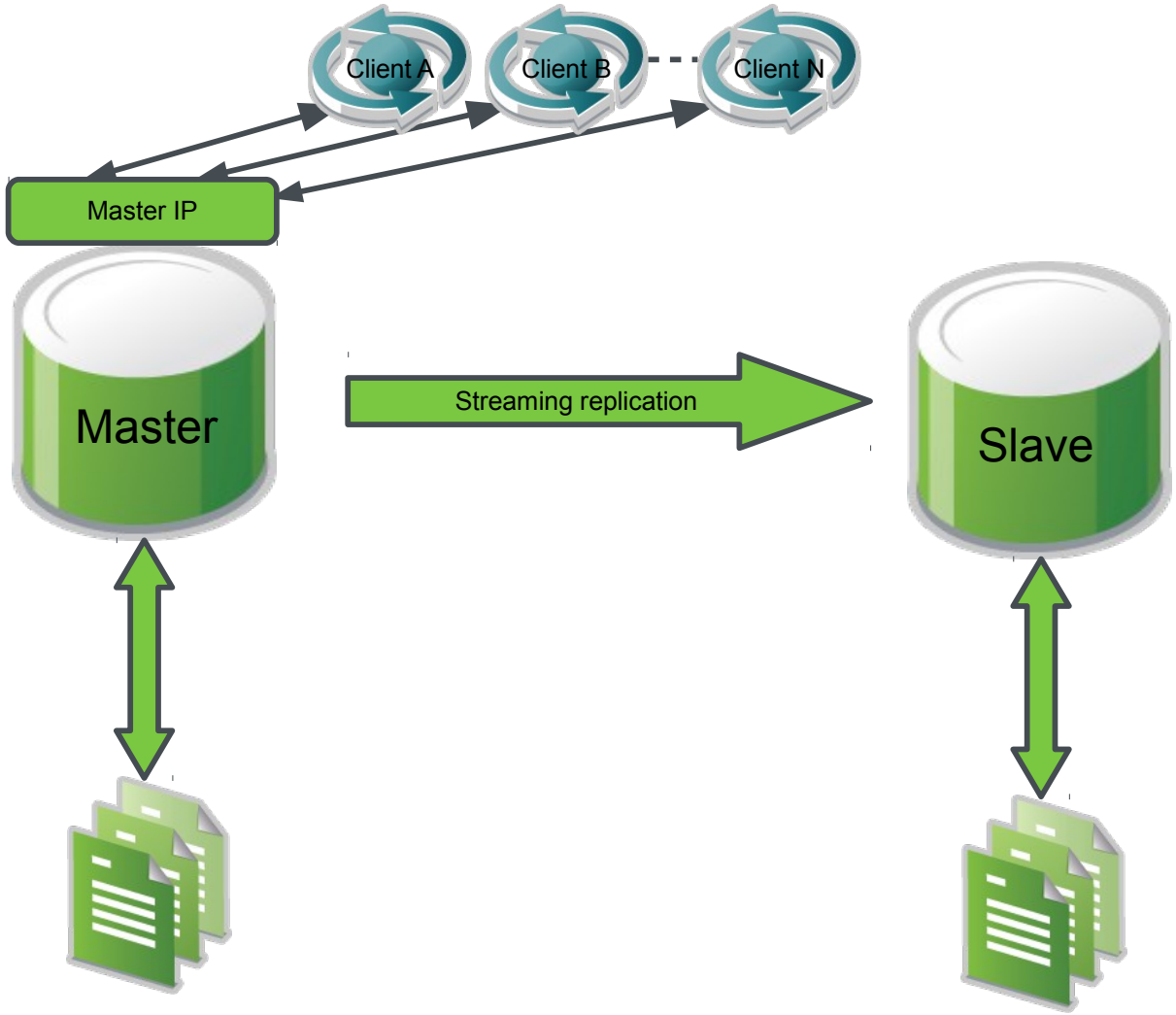
- *silhouette* contains 2 databases:

Main database: provisioned system and business data

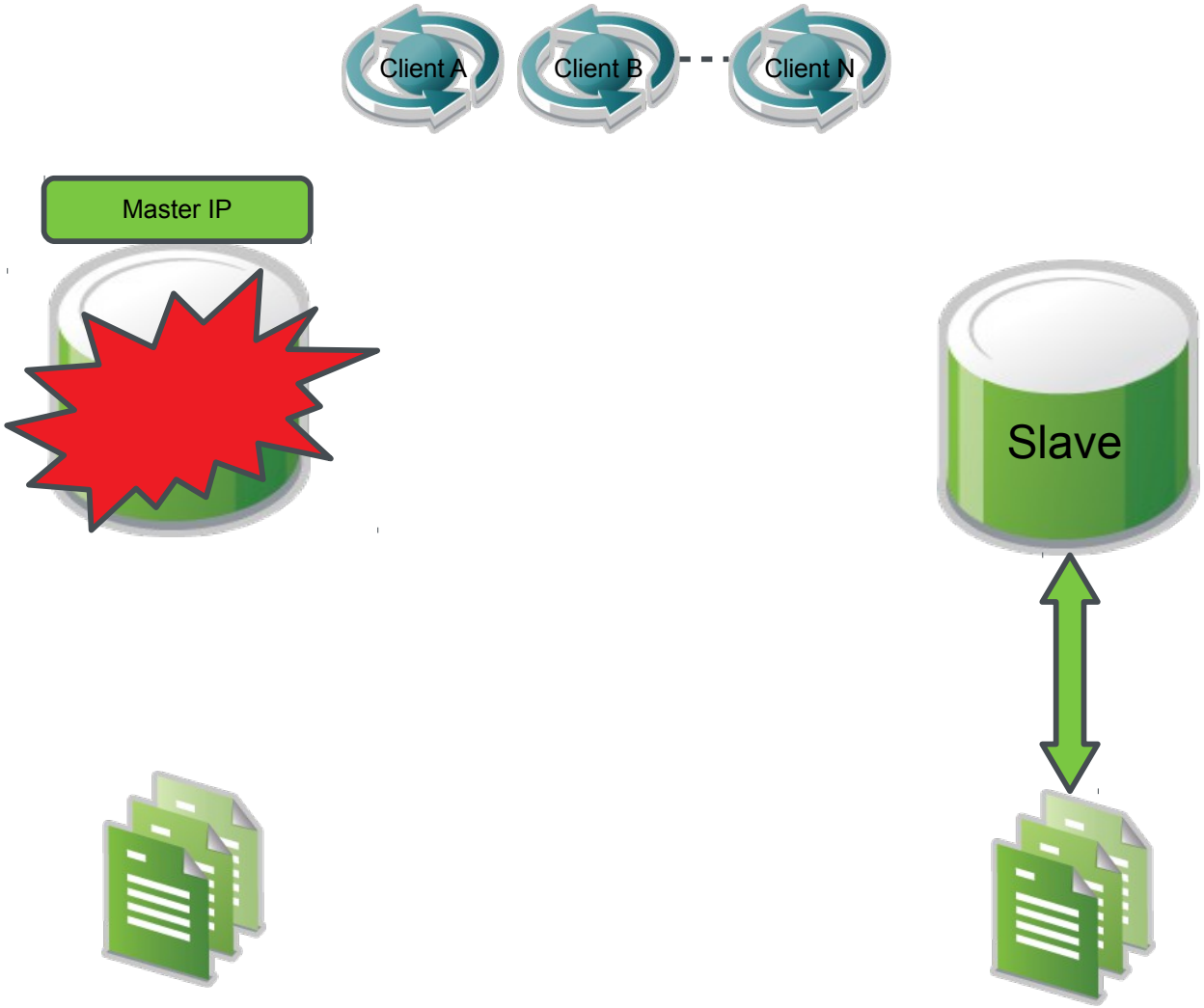
Billing database: call records

- Both databases are implemented with PostgreSQL
- Each database has HA requirements, and employs PostgreSQL streaming replication to a warm standby slave within local cluster
- Stock PostgreSQL resource agent (RA) was inadequate for this master/slave arrangement
- We developed a custom design and RA

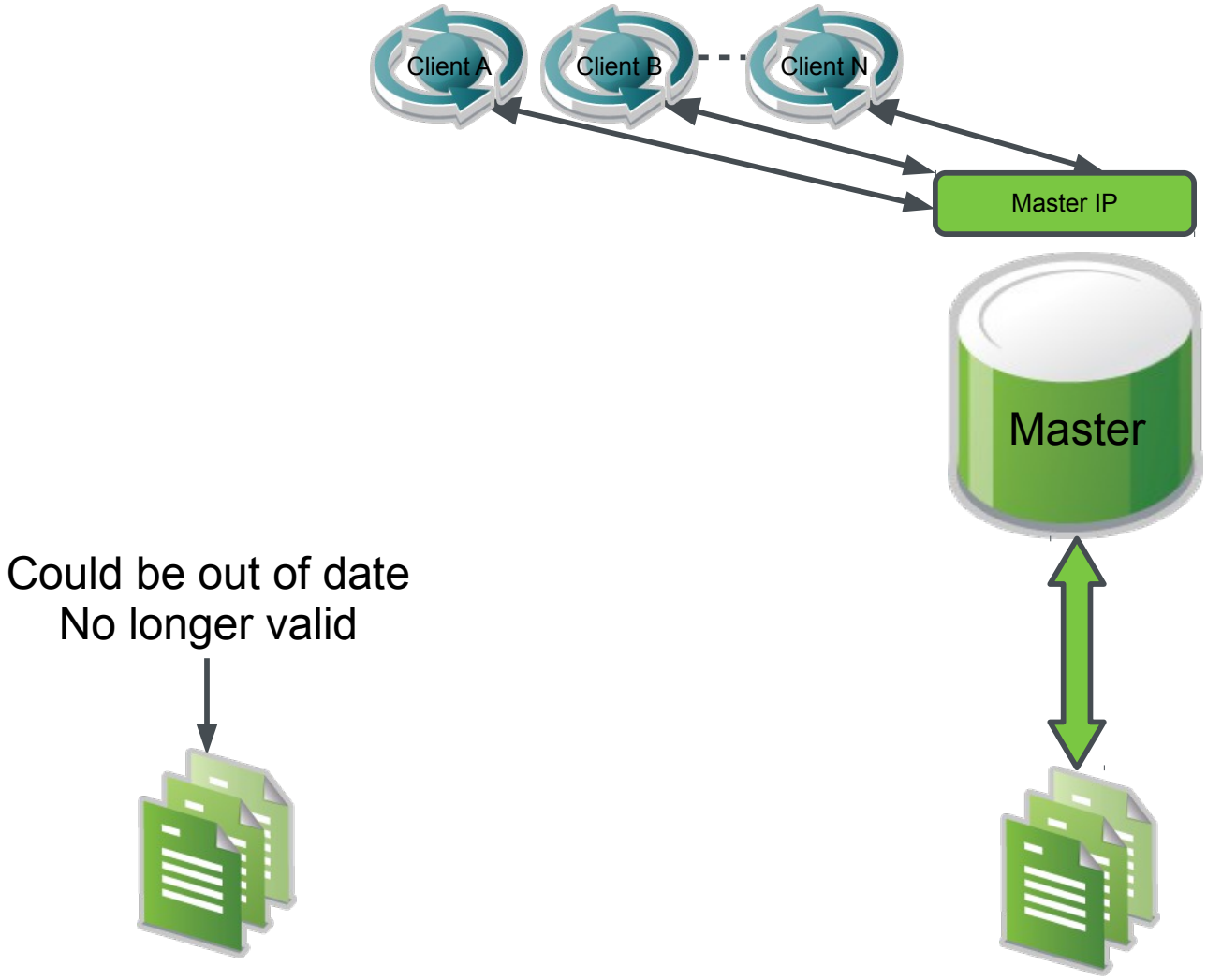
Warm Standby Slave with Streaming Replication



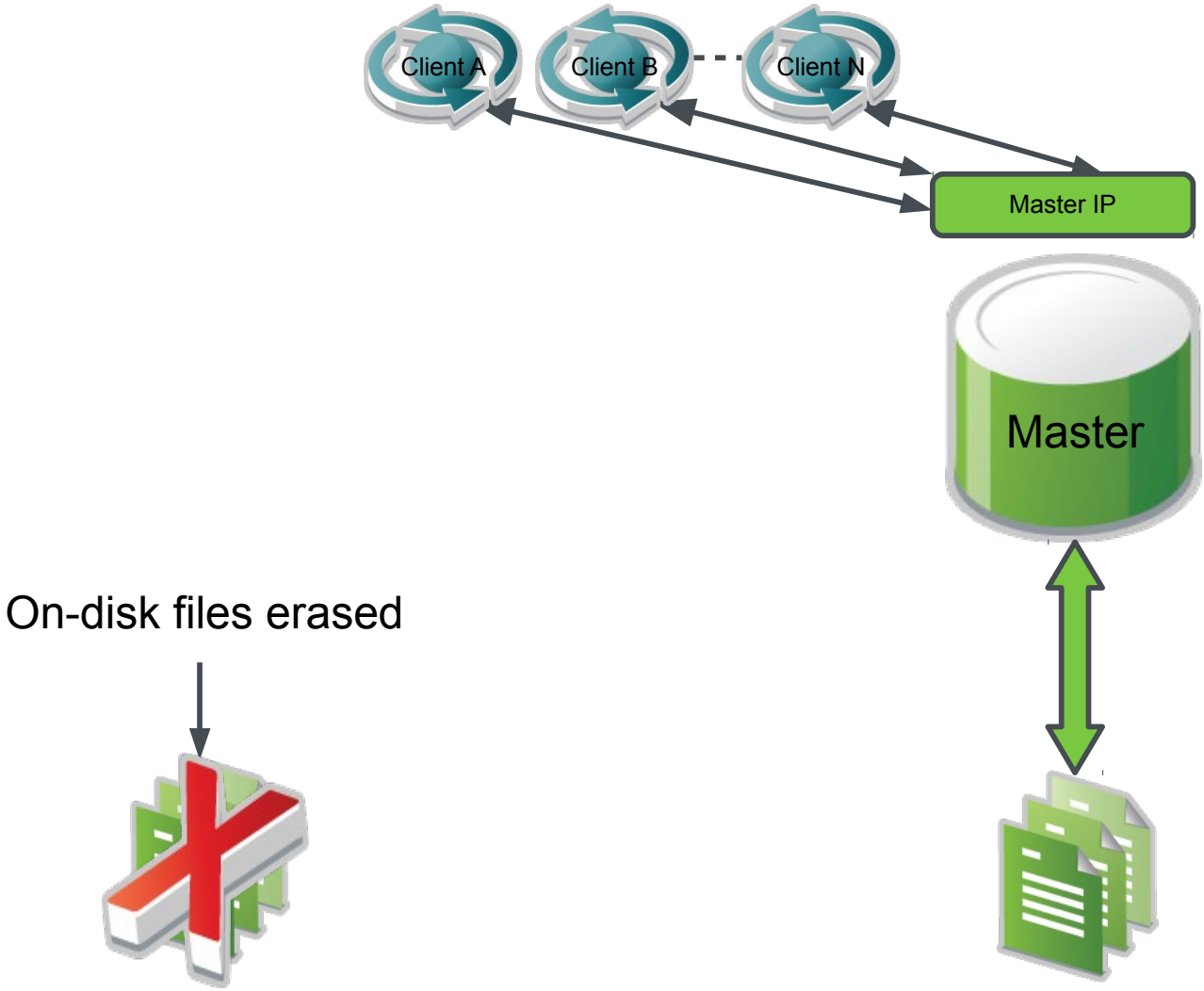
Master Fails



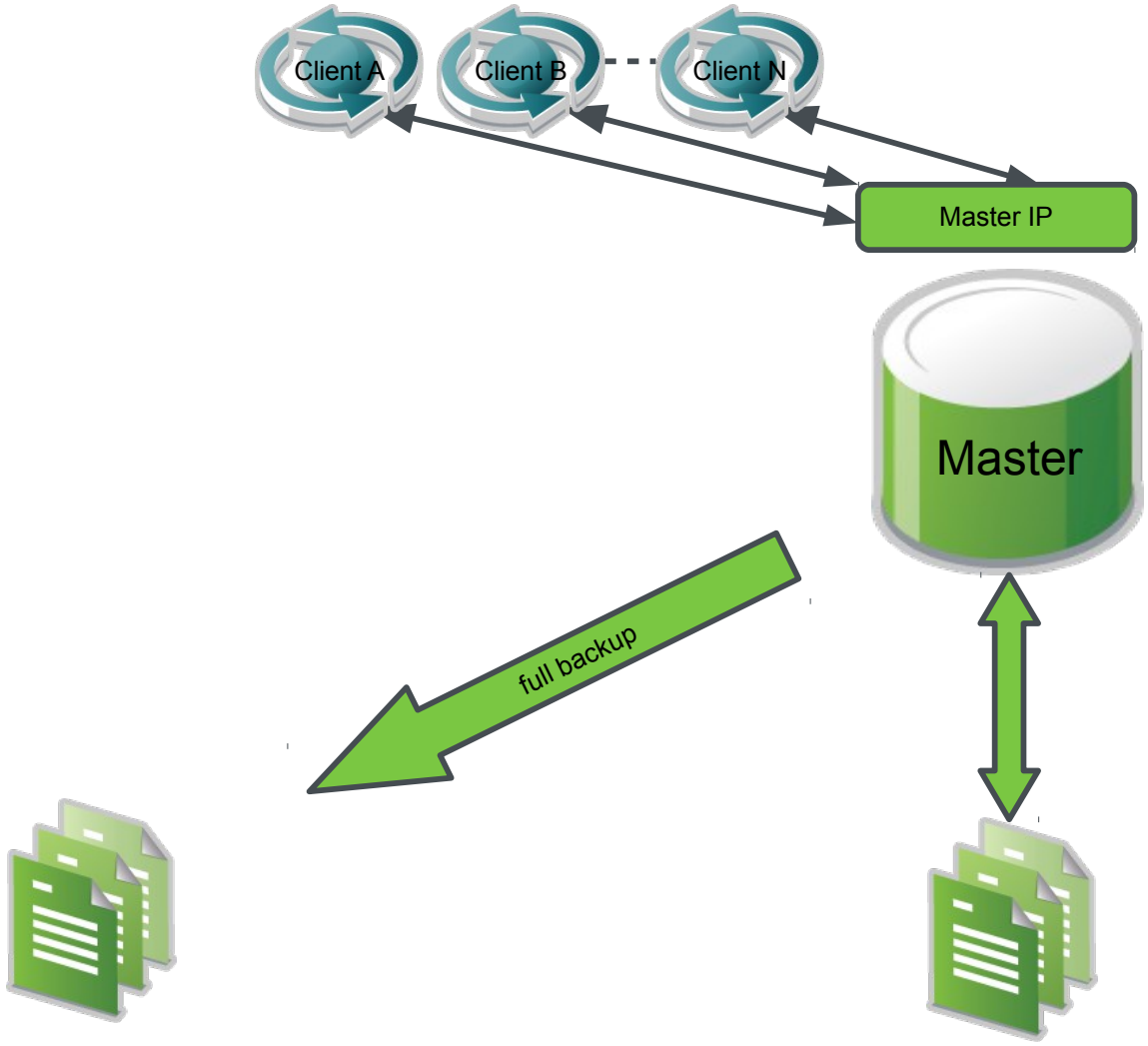
Slave is Promoted, IP Follows, Clients Reconnect



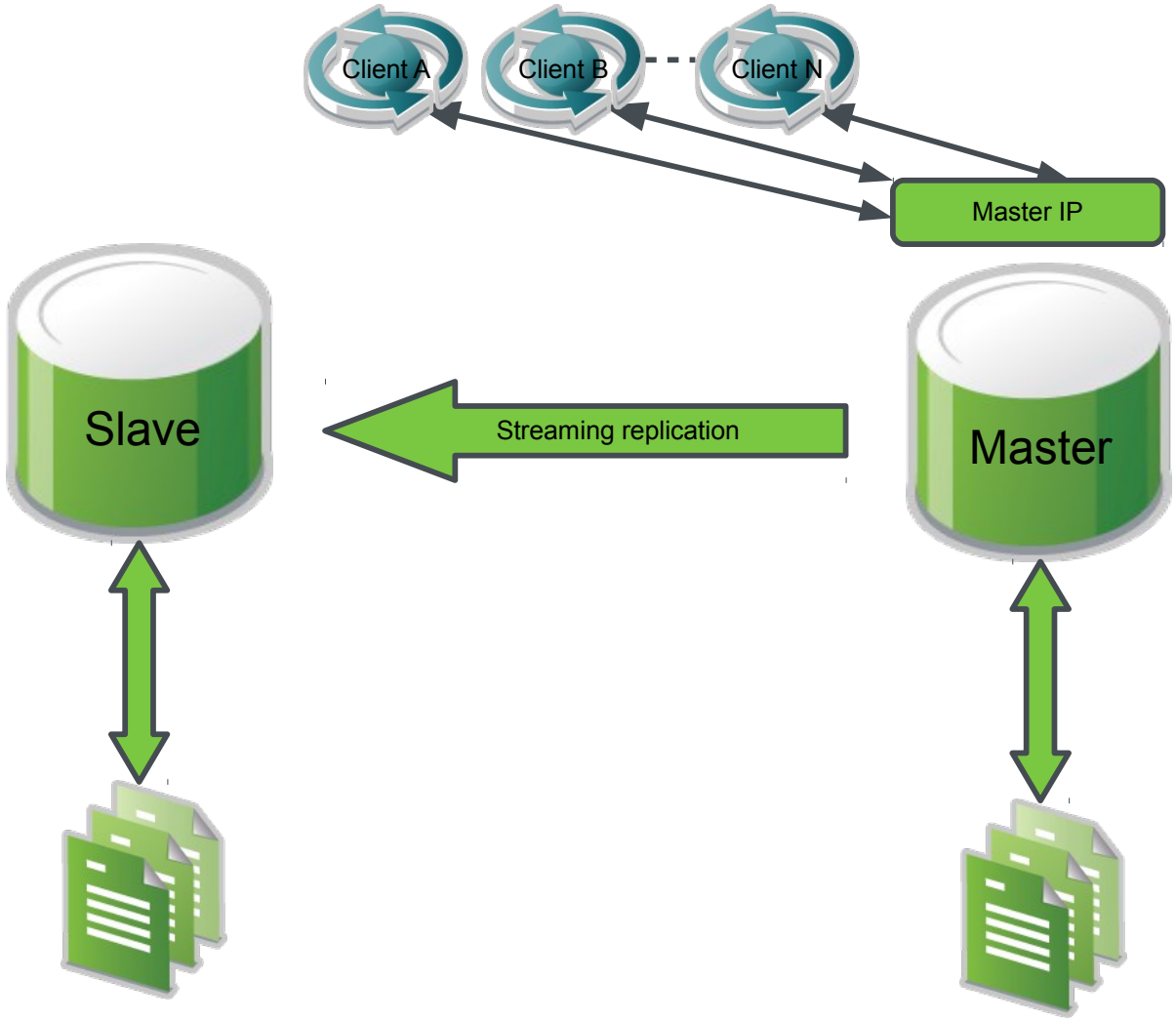
Failed Instance Restarts as Slave



Failed Instance Restarts as Slave



Failed Instance Restarts as Slave



Initial Startup

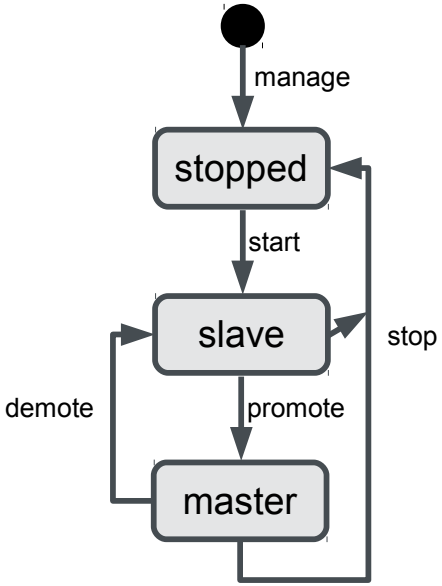
- In Pacemaker the startup sequence for a master / slave resource is as follows:

Pacemaker starts resource as a slave on node A

Pacemaker starts resource as a slave on node B

Pacemaker chooses one instance to promote to master

Master / Slave Resource State Machine



Initial Startup Problem

- Problem: for our design, “starting as a slave” means to prepare a slave database as follows:
  1. Erase files on disk (both instances would do this, and wipe out all data!)
  2. Obtain a full backup from master instance (this would fail - there isn't one yet)
  3. Start slave database instance in PostgreSQL “hot standby recovery-mode” with streaming replication

Initial Startup Problem: Solution

- Custom PostgreSQL resource agent
- When told to start as a slave:

If there is a running master, prepare and start slave as normal

If there is no running master:

Do nothing

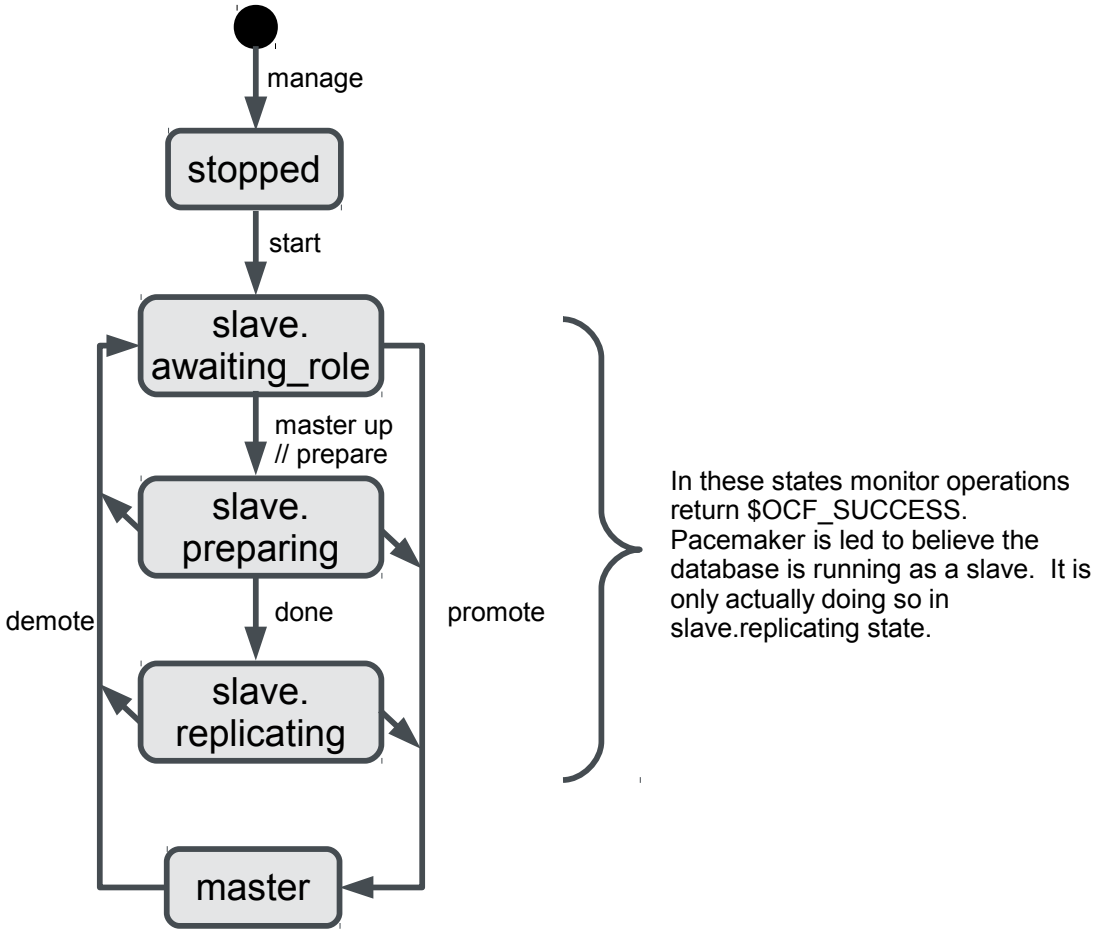
Return $OCF_SUCCESS to Pacemaker as if successfully started as a slave

When Pacemaker eventually promotes one instance:

Start that instance as a master from disk image

Prepare and start the other instance as a slave

Modified Master / Slave Resource State Machine



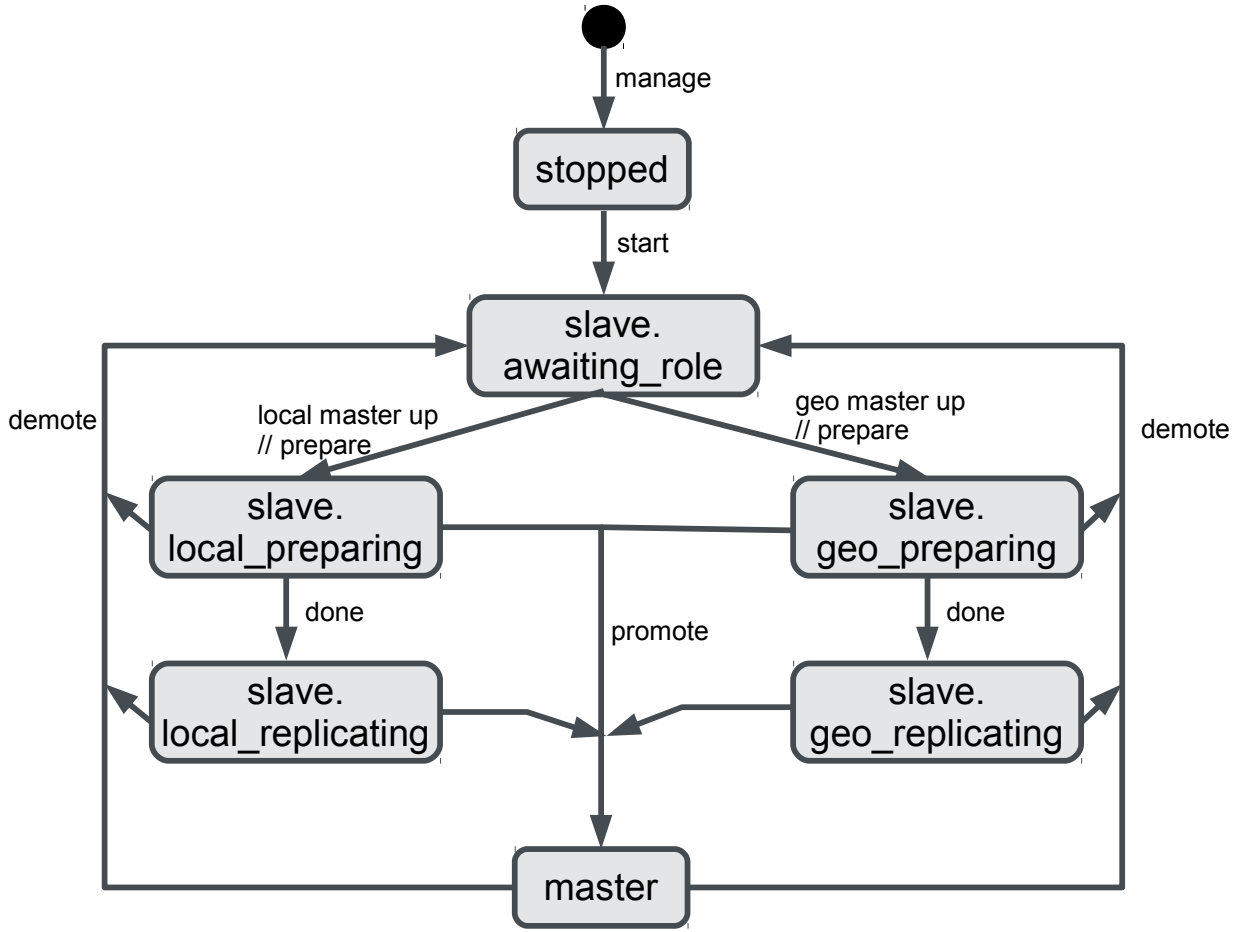
Additional Enhancements

- “Fallback” images: rotation of regular database backups are taken on each node and stored locally. If normal HA mechanisms fail, or the database is corrupted, the RA will start the database from a fallback image
- Enhanced monitoring: for our purposes, it is not sufficient to deem a database to be alive based on there being a running PID. Our RA performs representative database queries.

Geo Redundancy Extensions

- Database replication from primary geo site to backup geo site is the same PostgreSQL streaming replication
- Additional modifications to the state machine in the RA accommodate “local slaves” and “geo slaves”
- Significant additional complication when working across geo sites, as HA events (notifies, etc.) do not traverse the geo cluster
- The design pattern we used extensively is to perform various event transition checks during regular monitor operations

Geo Redundant Master / Slave Resource State Machine



Filesystem Replication

Filesystem Replication

- Some *silhouette* components are made highly available by storing their state on a filesystem shared between the nodes of the local cluster (e.g. DHCP server's conf and lease files)
- Select portions of this shared filesystem needed to be replicated to the backup geo site
- We explored various options, such as cluster filesystems (GFS, OCFS), DRBD layers, csync2, etc.
- For our application and constraints, these technologies were not appropriate
- We implemented a simple pull paradigm replicator component based on rsync

Dead Man's Switch

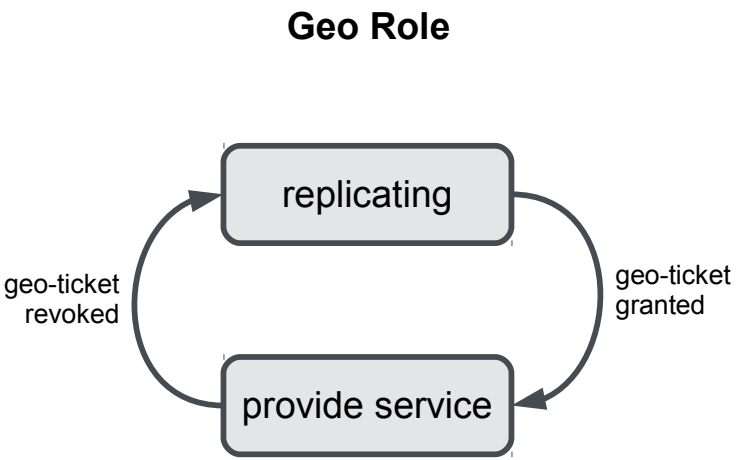
Dead Man's Switch

- *silhouette's* geo redundancy architecture includes a dead man's switch implemented by a custom component called the geo-manager
- the dead man's switch decouples geo cluster decisions from geo site service decisions
- basic idea: if the geo cluster decides that a geo failover should happen, a dead man's switch timer (default: 2 hours) is started
- the geo failover can be manually confirmed or aborted before the timer expires
- if the timer expires, geo failover happens automatically
- allows geo failover intervention by operations persons

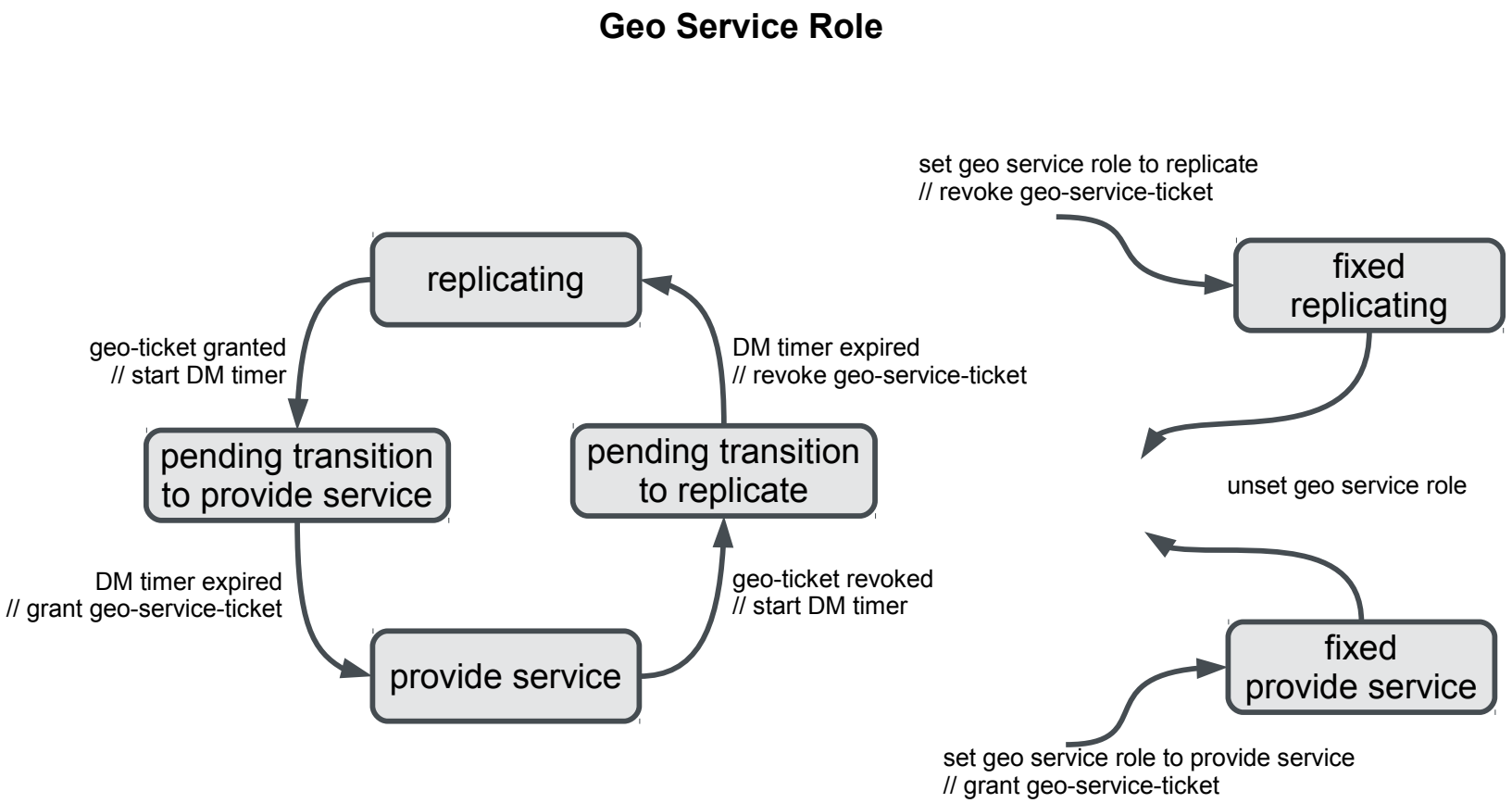
Geo Role vs Geo Service Role

- Geo role: the role that the geo cluster wants a geo site to be. If geo-ticket is granted, the geo role is “provide service”. If revoked, the geo role is “replicate”
- Geo service role: the actual service role (“provide service” or “replicate”) that a geo site takes on at a given moment.
- The geo role as embodied by the geo-ticket is a suggestion to the geo-manager. The geo-manager controls the actual geo service role by a local cluster ticket called geo-service-ticket

Geo Role State Machine



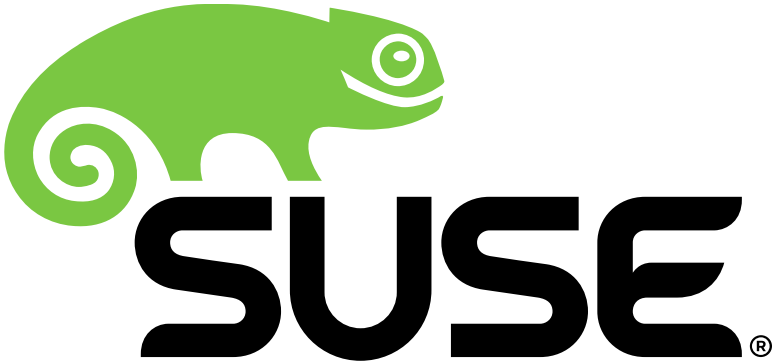
Geo Service Role State Machine



Lessons Learned

Lessons Learned

- Geo Clustering for SUSE Linux Enterprise High Availability Extension 11 SP3 was not robust enough for production deployment. We had to use the Geo Clustering extension from SUSE Linux Enterprise 12.
- We were forced to implement basic infrastructure such as filesystem replication
- Pacemaker and the geo cluster overlay did not always provide sufficient events for us to implement sophisticated resource agents. Our RAS rely heavily on regular monitor operations as an entry point to poll for events


SUSE®
We adapt. You succeed.