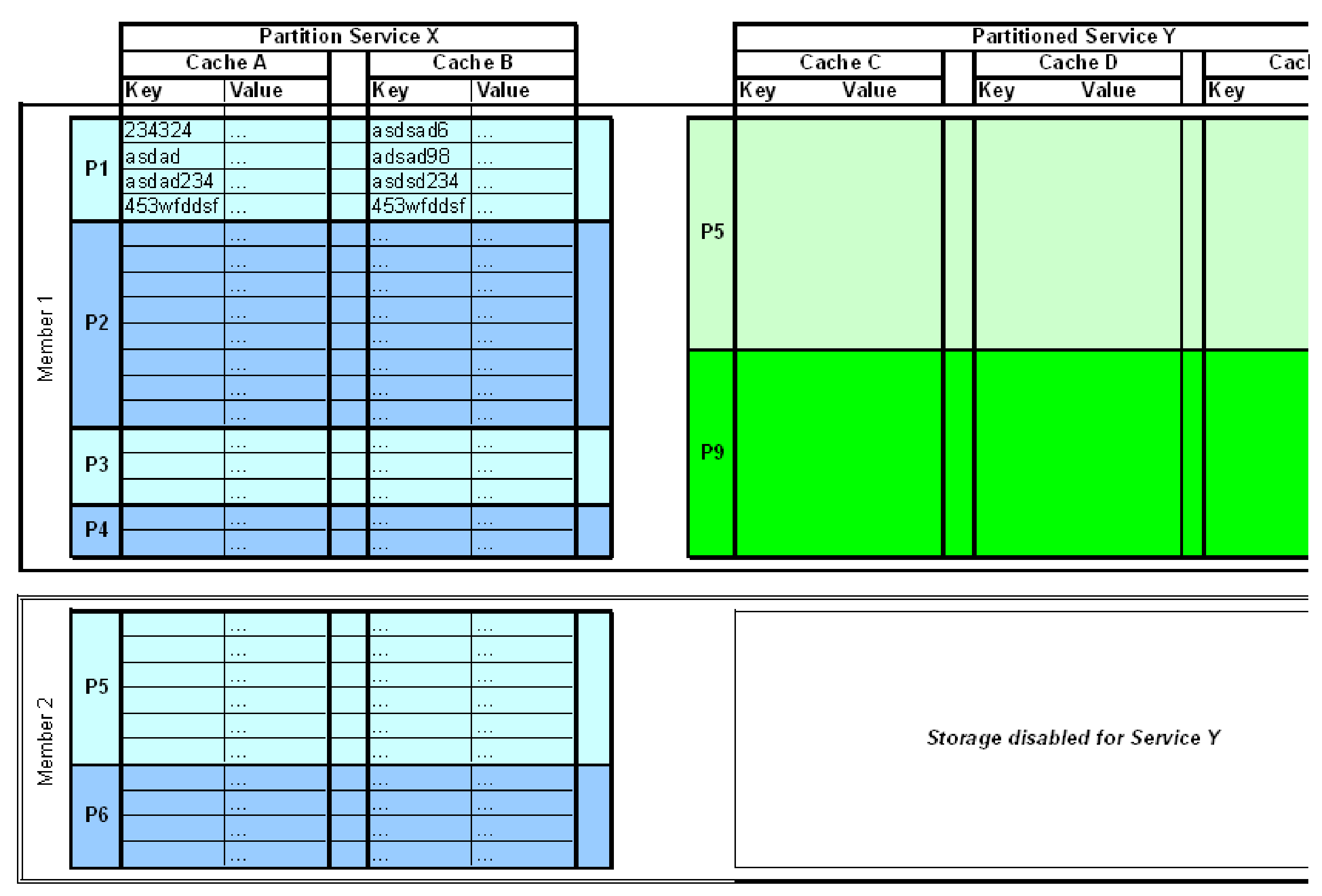
The Life of a Partition

Partition- and backing-map listeners

Partitioned services (distributed cache service)

- Algorithmically determined partition id
- Key affinity
- The partition has at most a single owner
- The partition moves atomically, and while it moves, it does not process operations

Distributed cache service



		Partition Service X			
		Cache A		Cache B	
		Key	Value	Key	Value
Member 1	P1	234324	...	asdsad6	...
		asdad	...	adsad98	...
		asdad234	...	asdsd234	...
		453wfddsf	...	453wfddsf	...
	P2		...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
	P3		...	...	...
			...	...	...
			...	...	...
	P4		...	...	...
			...	...	...
Member 2	P5		...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
			...	...	...
	P6		...	...	...
			...	...	...
			...	...	...
			...	...	...

	Partitioned Service Y				
	Cache C		Cache D		Cac
	Key	Value	Key	Value	Key
Member 1	P5				
	P9				
Member 2	*Storage disabled for Service Y*				

And what do we see of this from inside? – PartitionListener

- Starting with Coherence 3.3, you could register a PartitionListener

```
public interface PartitionListener {
  void onPartitionEvent(PartitionEvent evt);
}
```

- Notifications on the service thread when something happens with a partition

```
public class PartitionEvent extends
java.util.EventObject {
  PartitionSet m_setPartitions;
}
```

And what do we see of this from inside? – PARTITION_LOST event

- In Coherence 3.3 PARTITION_LOST event was delivered when you lost both the primary and all backups
  - Partition is now empty since its content was lost
  - Delivered on the storage senior

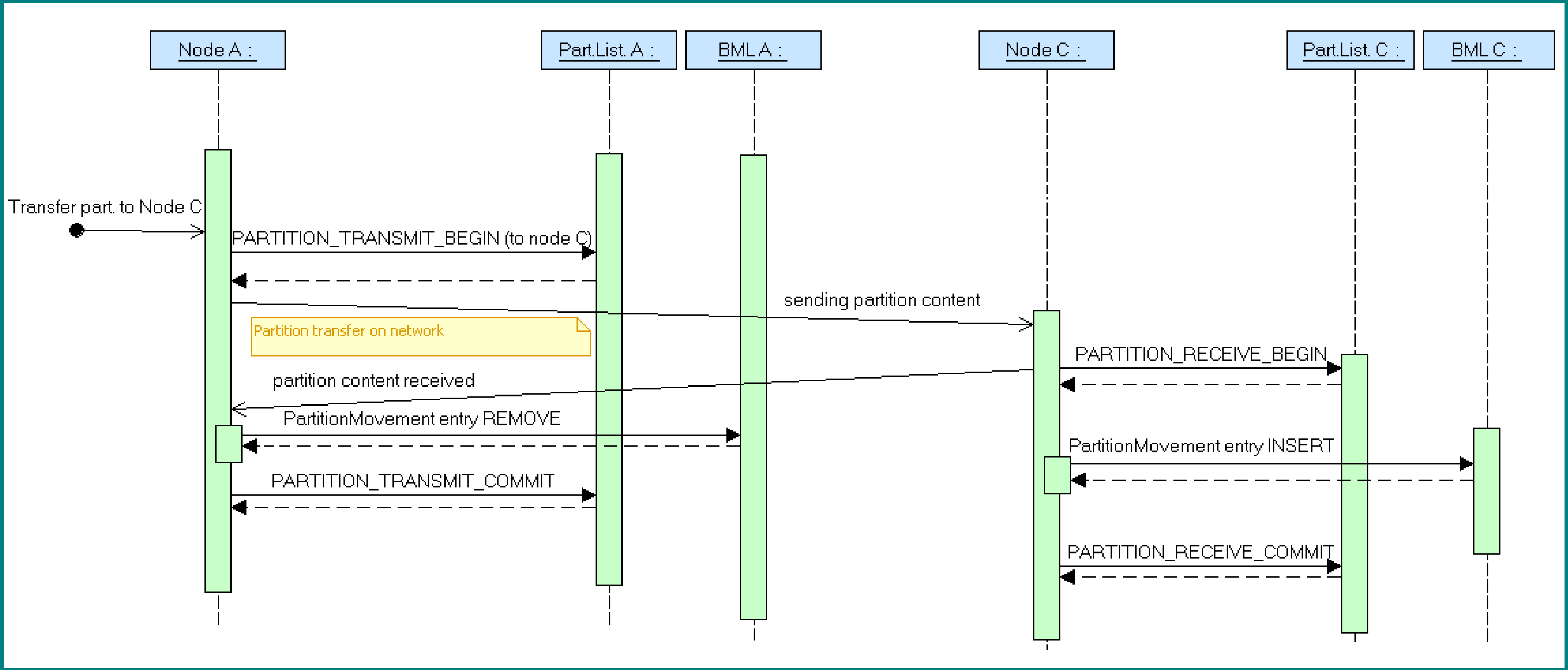
And what do we see of this from inside? – Coherence 3.5+

- PartitionEvents about the transfer of a partition
  - Old owner side:
    - PARTITION_TRANSMIT_BEGIN
    - PARTITION_TRANSMIT_ROLLBACK
    - PARTITION_TRANSMIT_COMMIT
  - New owner side:
    - PARTITION_RECEIVE_BEGIN
    - PARTITION_RECEIVE_COMMIT
  - The *memberFrom* and *memberTo* attributes
- Partitions are considered owned by the storage senior upon startup without notification

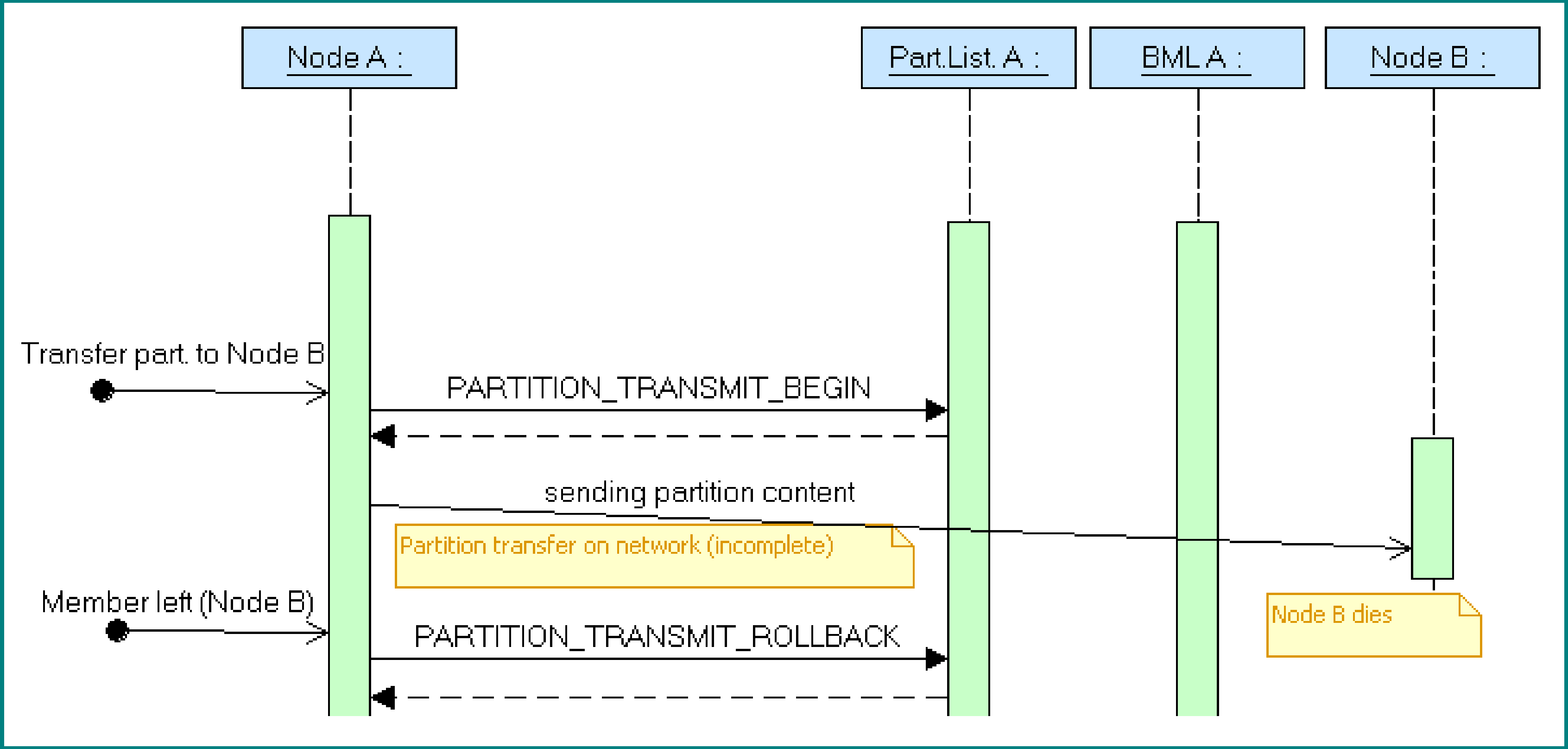
And what do we see of this from inside? – Coherence 3.6+

- PARTITION_ASSIGNED delivered on the storage senior
- As new nodes join, some of these partitions are transferred away based on advice from the PartitionAssignment-Strategy

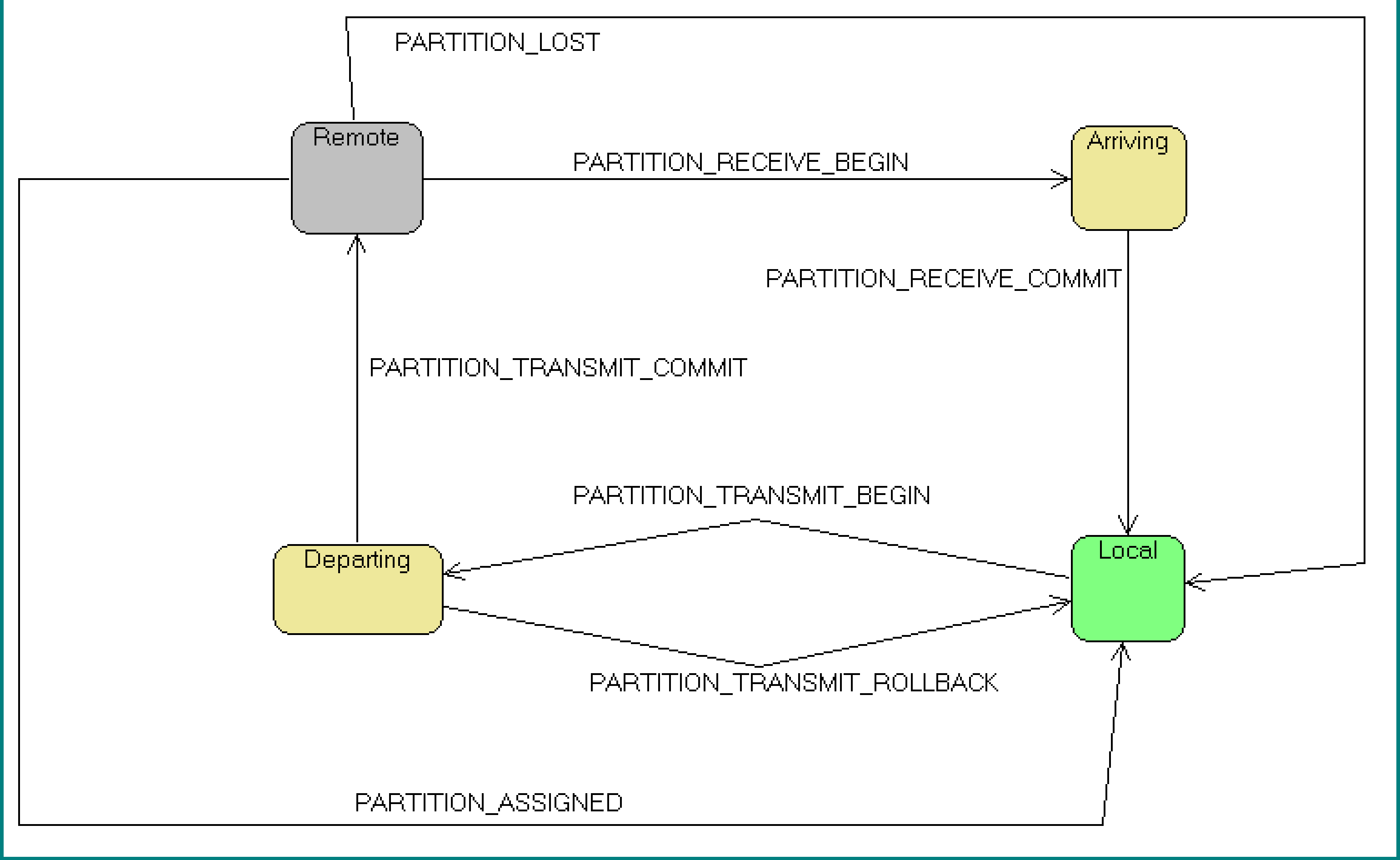
Partition transfer – success scenario


Node A :
Part.List. A :
BML A :
Node C :
Part.List. C :
BML C :
Transfer part. to Node C
PARTITION_TRANSMIT_BEGIN (to node C)
sending partition content
Partition transfer on network
PARTITION_RECEIVE_BEGIN
partition content received
PartitionMovement entry REMOVE
PartitionMovement entry INSERT
PARTITION_TRANSMIT_COMMIT
PARTITION_RECEIVE_COMMIT

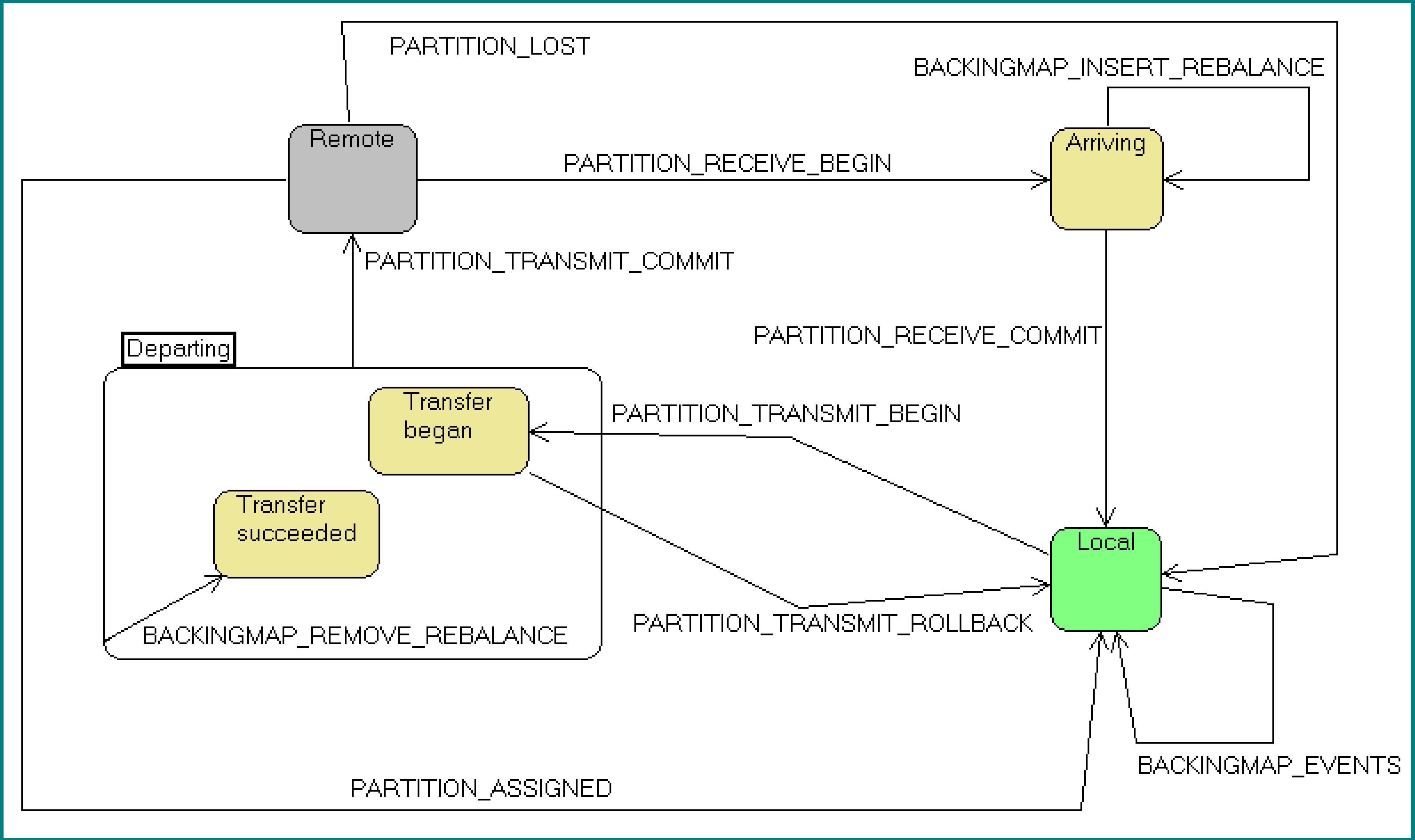
Partition transfer – failure scenario



State machine reacting to partition events



State machine reacting to partition and backing-map events



A Game of Partitions

PartitionAssignmentStrategies

Partition assignment strategies

- Configurable starting with Coherence 3.7
- Out-of-the-box
  - SimpleAssignmentStrategy
  - MirroringAssignmentStrategy

PartitionAssignmentStrategy

- Provides three methods
  - init(DistributionManager)
  - long analyzeDistribution()
    - Tells when to call next
  - getDescription()

DistributionManager

- Information about the current layout of partitions and state of the members (departing or not)
  - Member getMember(memberId)
  - PartitionSet getOwnedPartitions(memberId, storeId)
  - Set<Member> getOwnershipLeavingMembers()
  - Set<Member> getOwnershipMembers()
  - Ownership getPartitionOwnership(partitionId)
  - PartitionedService getService()
- Provides method for suggesting a partition transfer
  - suggest(PartitionSet, Ownership)

SimpleAssignmentStrategy

- Collects information about members and partitions:
  - Number of primaries and backups owned by the member
  - “Distance” of a member from the current owner of the partitions
  - Completely deterministic decisions
  - Caveat: ties are broken by comparing member ids

What can a partition assignment strategy be used for?

- Influence/override the out-of-the-box behaviour
  - Allows us to explicitly move partitions
    - Orchestrate rolling restart
  - Allows us to lay out partitions as we like
    - Recreate a stable layout saved before shutdown
  - Static partitioning
  - …

Wrapping DistributionManager and SimpleAssignmentStrategy

- Hide nodes which we don’t want the strategy to assign partitions to
- Make the original strategy believe that a node is departing
- Record transfers before letting the original strategy proceed.
  - If the recorded transfer leads to a balanced state, the strategy won’t override it

Orchestrated optimal rolling restart

- Start a new node and the strategy discovers it
- Show the old node as departing to the original strategy
- Explicitly transfer away partitions
- Once partitions are transferred, the now empty old node can depart
  - Caveat: MEMBERID_COMPARATOR

Reloading cache content dumped to local disk files

- Content of the caches dumped to files on local disk
- To read it back, partition layout must be the same
- Pre-initialize layout to the old distribution
  - Caveat: MEMBERID_COMPARATOR

Q&A

- Contact: robert@politext.info
- Web page: coherence.politext.info
  - This presentation
  - POF Serializer Generator (will be open-sourced soon)