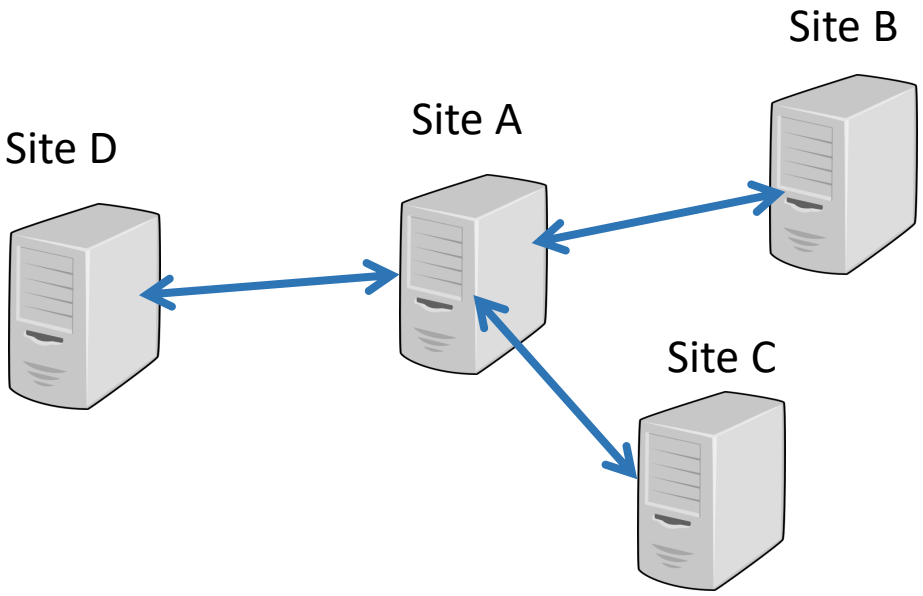
Samba KCC: Saying No to Full Mesh Replication

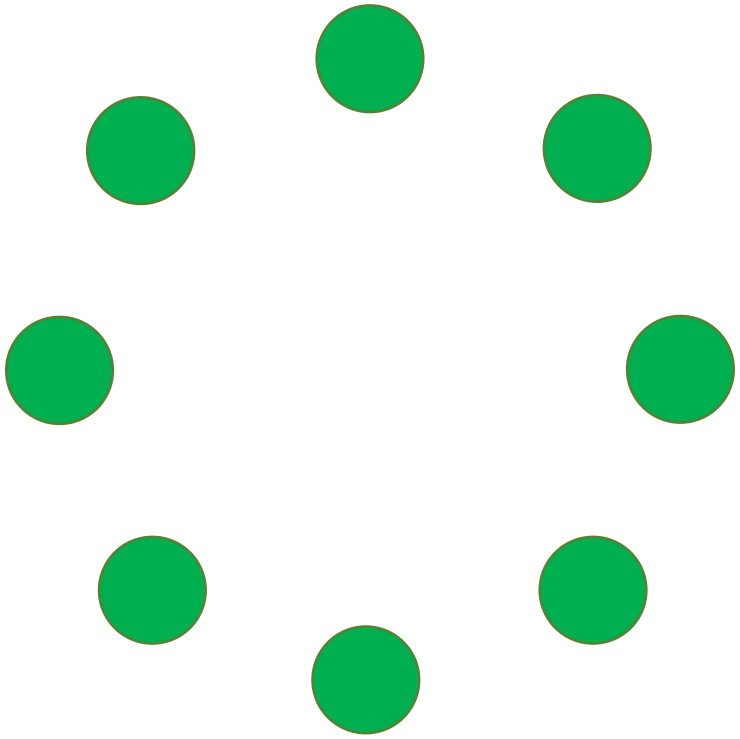
Garming Sam
Catalyst IT, Samba Team

What is the KCC?

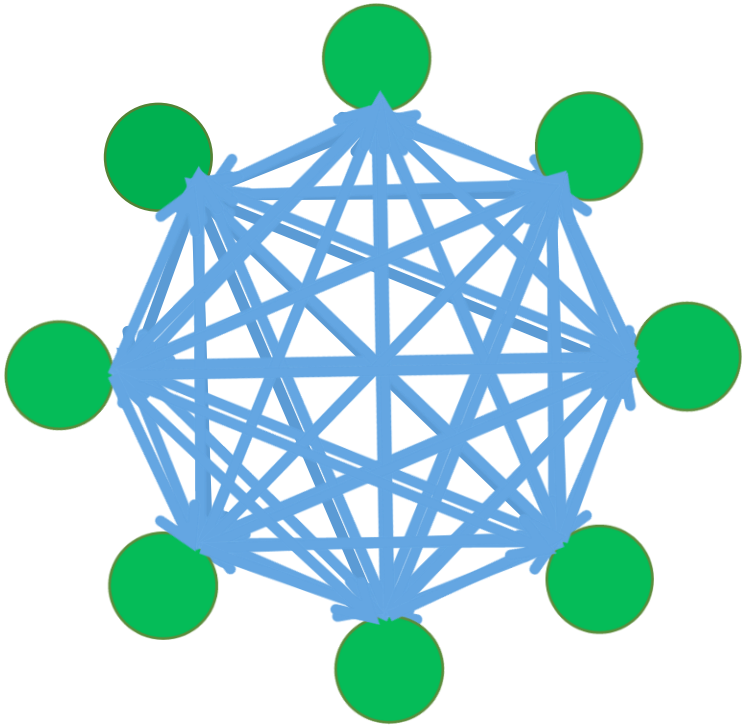
- Knowledge consistency checker
- Used to manage replication connections in AD
- Set of algorithms to produce efficient network topologies



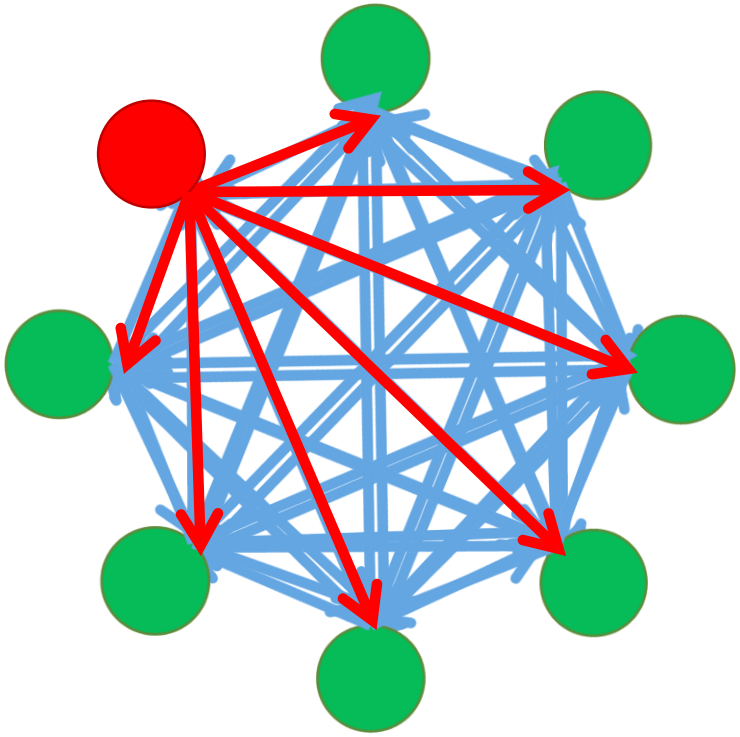
What is the KCC?



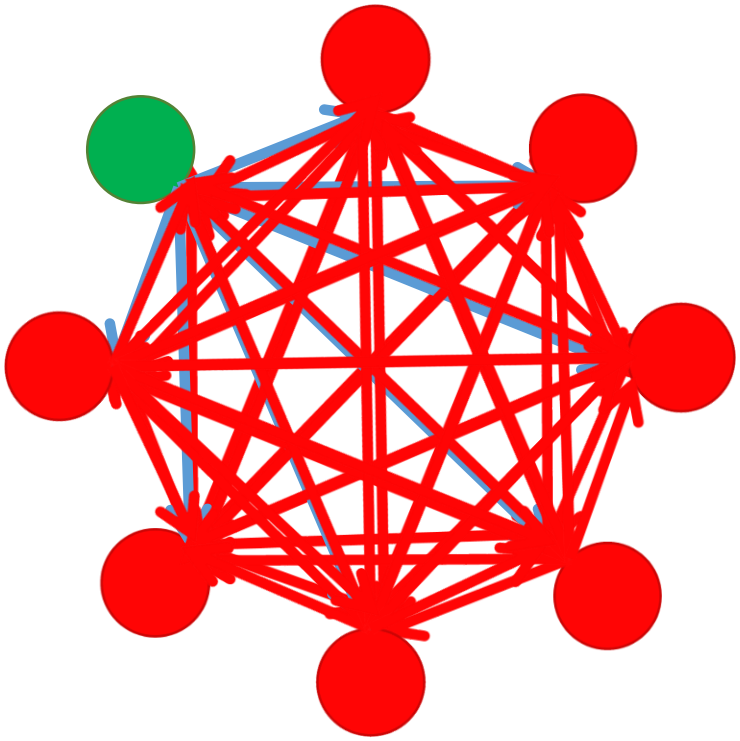
What is the KCC?



What is the KCC?



What is the KCC?



History of the KCC

- Original full-mesh C code
- Attempt at MS-ADTS algorithms in C
- Dave Craft (2011) on Python inter-site algorithms
- Late 2014—Early 2015 Douglas and myself
- Samba 4.3 introduced, Samba 4.5 set as default

Stages of the algorithm

- Intra-site algorithm
- Inter-site algorithm
- Removing unneeded connections
- Translate connections

Although the KCC creates ‘connection’ objects, they may not represent the underlying replication. They are only the implied connections given the current network topology.

Pre-requisites

- Transport – IP

```
dn: CN=IP,CN=Inter-Site Transports,CN=Sites,CN=Configuration,DC=example,DC=com
objectClass: interSiteTransport
```

- Sites – Default-First-Site

```
dn: CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=example,DC=com
objectClass: site
```

```
dn: CN=NTDS Site Settings,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=example,DC=com
objectClass: nTDSSiteSettings
interSiteTopologyGenerator: CN=NTDS Settings,CN=DC,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=example,DC=com
```

Pre-requisites

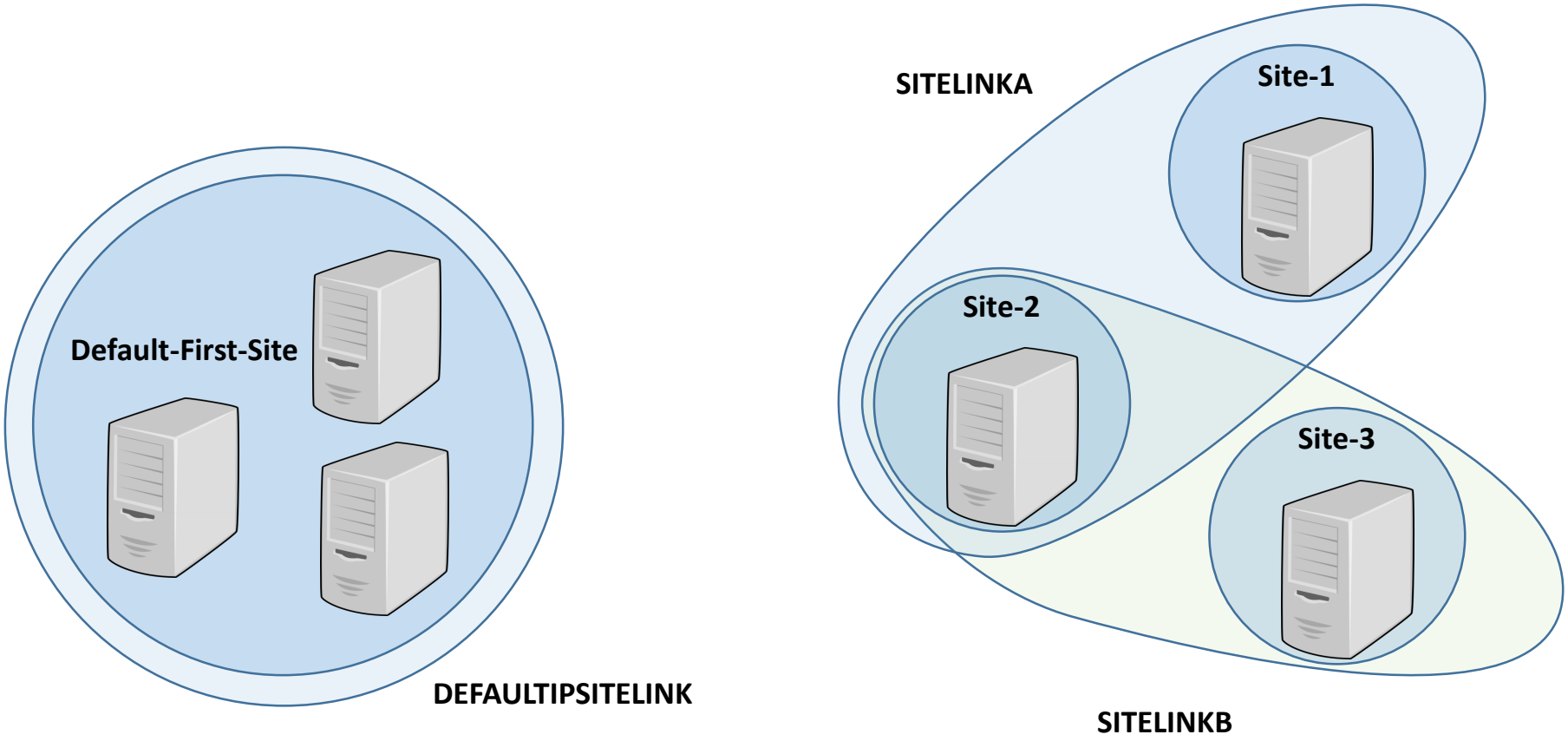
- Site-Links – DEFAULTIPSITELINK

```
dn: CN=DEFAULTIPSITELINK,CN=IP,CN=Inter-Site
Transports,CN=Sites,CN=Configuration,DC=example,DC=com
objectClass: siteLink
cost: 100
siteList: CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=example,DC=com
```

Site-links define the allowable connections between sites.

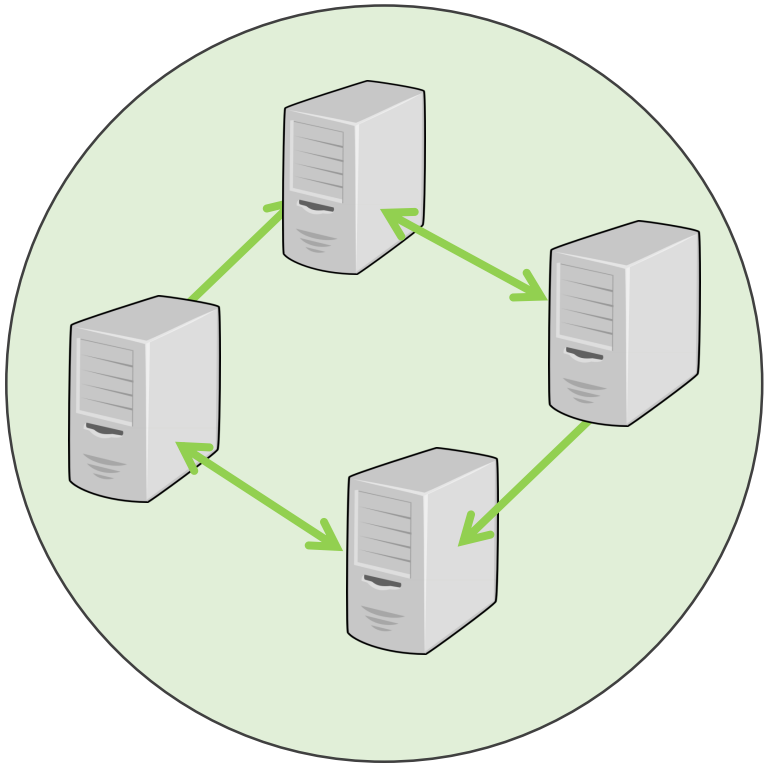
Site-links needs to collectively span your entire network.

Pre-requisites - Scenarios



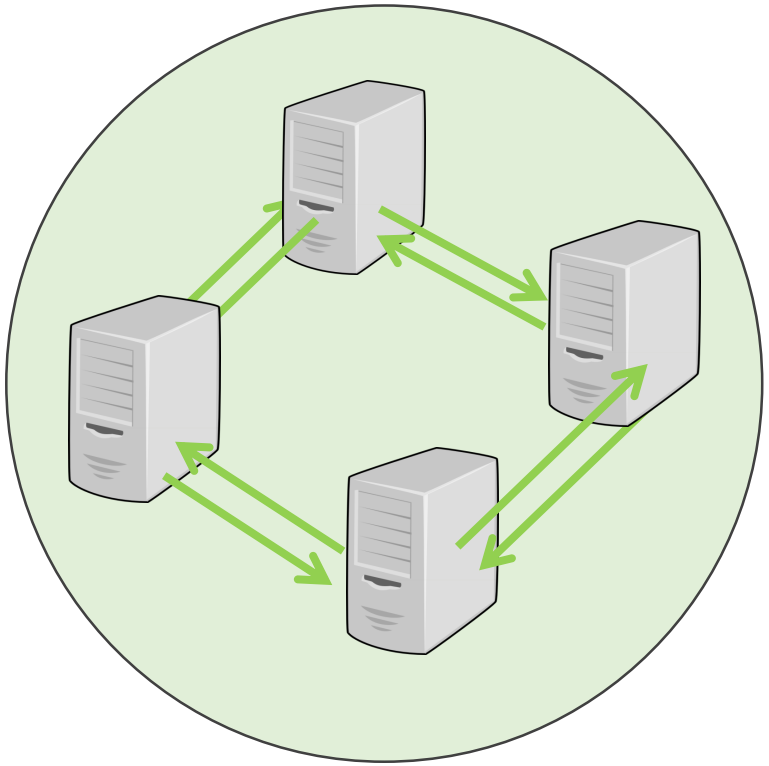
Intra-site algorithm

- Ring topology, with a few extra connections



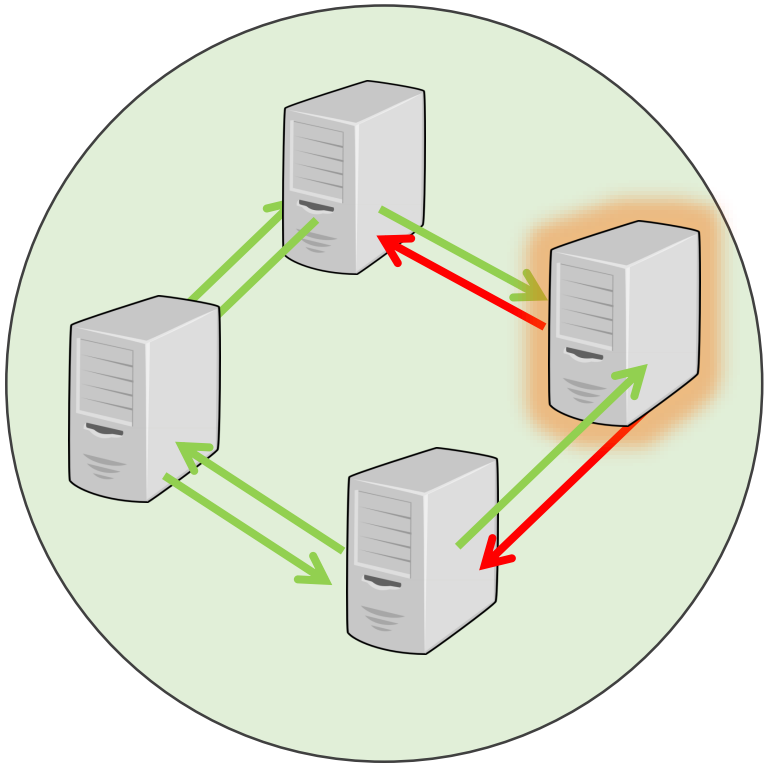
Intra-site algorithm

- Ring topology, with a few extra connections



Intra-site algorithm

- Every DC in the site has a sorted list of site DCs



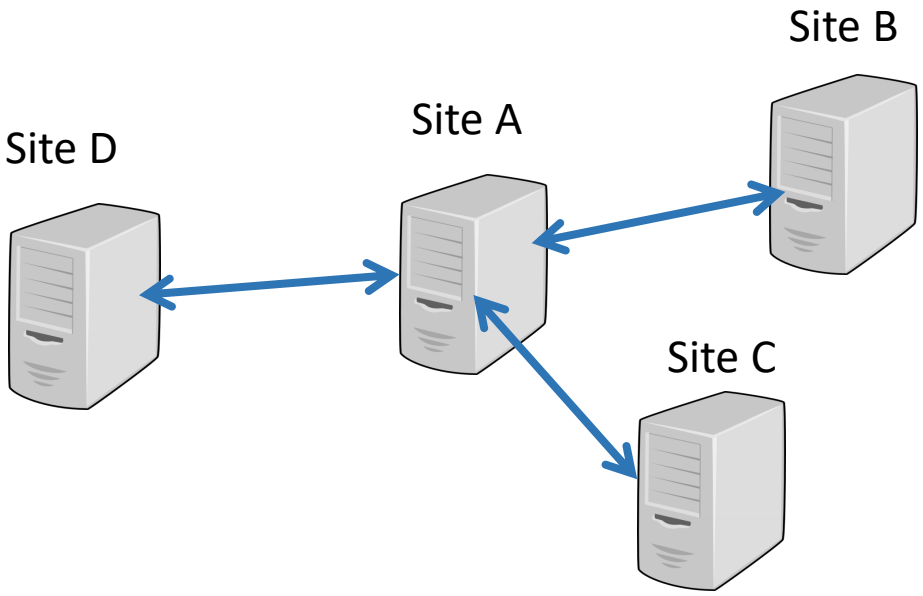
Intra-site algorithm

- Compared to the old KCC, there are fewer connections
- The algorithm is quite reliable, adding additional connections
- Information propagates in a more controlled manner

In a single-site use-case, with not that many DCs, behaviour should be quite similar to the old code.

Inter-site algorithm

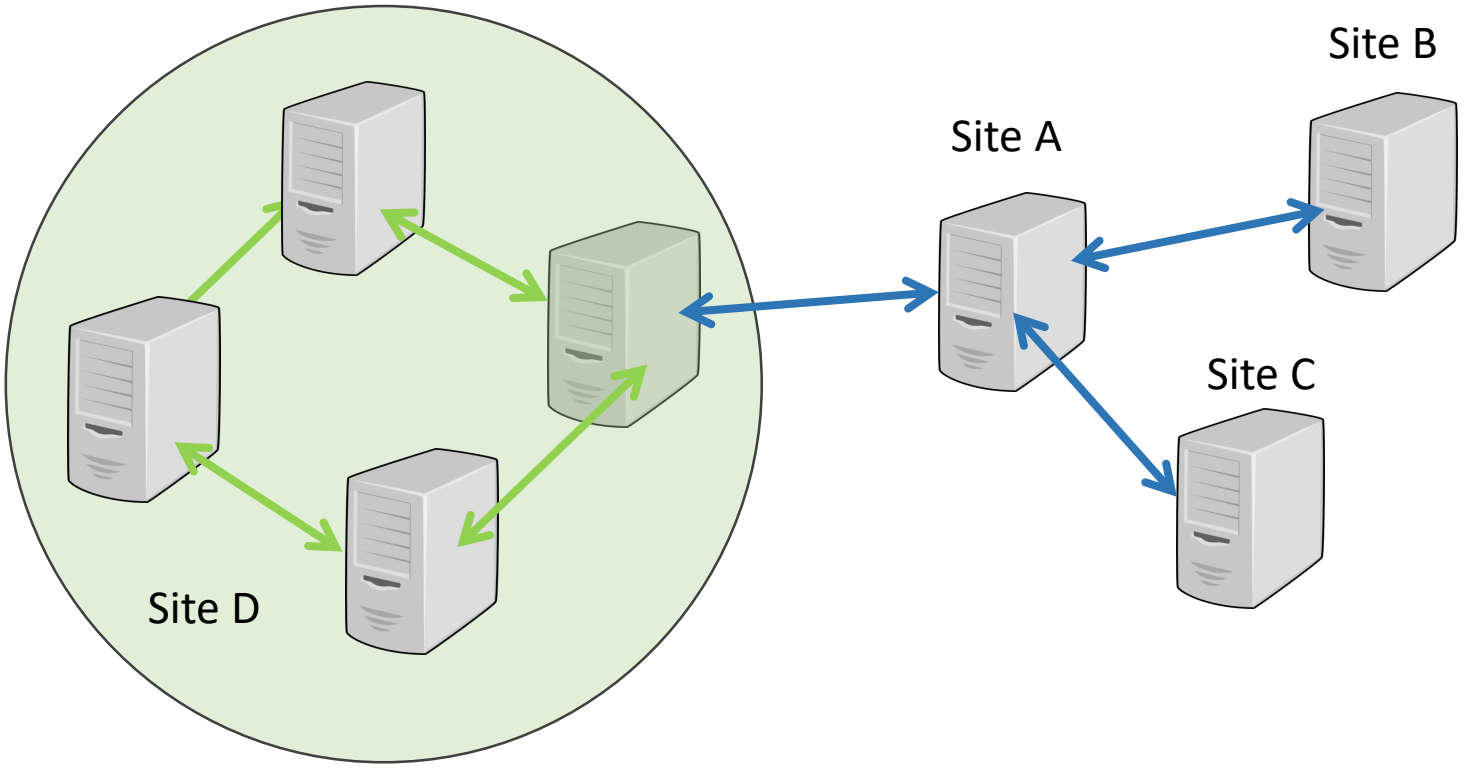
- Each site elects an inter-site topology generator (ITSG)
- Re-election attempts to occur if the ITSG is not responding
- Attribute: interSiteTopologyFailover



Inter-site algorithm

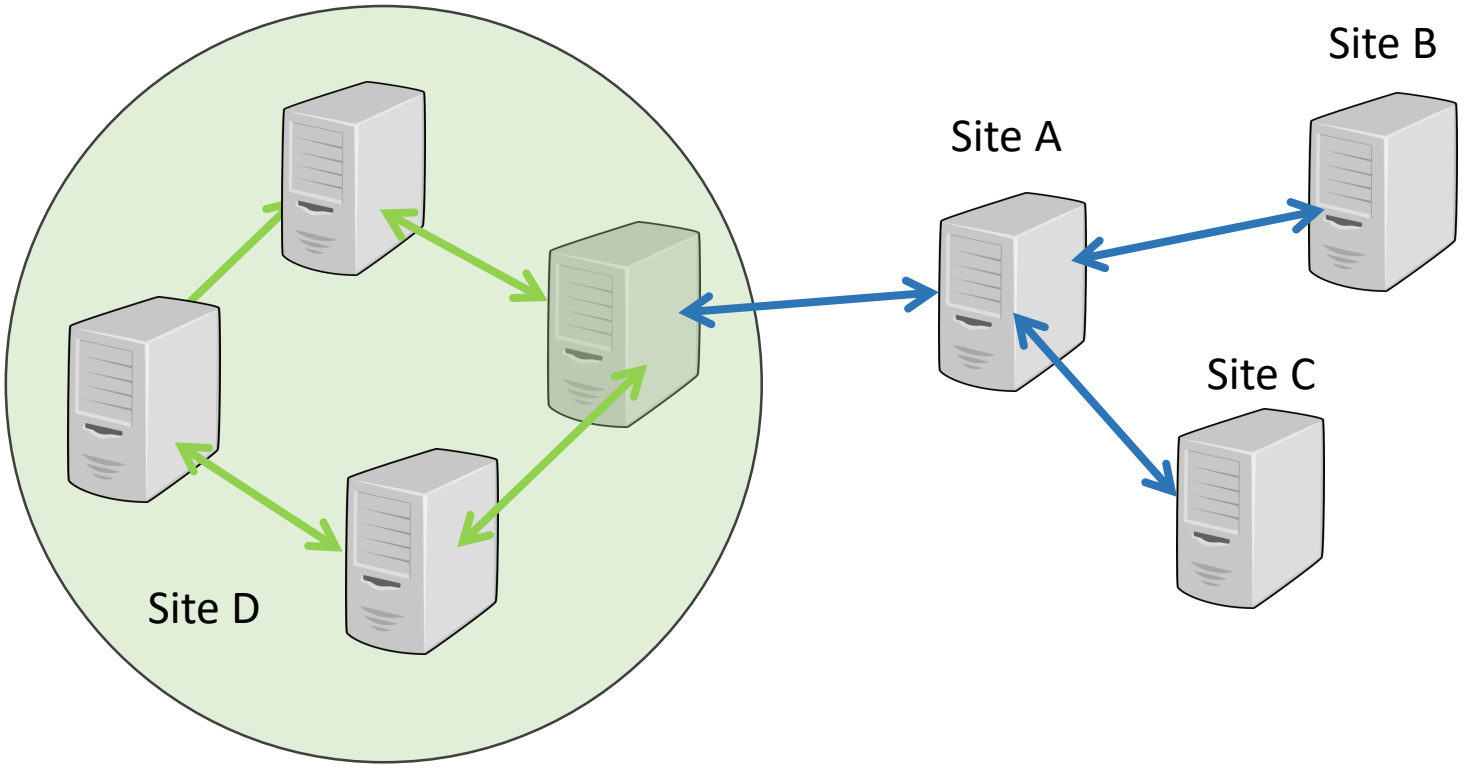
- Stable answer across entire DC network
- One DC per site managing inter-site connections
- Needs to be as fault tolerant as possible
- Must produce topology optimizing cost and schedules

Inter-site algorithm



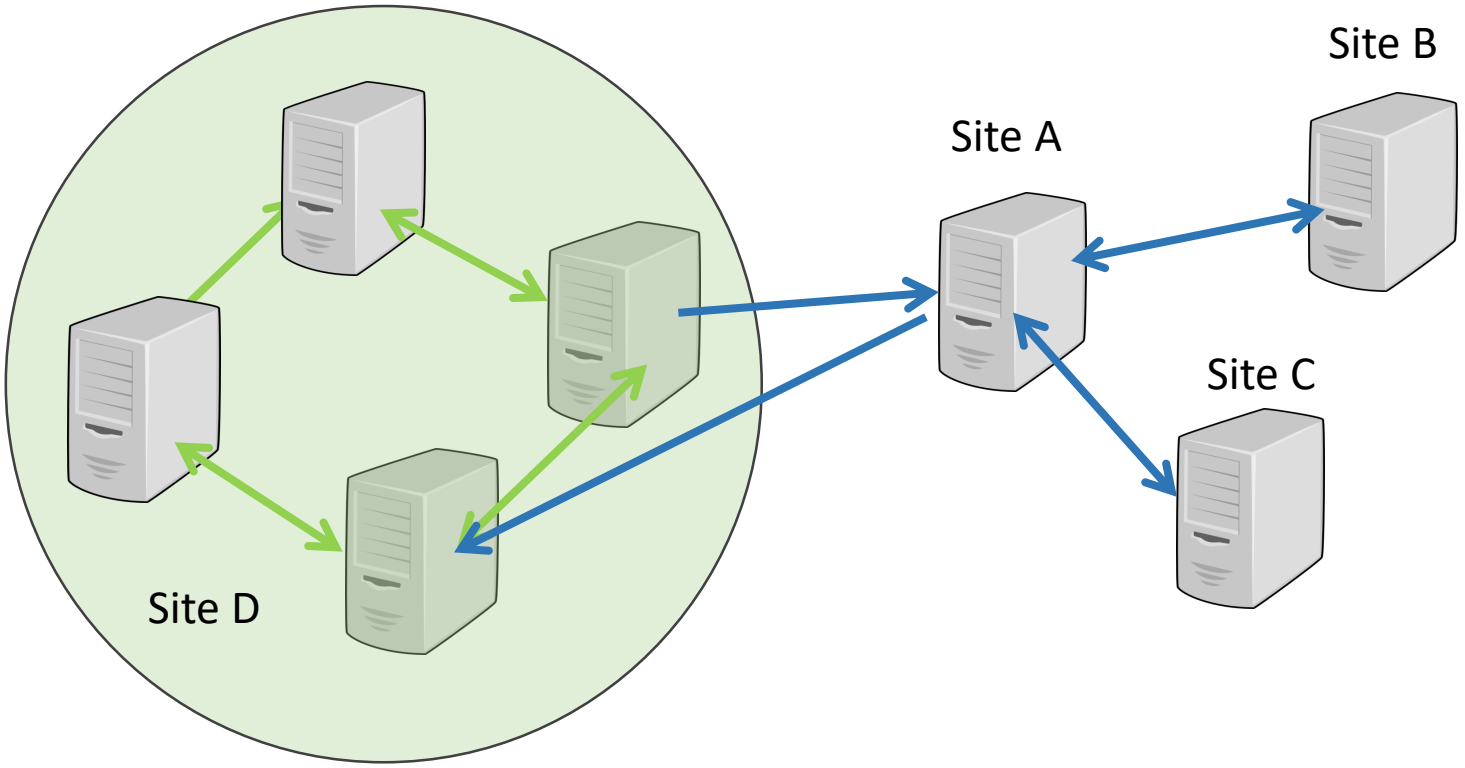
Bridgehead servers are the end-point connections between sites.

Inter-site algorithm



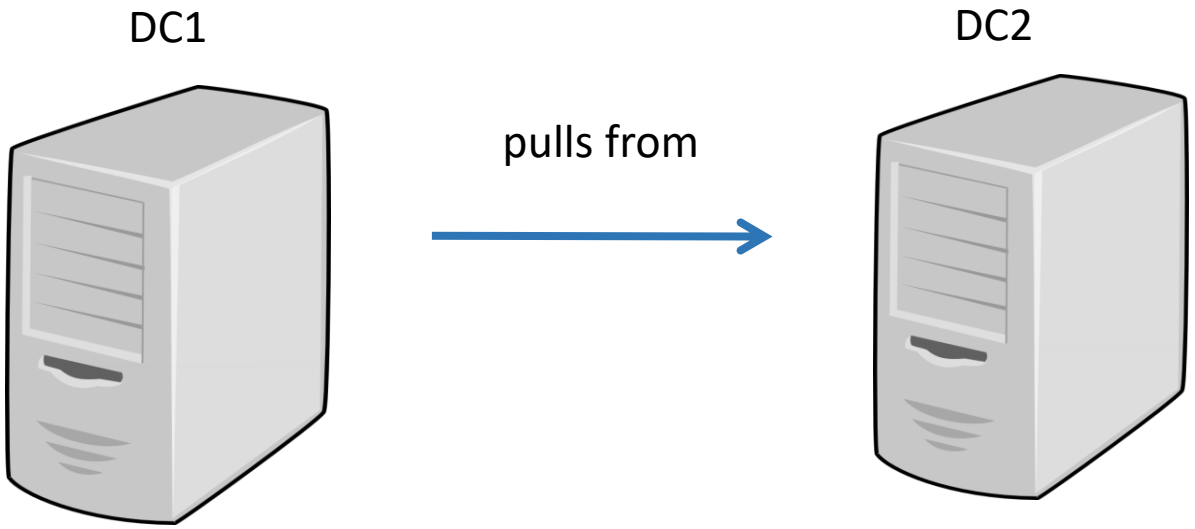
Being a bridgehead does not imply being an ITSG.

Inter-site algorithm



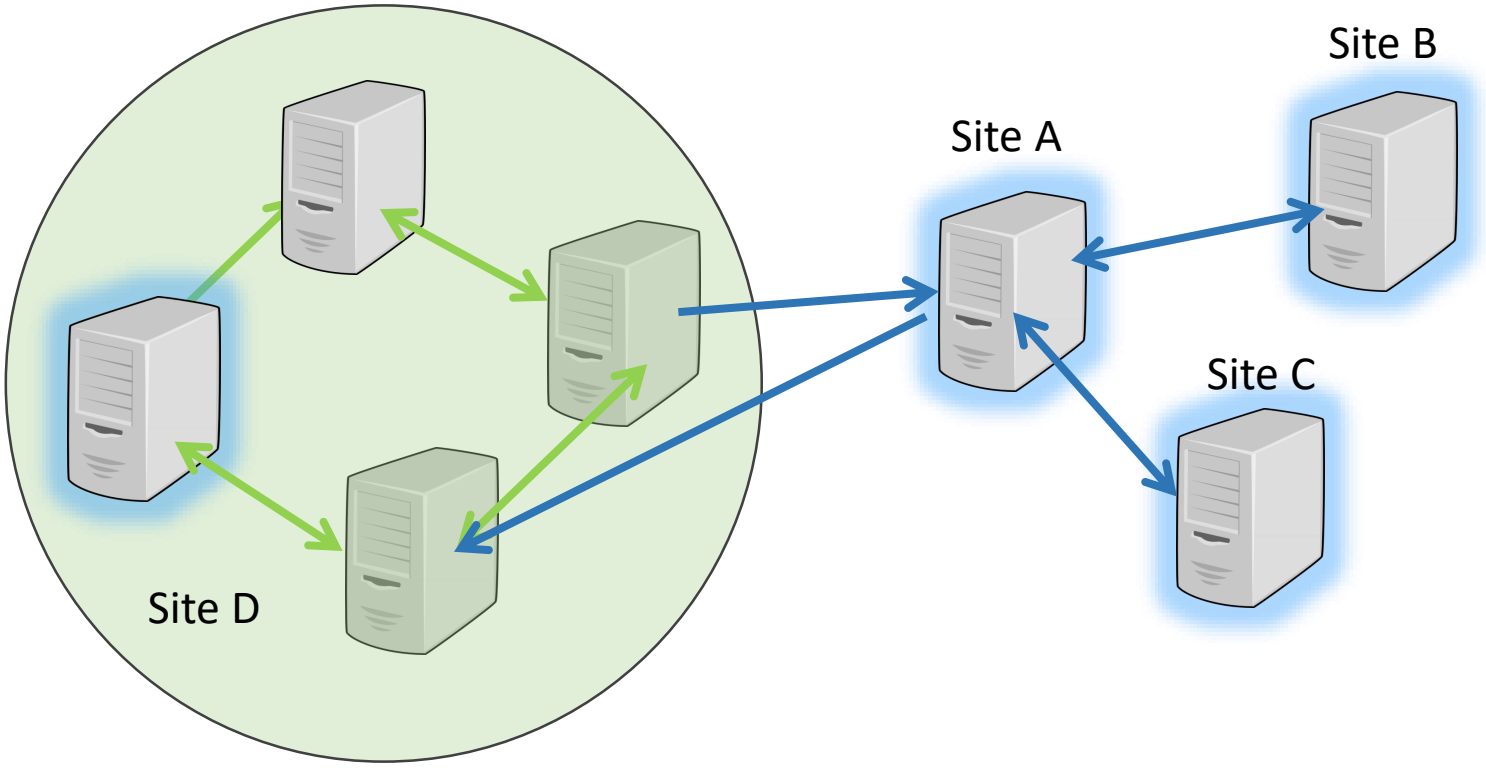
There is not necessarily a single bridgehead server.

Inter-site algorithm



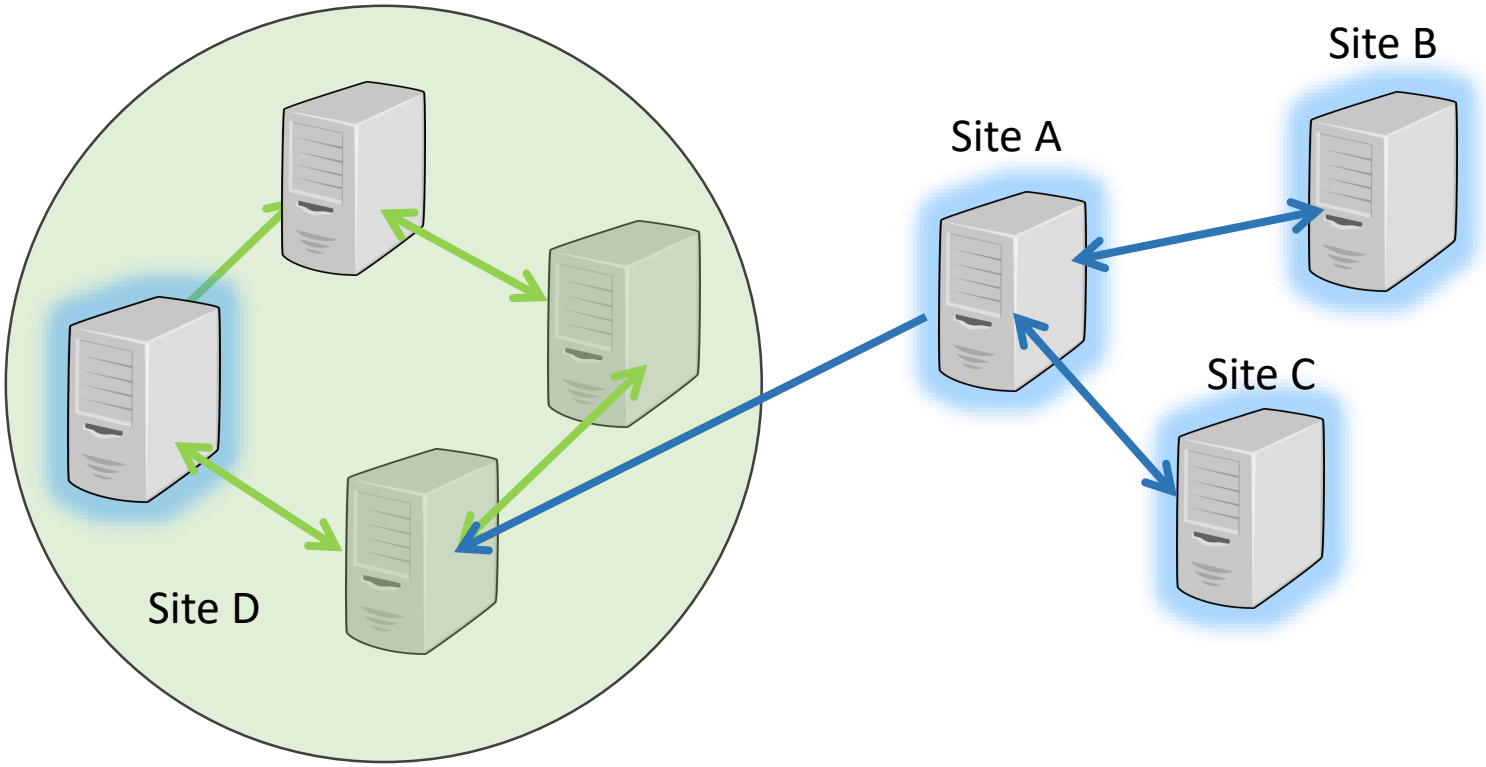
There is only pull replication.
Bi-directional replication must be done with two distinct connections.

Inter-site algorithm

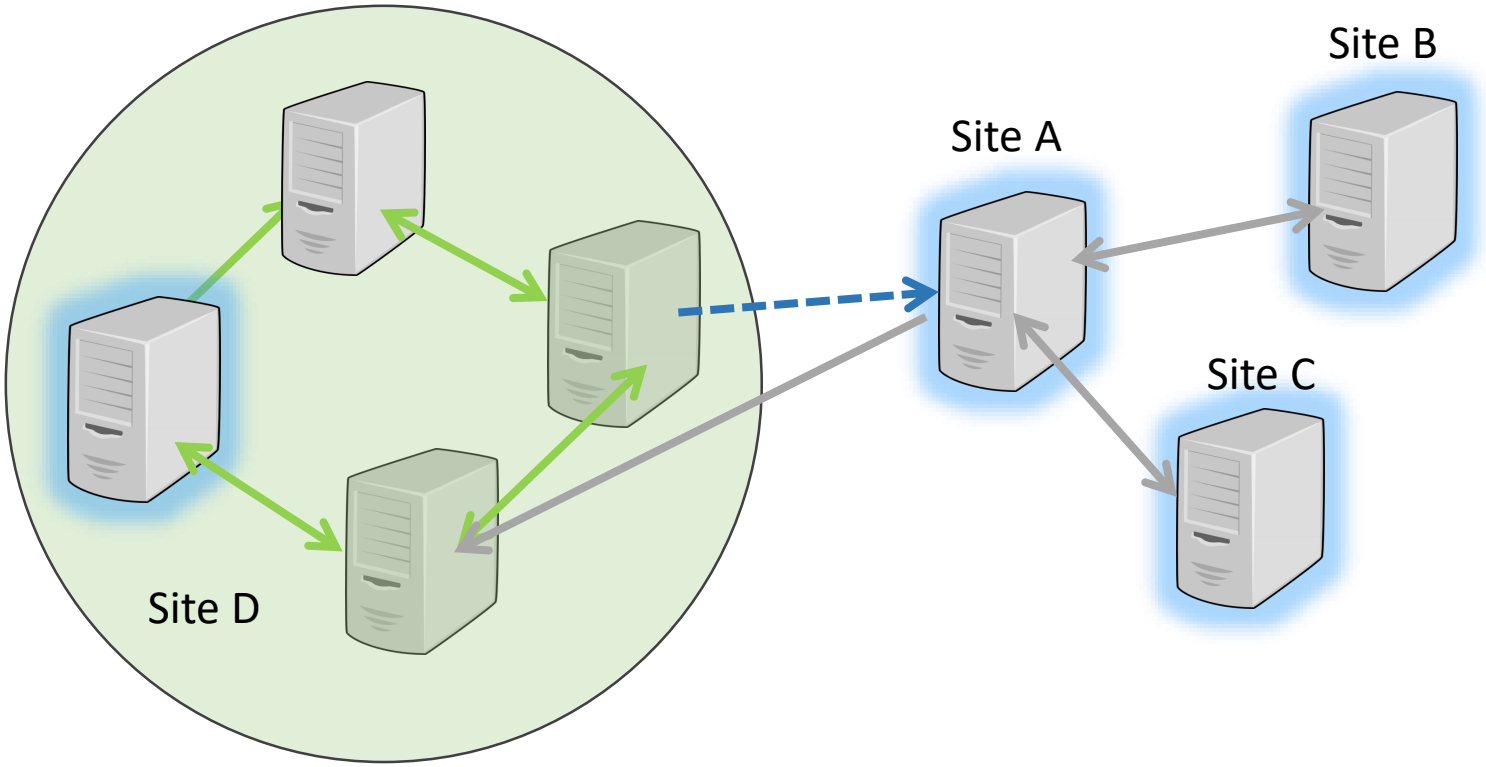


The inter-site algorithm only runs on the ITSG.

Inter-site algorithm

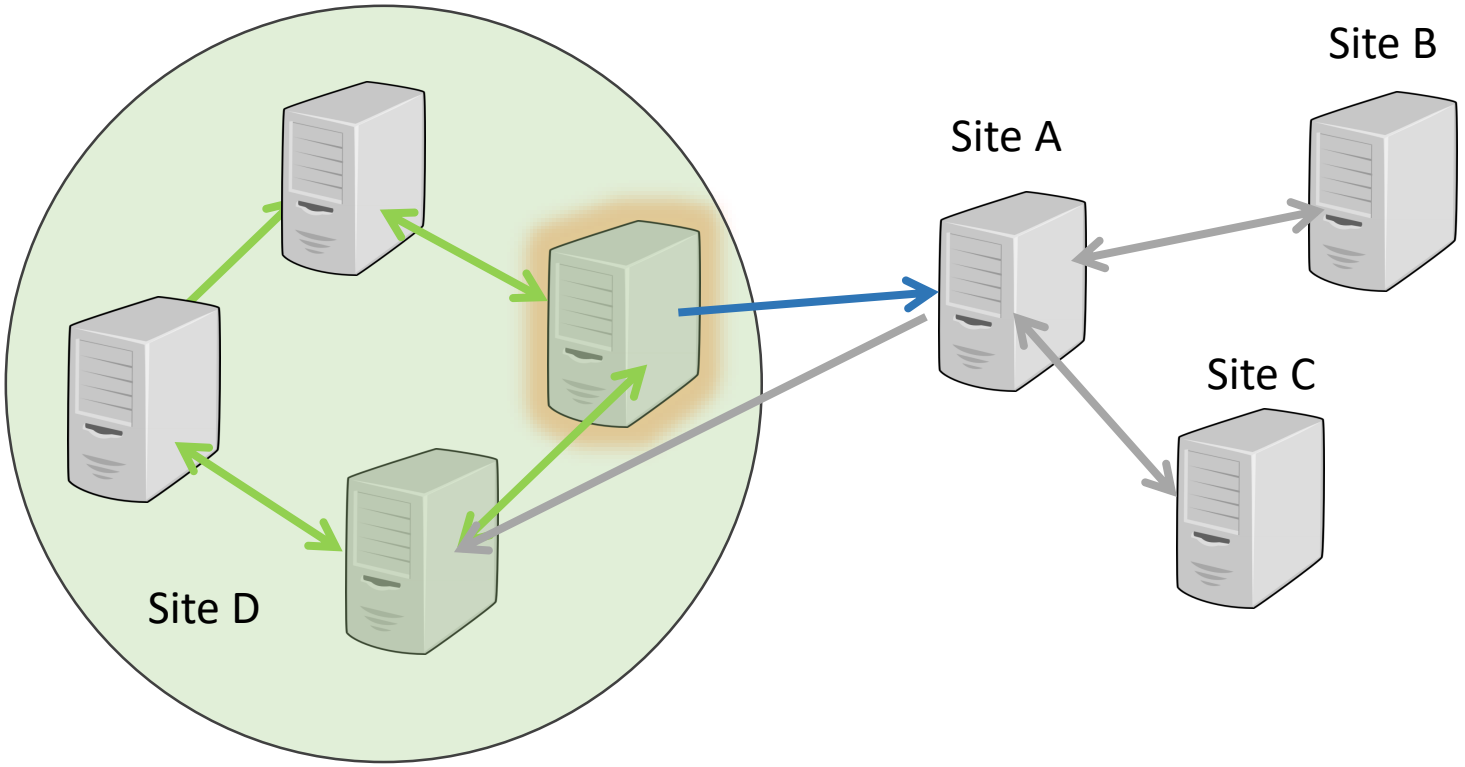


Inter-site algorithm



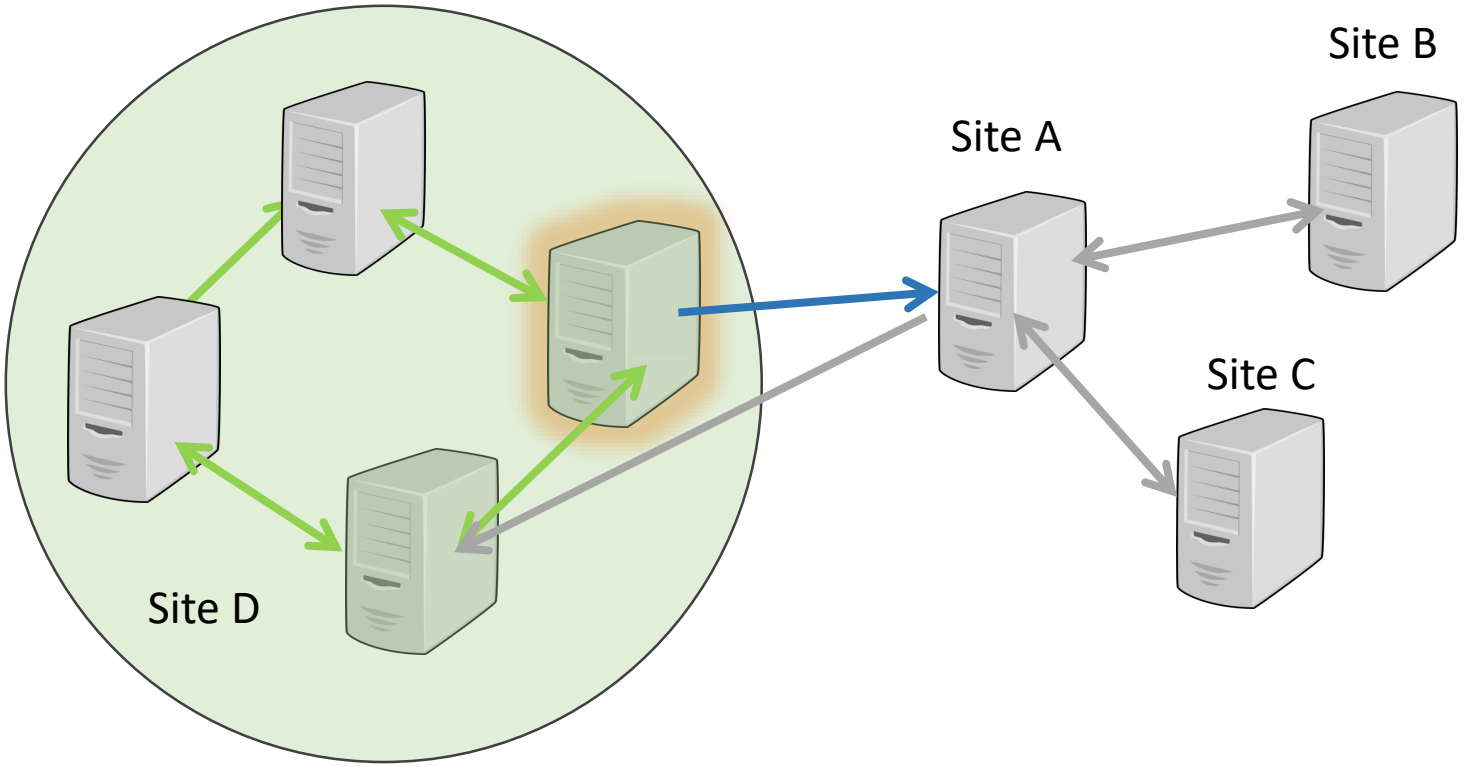
A new connection will be created in the database pointing to a randomly chosen bridgehead in Site A. Intra-site replication will propagate this to the necessary bridgehead in Site D.

Inter-site algorithm



The incoming bridgehead runs the KCC and notices the new connection.
It has no idea why it connects to the DC, that's the role of the ITSG.

Inter-site algorithm – TODO kruskal



The incoming bridgehead runs the KCC and notices the new connection.
It has no idea why it connects to the DC, that's the role of the ITSG.

Remove unneeded connections - TODO

- Stable answer across entire DC network
- Used to manage replication connections in AD
- Set of algorithms to calculate efficient network topologies
- ascending ReplInfo.Cost,
- descending available time in ReplInfo.Schedule,
- Overlapping schedule…
- One dc per site… you might want more connections (old KCC)

Two independent tasks running

- KCC running periodically
    - Creating NTDS Connection objects (ITSG or intrasite)
    - Translating NTDS Connections to repsFrom
- DREPL server
    - Reading repsFrom and pulling from the target

This means it can take some time to propagate.

Translate connections - TODO

- Stable answer across entire DC network
- Used to manage replication connections in AD
- Set of algorithms to calculate efficient network topologies

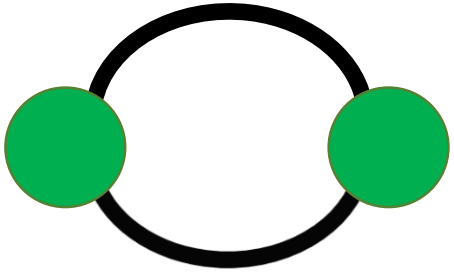
The overall effect

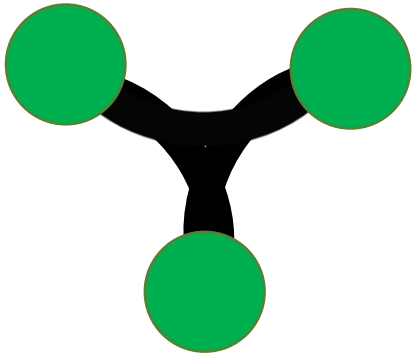
- Single path from any site to any site (property of a tree)
- Changes should not bounce around significantly
- Significantly reduced replication traffic
- Ability to customize who should talk to who

- Small networks (n <= 4) should have no visible effect
- Larger networks with varying connectivity shows huge effect

Challenges

- Verbose documentation
- ‘Multi-edge’, hyper-edge?
- White, red, black vertices?





More challenges

- Logical inconsistencies, ambiguities and omissions
- Pseudo-code vs textual description
- Easy to debug your own bugs
    - Testing?
    - --dot-file-dir
    - --export-ldif, --import-ldif

Incomplete features

- Trusted domains and global catalog replication
- RODC self-management
- Site-Link-Bridge Topologies
- Respecting schedules and other AD attributes
    - Preferred bridgehead servers

Incomplete features

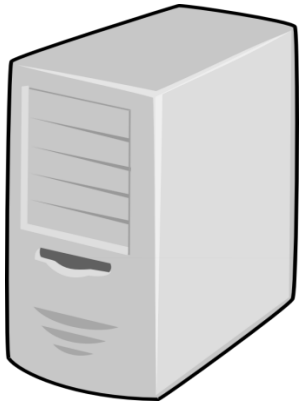
- Failed connection and failed DC failover
- Better stale connection clean-up
    - MS-DS-Replicates-NC-Reason
    - Use normal replication to propagate failure info
- Better debugging and failure information
- Better defaults for modern networks

Alternative topology strategies

- What is the best topology for various networks?

- Ring algorithm from intra-site for inter-site
- Minimum cost spanning tree plus additional connections
- Fully connected bridge-head servers

- With Samba 4.5, the new site-aware Samba Knowledge Consistency Checker (KCC) has been turned on by default. Instead of using full mesh replication between every DC, the KCC will set up connections to optimize replication latency and cost (using site links to calculate the routes). Although there is more effort required in establishing effective site topologies, it has enabled users to create larger and more distributed networks without any of the previous replication penalties. It has also meant that Samba AD can be aware of particular details of a network (such as satellite links or certain firewall restrictions) to ensure that information flows through the network in a reasonable way. The aim is to look at how sites in AD generally work, what role the KCC performs, and what implications this new feature has on a range of



```
.2637           # Step 1
.2638           self.refresh_failed_links_connections(ping)
.2640           # Step 2
.2641           self.intrasite()
.2643           # Step 3
.2644           all_connected = self.intersite(ping) # overlay a failed
network ontop of the expected
.2646           # Step 4
.2647           self.remove_unneeded_ntdsconn(all_connected)
.2649           # Step 5
```

.i1929 The KCC does not create more than 50 edges directed to a

.1930 single DC. To optimize replication, we compute that each node

.1931 should have n+2 total edges directed to it such that (n) is

.1932 the smallest non-negative integer satisfying

.1933 (node_count <= 2*(n*n) + 6*n + 7)

.Select ITSG

.1140 An intersite graph has a Vertex for each site object, a

.1141 MultiEdge for each SiteLink object, and a MutliEdgeSet for

.1142 each siteLinkBridge object (or implied siteLinkBridge). It

.1143 reflects the intersite topology in a slightly more abstract

.1144 graph form.