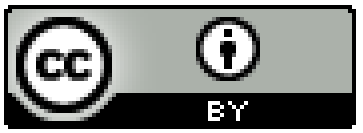
Raft Recursive Domains

On how-to replicate between separated Raft clusters

© 2016 Samo Pogačnik <*samo_pogacnik@t-2.net*>

Škofja Loka, 9.12.2016



Overview:

1. Raft Recursive Domains (replication between separated Raft clusters)
2. Drawing recursive domains
3. Asynchronous vs Synchronous secondary domains and commit modes
4. Inter-domain communication
5. Secondary leader operation – asynchronous domain
6. Primary domain switch-over
7. Still to cover
8. Resources

1. Raft Recursive Domains (replication between separated Raft clusters) [1/2]:

Purpose of this material...

- To explore possibilities
- To look for potential use cases
- To think about possible ways to achieve recursive domain replication
- To explore potential options/modes of recursive domain replication
- To define some terminology
- …

Speaking of terminology inside this material…

- Clusters being subject of replication are called **domains**
- Replication always takes place from the **primary** towards the **secondary** domain
- Domains are being called **recursive** (domains) in the perspective of how domain replication recursively triggers secondary domain (Raft consensus algorithm) log replication from within existing primary domain log replication.
- There may be **synchronous** vs **asynchronous** domain replication
- **Replication synchronism** refers to how secondary domain commits are being in sync with commits in the primary domain (after users receive confirmations about commits)
- Replication Synchronism always refers to secondary domains, therefore **synchronous** and **asynchronous** (secondary) **domains** (and **commit modes**).

1. Raft Recursive Domains (replication between separated Raft clusters) [2/2]:

Regarding possible ways to achieve recursive domain replication...

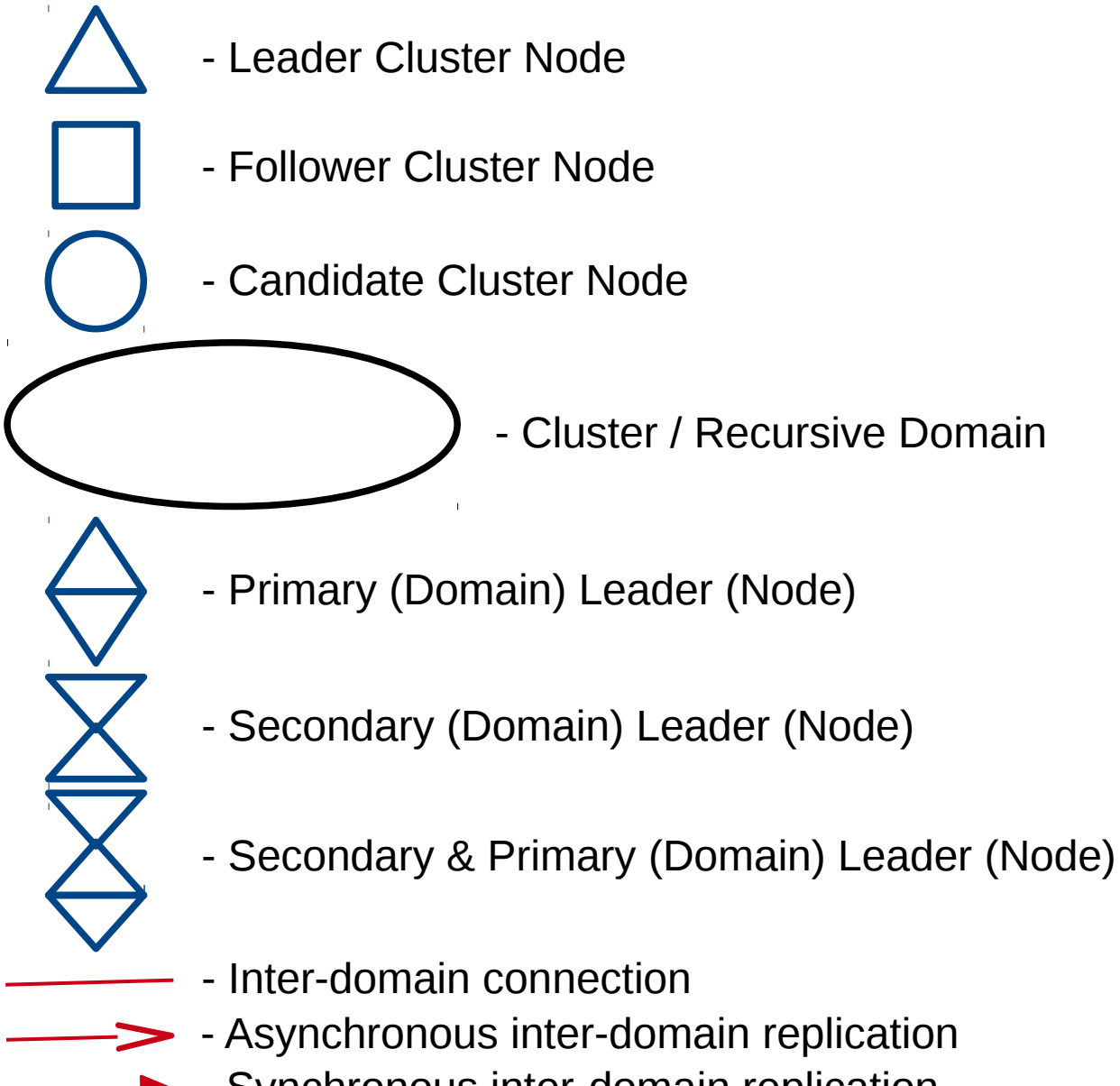
- There might be two general approaches

1. **It seems (to me) more natural to achieve secondary domain replication only via primary domain leader.**
   - Why? Primary leader replicates its log towards secondary leader in parallel with primary followers, using the same techniques as if the secondary leader was a real primary follower.
   - This approach is the primary focus of the following material.

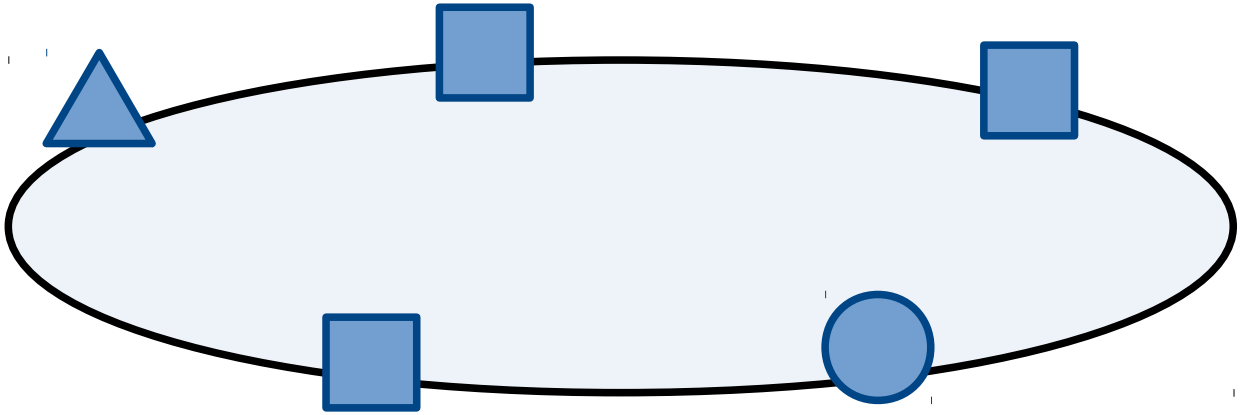
Let's just touch the other approach:

2. **How about replication to secondary domain via any of current primary followers instead via primary leader?**
   - Primary follower would start log replication towards secondary domain as special kind of a client of the secondary domain (however a secondary leader should contact primary followers).
   - This might work even better for asynchronous replication:
     - It would offload some primary leader processing to a primary follower. Actually a different follower could serve a different secondary domain.
     - This approach adds more to the commit latency within the secondary domain
     - Adds complexity in finding adequate primary followers, ...
   - Regarding synchronous commit mode, it would be necessary to make each primary follower serving secondary domain, a mandatory member of the commit quorum.

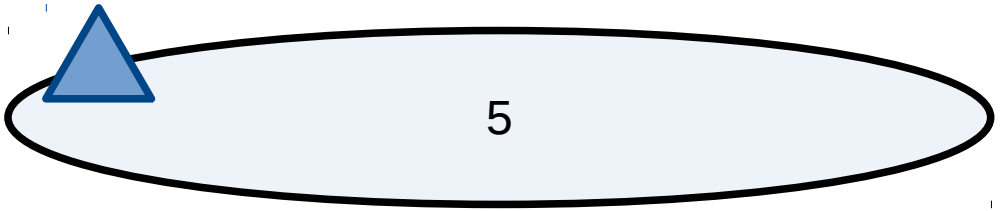
2. Drawing recursive domains - A legend:



2. Drawing recursive domains - A single domain:

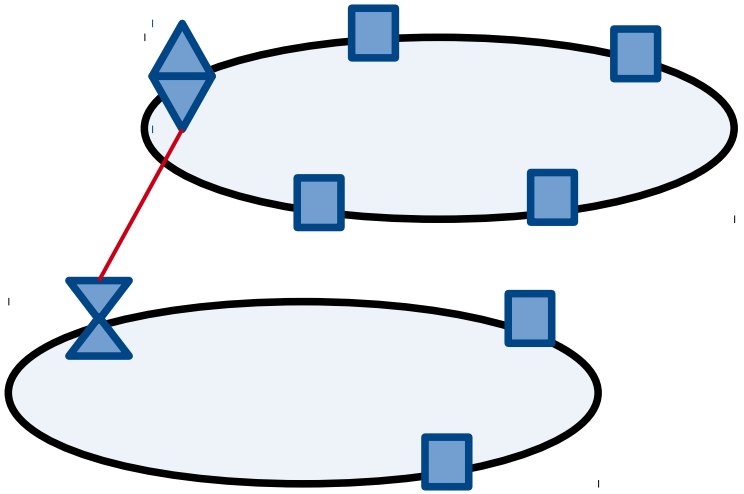


A single 5 node cluster in its current state of node roles.

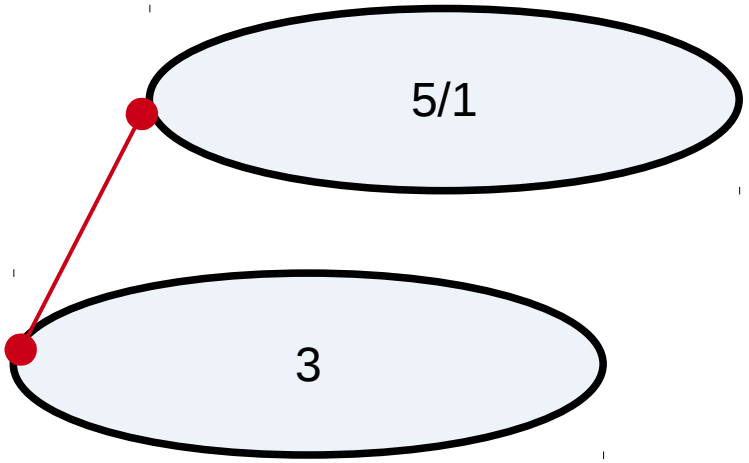


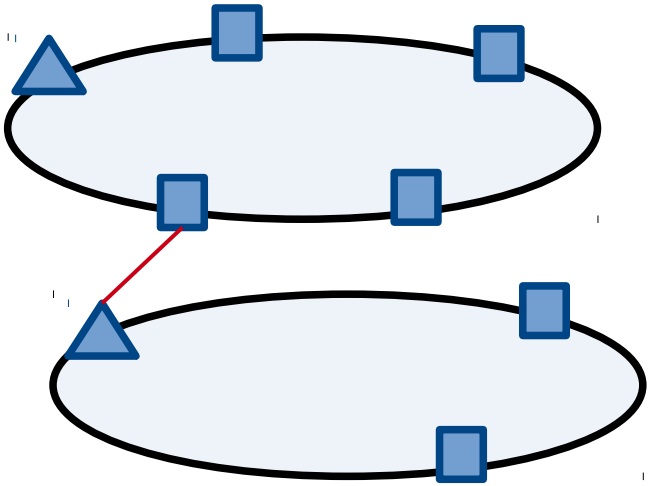
The same single 5 node cluster exposing only its current leader and number of cluster nodes.

2. Drawing Recursive domains - Leader based vs Follower based

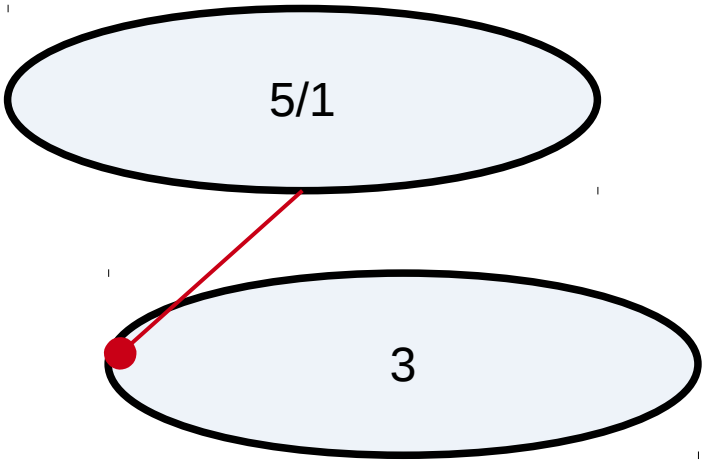


or

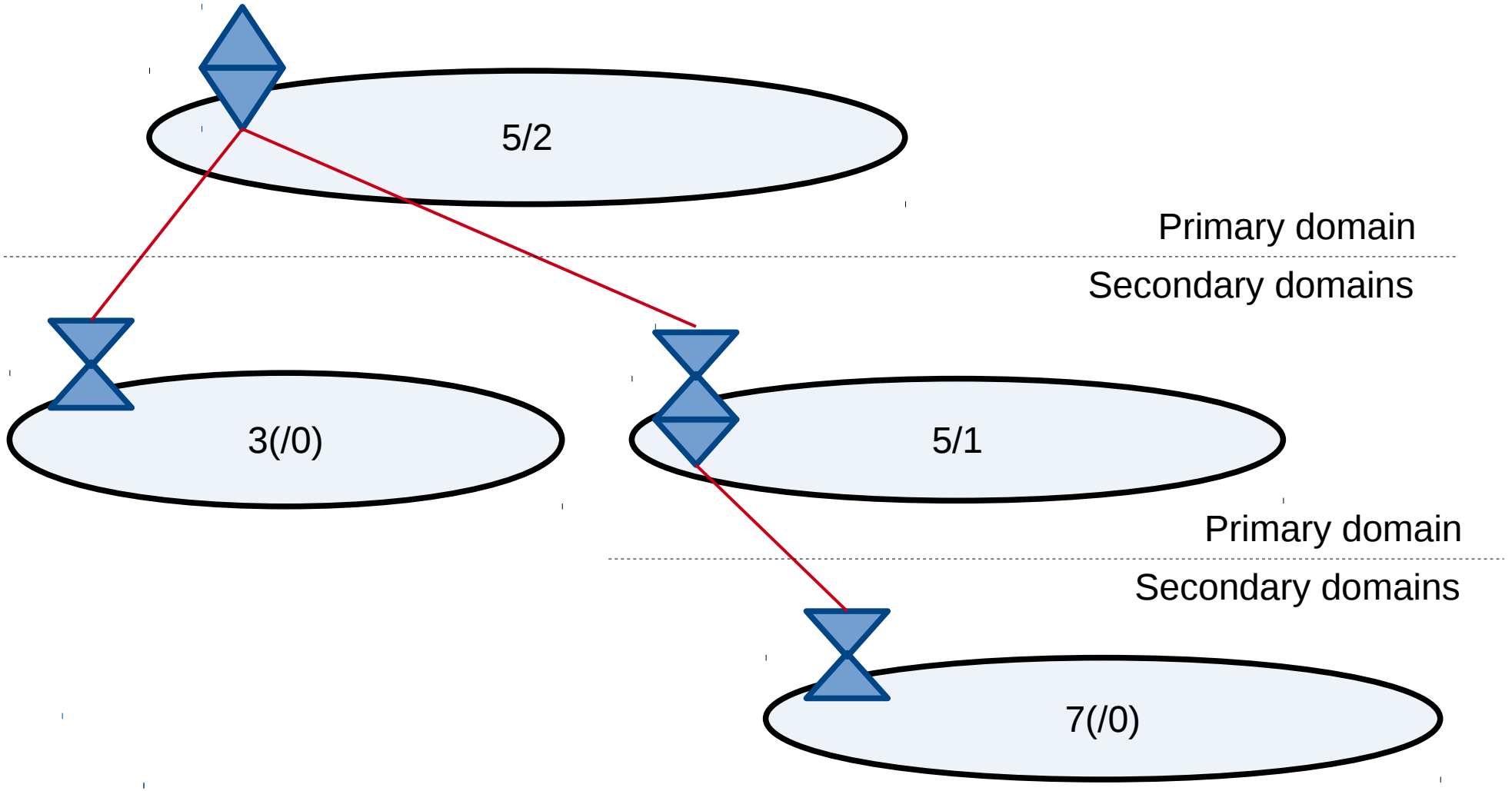




or

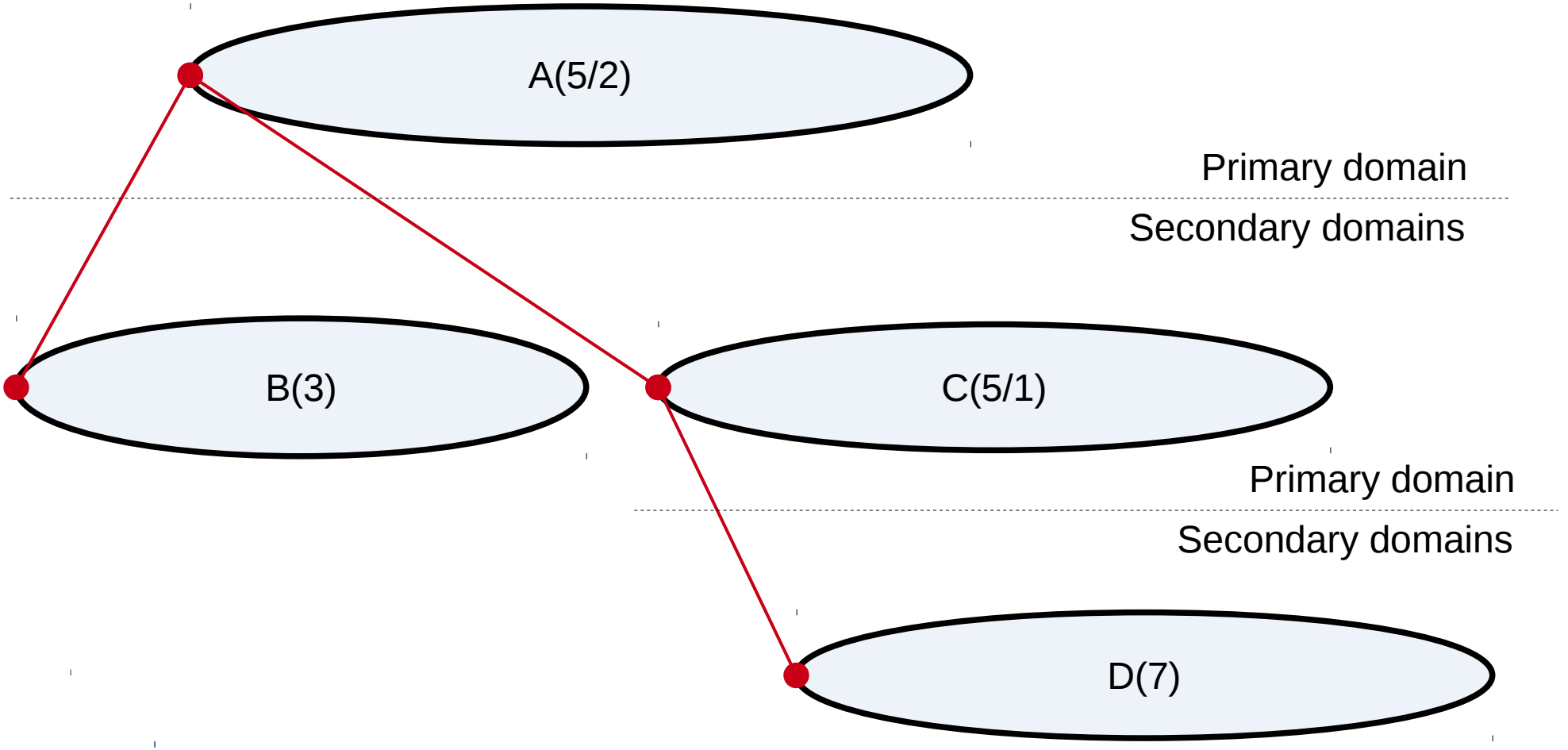


2. Drawing recursive domains - Layers of recursive domains:



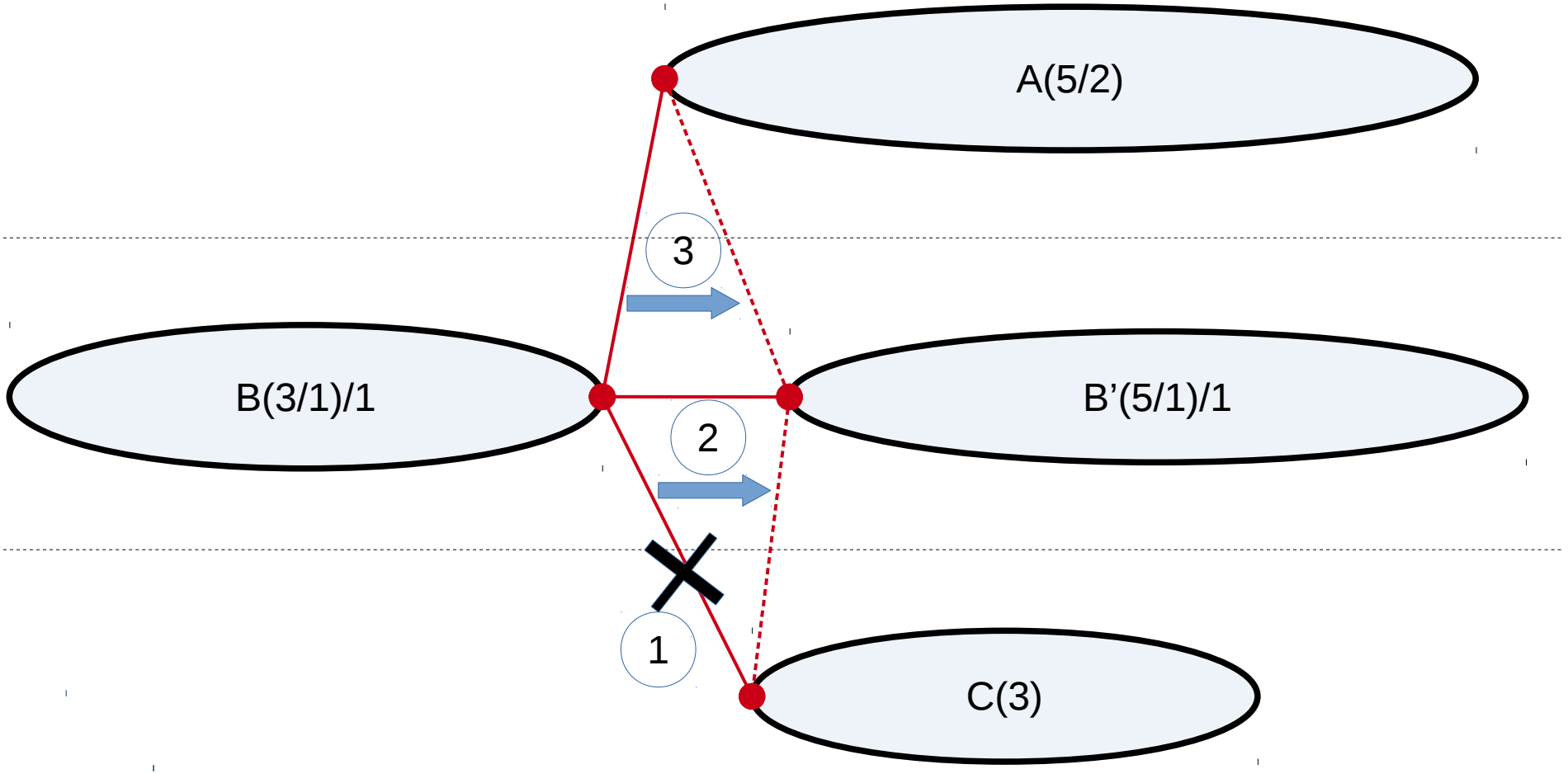
A layout of 5 recursive domains exposing their current leaders, a number of each domain nodes, slash a number of their direct subordinate recursive domains.

2. Drawing recursive domains - Same as previous (no leaders exposed):



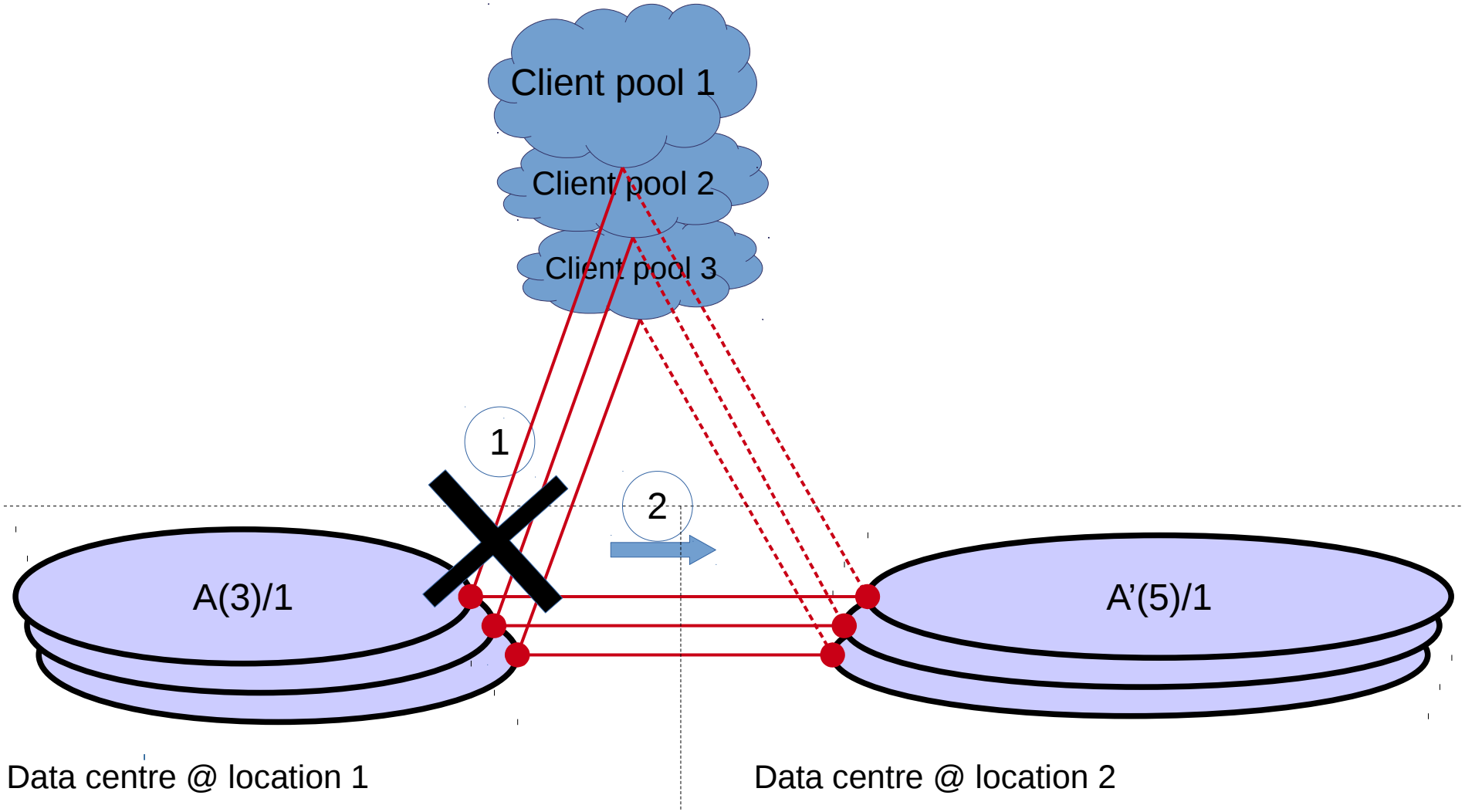
A layout of 5 recursive domains exposing a number of each domain nodes, slash a number of their direct subordinate recursive domains.

2. Drawing recursive domains - Geographically dislocated clusters B and B':



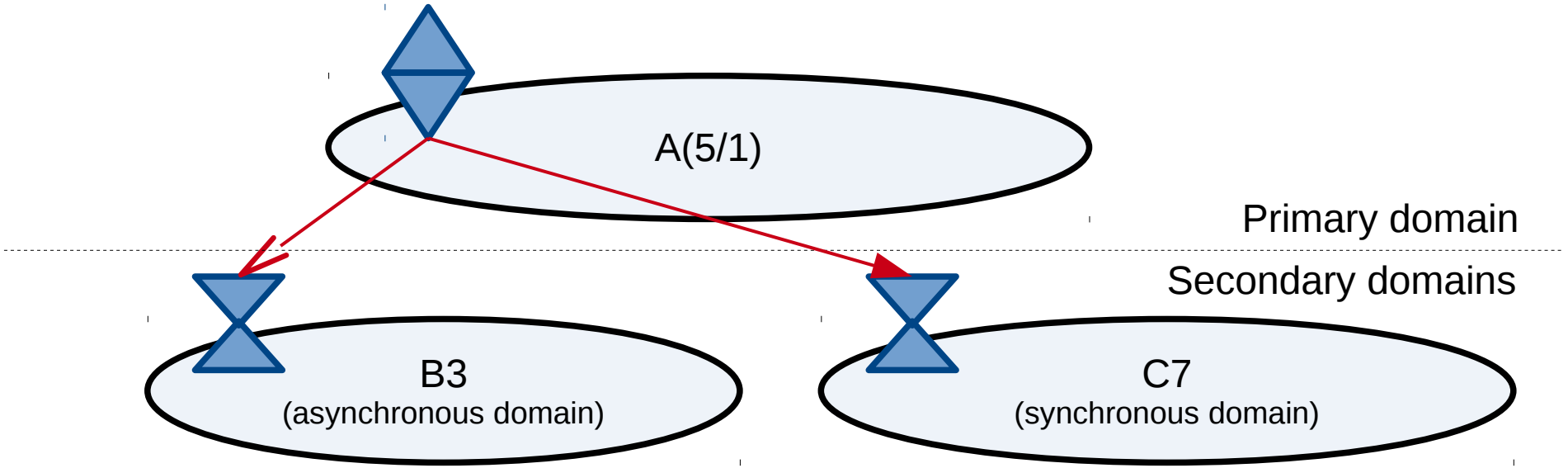
A layout of 4 recursive domains A, B, B' and C. Recursive domains B and B' switch each others Primary/Secondary relation, after connectivity outage between domain B and domain C. What if there is also a domain B''?

2. Drawing recursive domains - Again some geo-dislocated clusters with clients:



A layout of 3 primary recursive domains A, B and C geographically replicated for disaster recovery scenario (switching primary roles to domains A', B' and C').

3. Asynchronous vs Synchronous secondary domains and commit modes:



Asynchronous secondary commit mode?

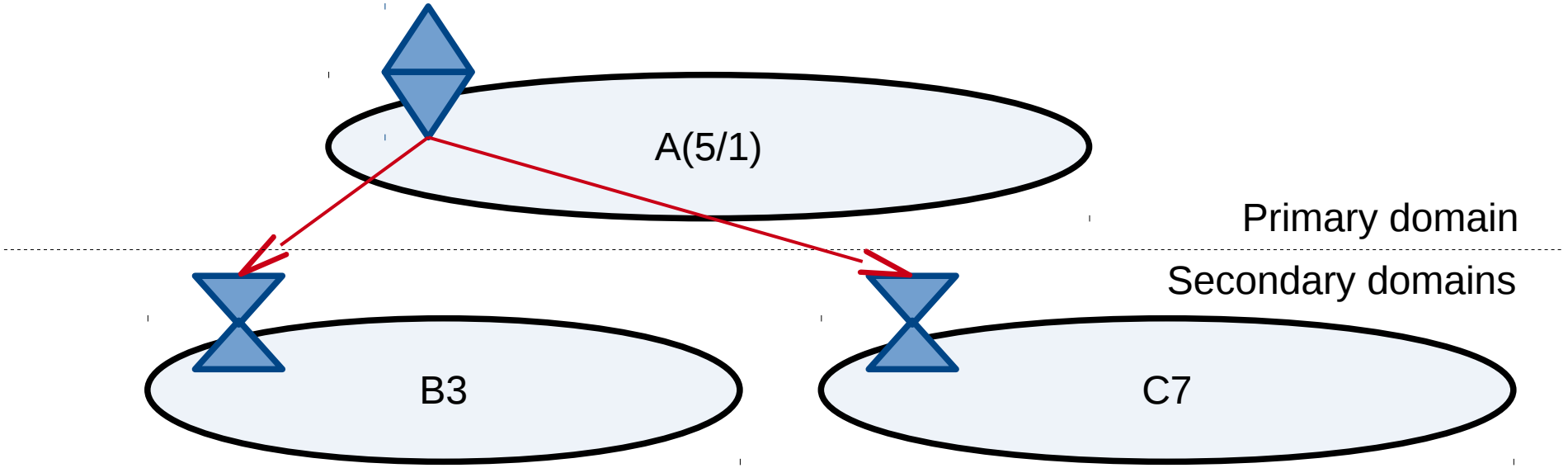
1. In this mode of operation, each commit within the secondary domain happens asynchronously with every commit within the primary domain.
2. Clients receive commit confirmations after commits have been confirmed within the primary domain only (majority of just primary nodes) and secondary commits complete eventually (after or even before clients receive commit confirmations).

Synchronous secondary commit mode?

1. In the synchronous secondary commit mode, clients receive commit confirmations only after a majority of confirmations from primary nodes has been reached and after all synchronous secondary domains received commit confirmations from majority of their secondary nodes.
2. When a client successfully commits a new state, it is assured that all synchronous domains already share this same state.

- **Mode of operation does not affect any leader election in any domain** (secondary leaders are always followers within the primary domain and as such they never participate in primary leader elections)**!**
- **Clients commit new states faster in the asynchronous mode!**
- **Atomicity of client operations over asynchronous domains is lost!**

4. Inter-domain communication:



How to include secondary leaders into the primary domain?

1. Awareness of all secondary domains need to be stored within the primary domain!
2. Secondary leaders get elected via normal Raft elections within each secondary domain.
3. Secondary leaders connect to the primary domain's leader via the same mechanisms as clients (i.e. connect to any primary domain node and redirect to their leader) – a unique domain ID separates secondary leaders from clients.

How does a secondary connection affect primary domain (asynchronous commits)?

1. Let's keep things simple and assume static configuration of secondary domains within the primary domain.
2. Also assume asynchronous commits from the primary to all secondary domains.
3. Saying that, any primary-secondary interaction does not count into any quorum (majority) decision (not for primary leader election nor for commit quorum).
4. Only log replication needs to be performed additionally from primary to each secondary leader.
5. A client also does not need to wait for secondary leaders to confirm commit within their secondary domain.

4. Inter-domain communication (continued):

How to include secondary leaders into the primary domain?

1. A primary domain does not need to know all secondary nodes from each secondary domain.
2. Instead unique domain IDs would be used to recognise each secondary leader, when primary domain nodes are being contacted.
3. Using domain IDs, secondary leaders would be differentiated from clients, when connected to primary leader.

How does a secondary connection affect primary domain?

1. Any secondary leader is always a follower in the primary domain, meaning that secondary domains do not affect primary domain leader elections in any way. Not even during primary domain configuration change, when a secondary domain is added or removed from configuration.
2. ...

What if synchronous commit mode was required for some secondary domains?

1. This requirement does not affect any raft leader elections (again, secondary leader is always a follower in the primary domain).
2. In this case, additional response from each synchronous secondary domain leader is required (beside the majority of responses from primary followers), to consider a new entry committed.
3. This kind of operation could be very problematic:
   a) Slow secondary response causes delays in client commits.
   b) Unstable secondary leader causes even more delays in client commits.
   c) An outage of a synchronous secondary domain blocks all primary commits. A potential solution would be to prevent secondary domain's data usage, until secondary domain reconnects and catches primary domain. While secondary domain can not be used, primary domain excludes secondary leader confirmations from commit quorum.
   d) ...

5. Secondary leader operation – asynchronous domain:

Within the primary domain, the secondary leader...

- Regarding log replication, the secondary domain leader operates as any other Raft follower in the primary domain.
- When a secondary domain leader stops receiving heartbeats from the primary domain leader, the secondary leader starts reconnection procedure to restart receiving heartbeats from the primary leader.
- While secondary leader reconnection procedure takes place, the secondary leader may be redirected to the new primary leader, if another primary leader has been elected in the mean time.

Within the secondary domain, the secondary leader…

- Plays normal Raft leader role within its secondary domain.
- Takes primary leader's input for its secondary domain log replication process.
- Potentially, a secondary leader maintains its extended replication log in the following way:
  - Each log entry contains a state machine command and each additional information (i.e. the term) twice. Once for its primary domain follower role and once for its secondary domain leadership.
  - It may be that some of primary domain additional data (like primary term) would have to be replicated together with state machine command within the secondary domain.
- ...

6. Primary domain switch-over:

It would be very interesting to explore the possibility to switch one secondary domain into a primary one under certain conditions:

1. Only two domains and only two levels?
2. More domains, more levels?
3. ...

7. Still to cover:

- **About affecting Raft's Safety requirement.**
- **There should be more said about potential use cases.**
- **There are a lot of details and scenarios about inter domain interaction to be exposed and resolved.**
- **…**

8. Resources:

- **Raft Consensus Algorithm** (https://raft.github.io/)
- **In Search of an Understandable Consensus Algorithm by Diego Ongaro and John Ousterhout** (https://raft.github.io/raft.pdf)
- **Raft user Study** (video: https://raft.github.io/raft.pdf, PDF: https://ramcloud.stanford.edu/~ongaro/userstudy/raft.pdf)
- **Diego Ongaro's Ph.D. dissertation** (https://github.com/ongardie/dissertation#readme)
- **raft-dev mailing list** (https://groups.google.com/forum/#!forum/raft-dev)
- **Raft "locations" concept proposal** (https://groups.google.com/forum/#!topic/raft-dev/o1_vMMUZN4c**)**
- **…**

Thank you for your interest!