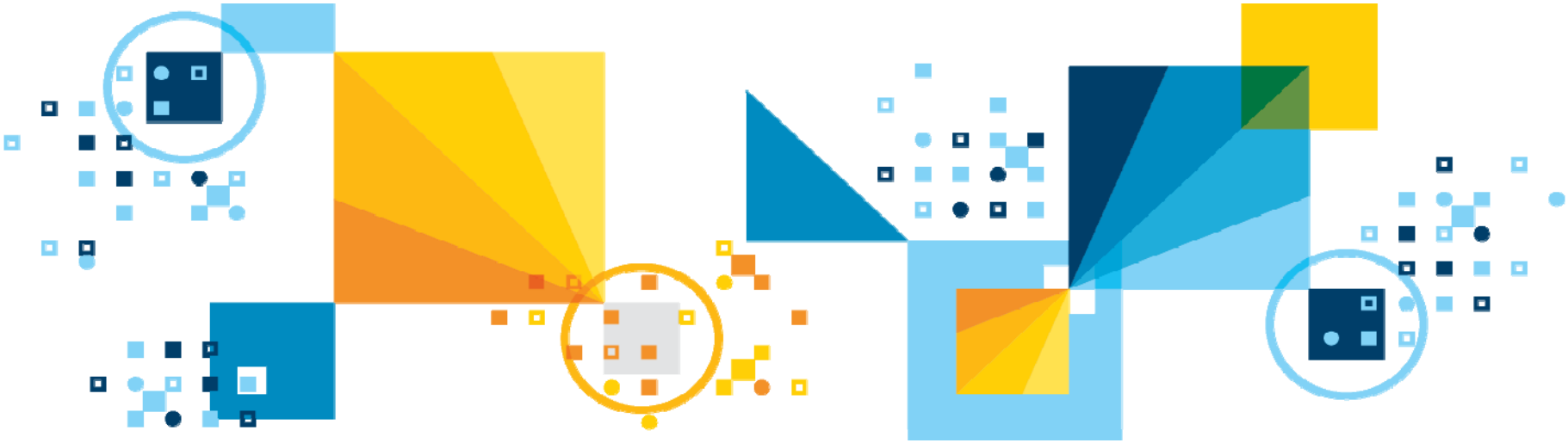
DB2 – High Availability and Disaster Recovery (HADR)

Module ID	10110
Length	1 hour



For questions about this presentation contact askdata@ca.ibm.com

January 22, 2015

Disclaimer

Module Information

- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
  - DB2 Fundamentals

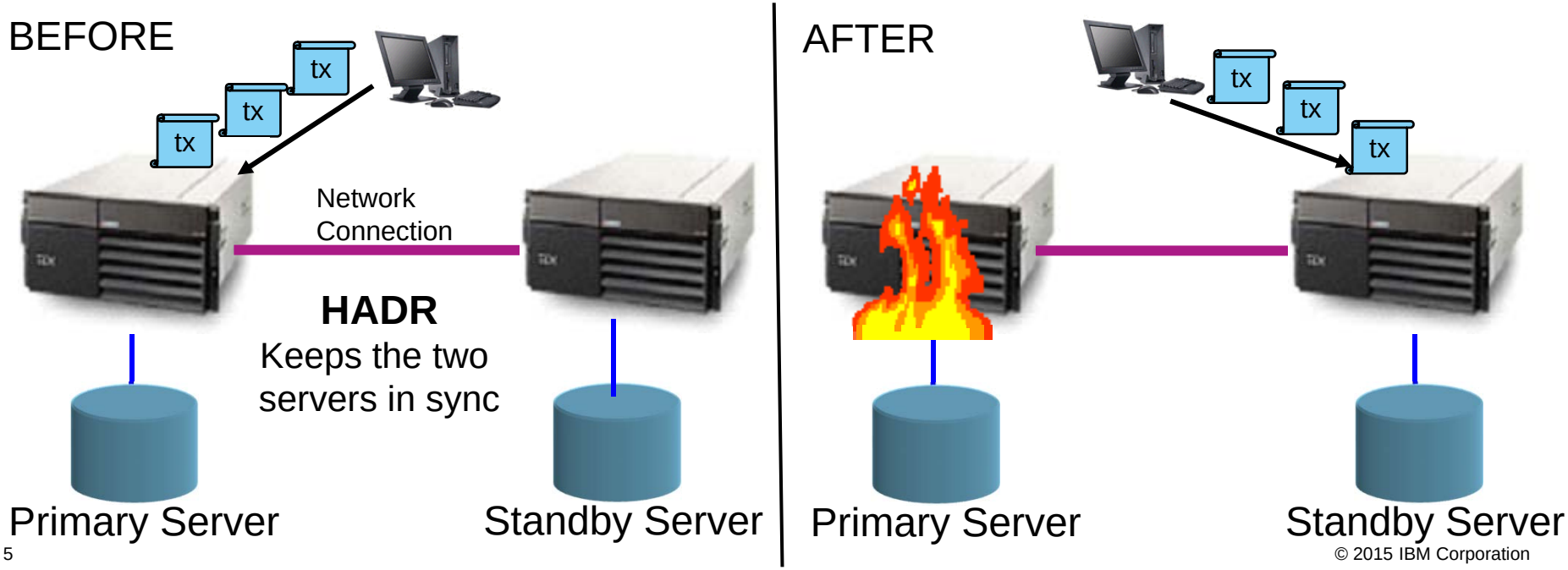
- After completing this module, you should be able to:
  - Explain the concepts of High Availability and Disaster Recovery
  - Describe what standby databases are, and how to fail over to another database

Module Content

- High Availability and Disaster Recovery (HADR): Concepts and Capabilities
- Requirements
- Configuration
- Reads on Standby
- Takeover Operation
- Multiple-Standby Overview

HADR Concept

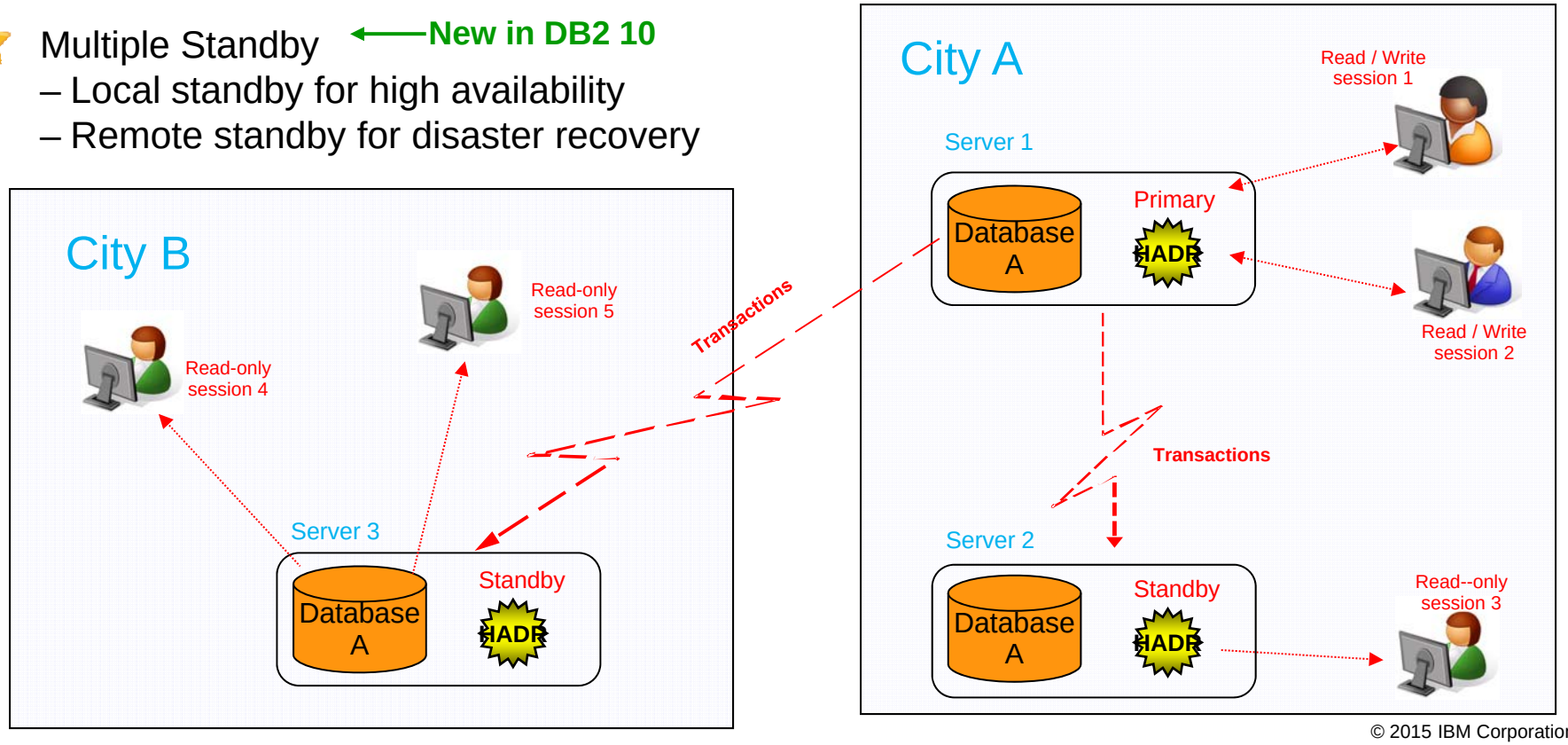
- HADR is a **database replication feature** that provides a high availability solution for both partial and complete site failures.
- HADR is based on a pair (primary and standby) of databases. In case of failure, the standby database can take over the workload
  - **Primary**
    - Handles all client connections and processes transactions
    - Continuously ships transaction logs to the standby over the network
  - **Standby**
    - Kept in sync with primary by applying received transaction logs



HADR Capabilities

- High Availability
  - Automatic Failover with Tivoli System Automation
  - Automatic Client Reroute
- Reads-on Standby
  - Offloading read transactions from the primary server to standby servers

Multiple Standby ⟵ **New in DB2 10**
  - Local standby for high availability
  - Remote standby for disaster recovery

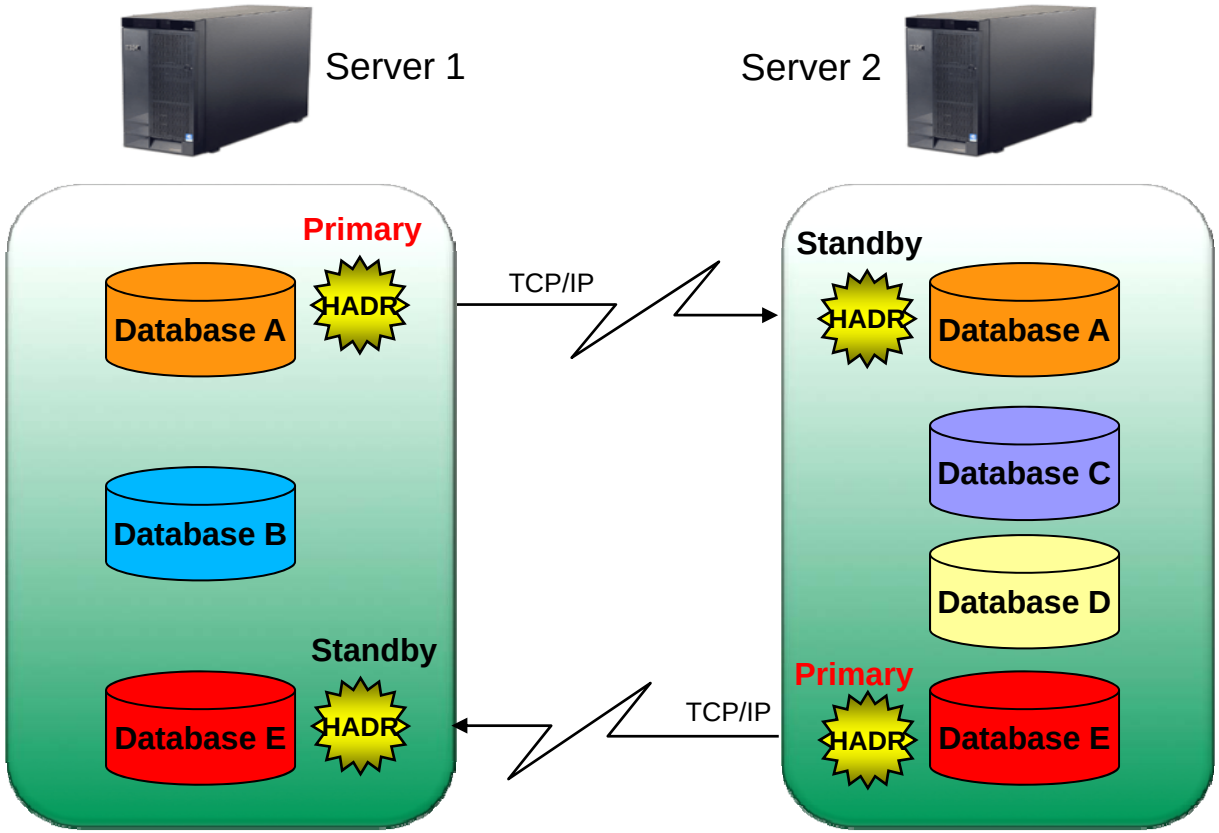


Benefits of HADR

- **Minimizes impact** of planned and unplanned outage
  - Eg: Software upgrades without interruption
- Client applications can **easily recover** from disaster **reducing data loss** and **downtime**
  - No rewrite of your application
  - Ultra fast and transparent failover
  - Easy integration with high availability clustering software
- **Negligible impact on performance**
- **Easy administration and setup**
  - No specialized hardware required
  - Setup in only minutes with graphical wizard

HADR Scope

- HADR replication takes place at database level



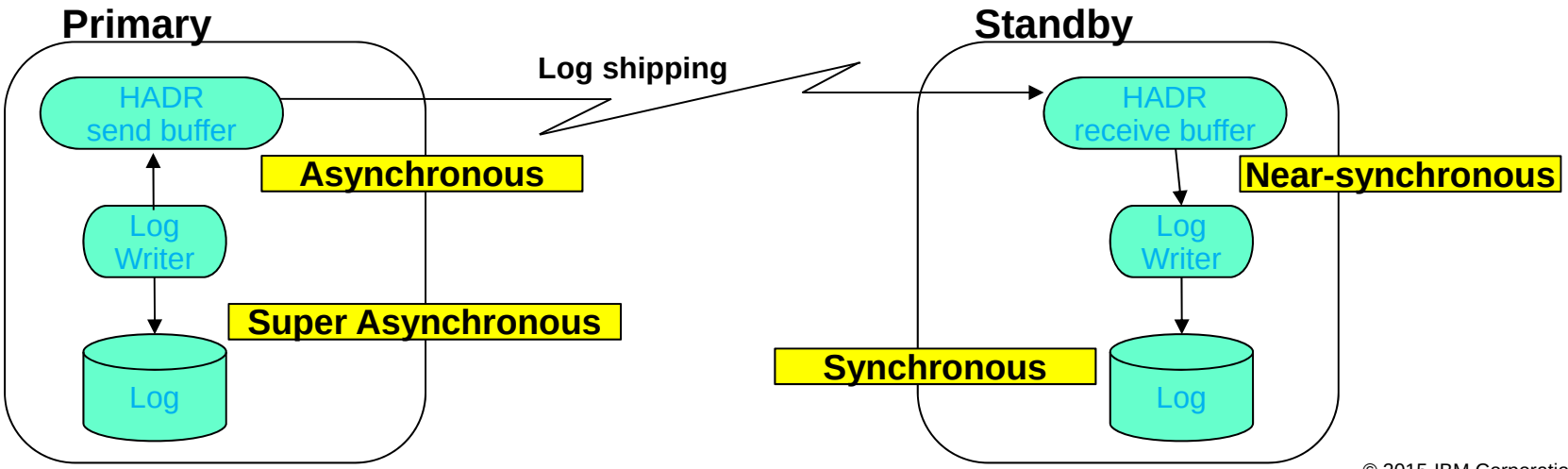
What Gets Replicated?

- Operations replicated under HADR:
  - DDL and DML statements
  - Buffer pool and table space operations, as well as storage-related operations performed on automatic storage databases (ALTER DATABASE)
  - Online/Offline reorganization
  - Changes to metadata (system catalog information) for stored procedure and user-defined functions (UDFs)
  - Load operations with **COPY YES** option

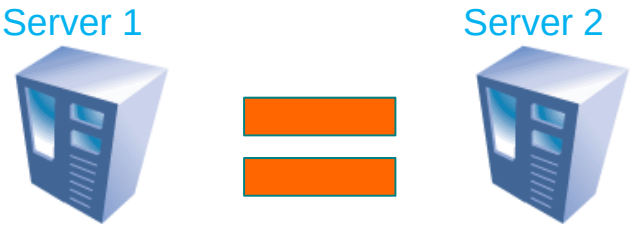
- Operations **not** replicated under HADR:
  - Database configuration parameters
  - Tables created with the **NOT LOGGED INITIALLY** option
  - **Non-logged LOB** columns
  - UDF libraries
  - Index pages (unless **LOGINDEXBUILD** is enabled)

HADR Synchronization Modes

- Synchronous
  - Greatest protection against transaction loss
  - A higher cost of transaction response time
- Near Synchronous (**default**)
  - Better performance than Synchronous mode
  - no waiting for the log buffer to be written to disk on standby
- Asynchronous
  - If latency of the network between the two servers is too great
- Super Asynchronous
  - Better response time during network interruptions or congestion



HADR Requirements



- Both primary and standby systems must have similar hardware and software configuration
- Hardware
  - **Server**: same vendor and architecture
  - **Memory**: identical amounts of memory to support replayed buffer pool operations
  - **Network**: a TCP/IP network between the HADR systems, connected over a high-speed, high-capacity network
    - **Recommendation**: an extra network to separate internal HADR traffic from workload / client traffic
- Software
  - **OS**: identical version including patch level
  - **DB2**: identical bit size (32 or 64) and version
    - Supports rolling upgrades
- Database Configuration
  - Not supported in a partitioned database environment
  - Primary and standby databases must have the same database name
  - Table spaces must be identical on the primary and standby databases

HADR Configuration Parameters Explained

- hadr_db_role (**READ-ONLY**)
  - This parameter indicates the current role of a database
- hadr_<local/remote>_host
  - specifies the local/remote host for HADR TCP communication
- hadr_<local/remote>_svc
  - specifies the TCP service name/ port number for which the local/remote HADR process accepts connections
- hadr_peer_window
  - Specifies the amount of time that HADR primary-standby database pair continues to behave as though still in peer state, if the primary database loses connection with the standby database.
- hadr_remote_inst
  - specifies the instance name of the remote server.
- hadr_timeout
  - specifies the time (in secs) that the HADR process waits before considering a communication attempt to have failed

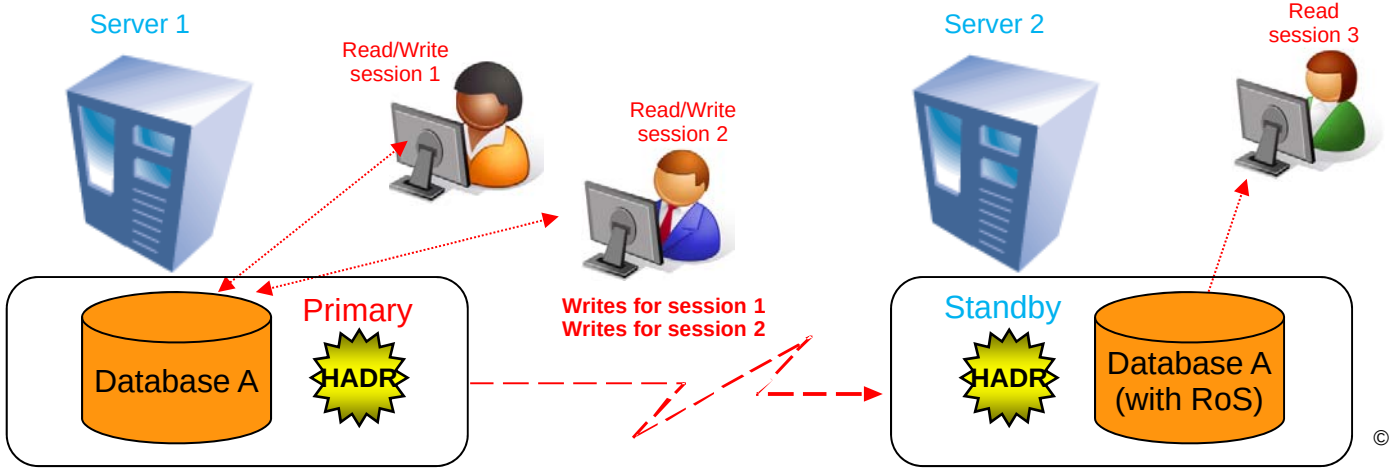
    Recommend: at least 10 seconds.
  - **Too long**
    - In peer state, if standby is not responding, transactions on primary can hang for HADR_TIMEOUT period.
  - **Too short**
    - You get false alarm on the connection.
    - Frequent disconnection and reconnection wastes resource.
    - HA protection also suffer as disconnection brings primary out of peer state.

Basic HADR Configuration

Primary Setup	Standby Setup
1. db2 backup db hadr_db to backup_dir	**2. db2 restore db hadr_db from backup_dir**
3. db2 update db cfg for hadr_db using HADR_LOCAL_HOST host_a HADR_REMOTE_HOST host_b HADR_LOCAL_SVC svc_a HADR_REMOTE_SVC svc_b HADR_REMOTE_INST inst_b HADR_TIMEOUT 120 HADR_SYNCMODE ASYNC	**4. db2 update db cfg for hadr_db using** HADR_LOCAL_HOST host_b HADR_REMOTE_HOST host_a HADR_LOCAL_SVC svc_b HADR_REMOTE_SVC svc_a HADR_REMOTE_INST inst_a HADR_TIMEOUT 120 HADR_SYNCMODE ASYNC
6. db2 start hadr on database hadr_db as primary	**5. db2 start hadr on database hadr_db as secondary**

Configuring Reads on Standby (RoS)

- Perform read-only operations on HADR standby database
  - Run concurrent read-only workloads offloading reporting, DSS/BI workloads to standby with minimal impact
  - Enabled by **DB2_HADR_ROS** registry variable
- During failover, DB2 seamlessly turns the read-on-standby to a primary read/write server
- Restrictions
  - Only **Uncommitted Read (UR)** isolation level is supported on the standby
  - Workload manager DDL statements on the primary are replayed on the standby, but they will not be effective on the standby
  - Some DB2 features and objects are not supported
    - CGTTs, DGTTs, not logged initially tables,not-inlined LOBs and XML, LONG VARCHAR, LONG GRAPHIC, STMM



HADR Configuration – HADR Delayed Replay

New in DB2 10

- **HADR_REPLAY_DELAY** – DB2 Configuration parameter that controls how far behind (in seconds) the standby will intentionally remain at all times
  - Max value is 2147483647 seconds
  - Very useful to prevent data loss due to errant transactions.
  - Can only be set on a standby database
- Example:
  - You can delay the HADR replication by 24h, creating a safe point-of-restore for unwanted transactions.
- A **TAKEOVER** command on a standby with replay delay enabled will fail
  - You must first set the hadr_replay_delay configuration parameter to 0 and then deactivate and reactivate the standby to pick up the new value, and then issue the TAKEOVER command
- The delayed replay feature is supported only in SUPERASYNC mode

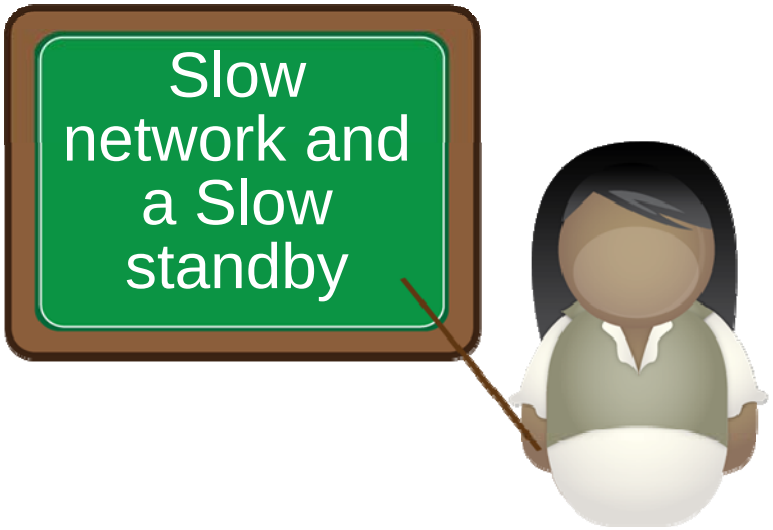
HADR Fine Tuning – Standby Log Spooling

New in DB2 10

- The log spooling feature allow the standby to spool log records arriving from the primary
  - This decouples log replay on the standby from receiving of the log data from the primary
  - Logs will be spooled in the standby DB's active log path
- **Reduces impact on primary database caused by slow log replay on standby.**
  - E.g.: When intensive operations like reorgs are replayed on the standby.
- Set through a new DB CFG parameter: **HADR_SPOOL_LIMIT**
- Can limit the amount of space allocated to the log spool by specifying the maximum amount of disk space to use
  - Value of 0 disables spooling (default)
  - Value of -1 defines the spool to be unlimited (limited by file system free space)
- Log files will be automatically deleted when they are no longer required
- Log spooling does not cause data loss. The data shipped from the primary is still replicated to the standby using the specified synchronization mode.
- Takeover may take longer as the spool must be drained before takeover completes

SUPERASYNC or Log Spooling?

- Superasync mode
  - Eliminates back pressure on the primary
  - Useful network is slow
- Log spooling
  - Allows receipt of log records regardless of log replay on the standby
    - Useful if standby has limited resources
    - Can be used with any synchronization mode
- Both features can be combined



HADR Takeover Operation

- Normal Takeover
  - 1. Standby tells primary that it is taking over
  - 2. Primary forces off all client connections and refuses new connections
  - 3. Primary rolls back any open transactions and ships remaining log, up to the end of log, to standby
  - 4. Standby replays received log, up to end of the log
  - 5. Primary becomes new standby, standby becomes new primary
  - 7. ACR will automatically reroute client applications to new primary

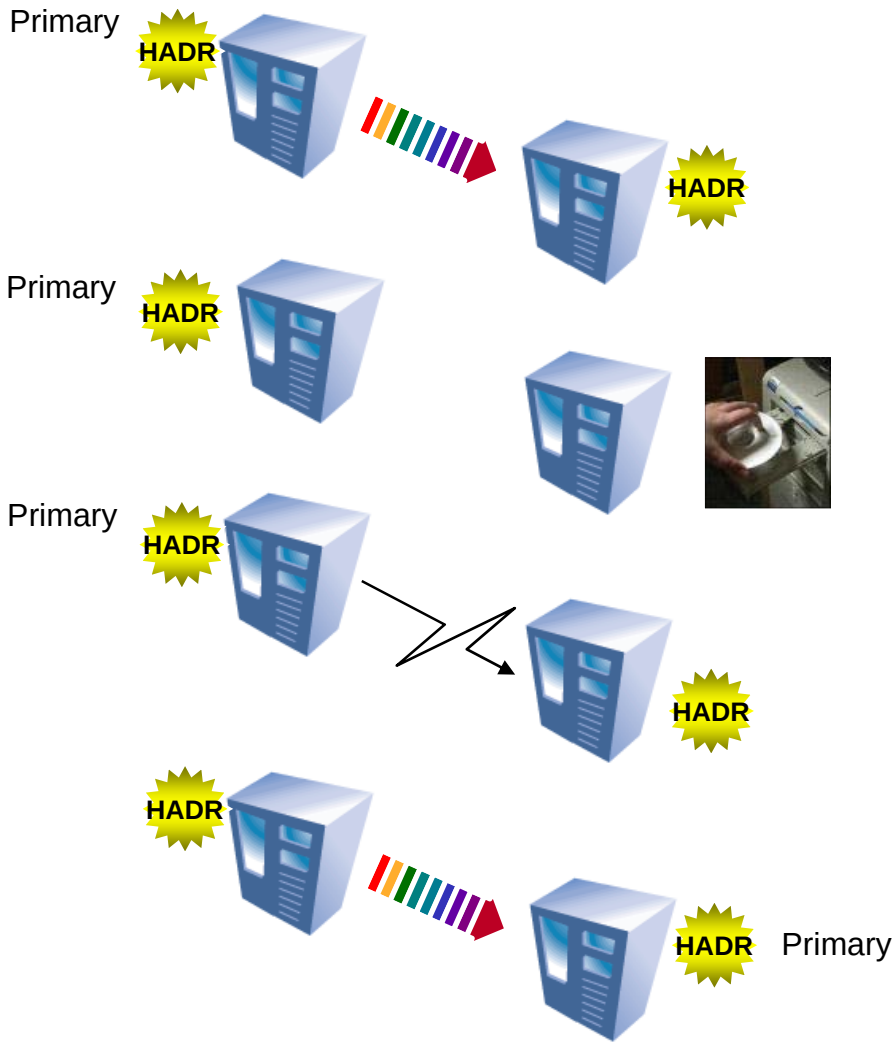
```
TAKEOVER HADR ON DATABASE <dbname>
```

- By Force
  - 1. Standby sends a notice asking the primary to shut itself down
  - 2. Standby does NOT wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down (must be within peer window)
  - 3. Standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

```
TAKEOVER HADR ON DATABASE <dbname> BY FORCE
```

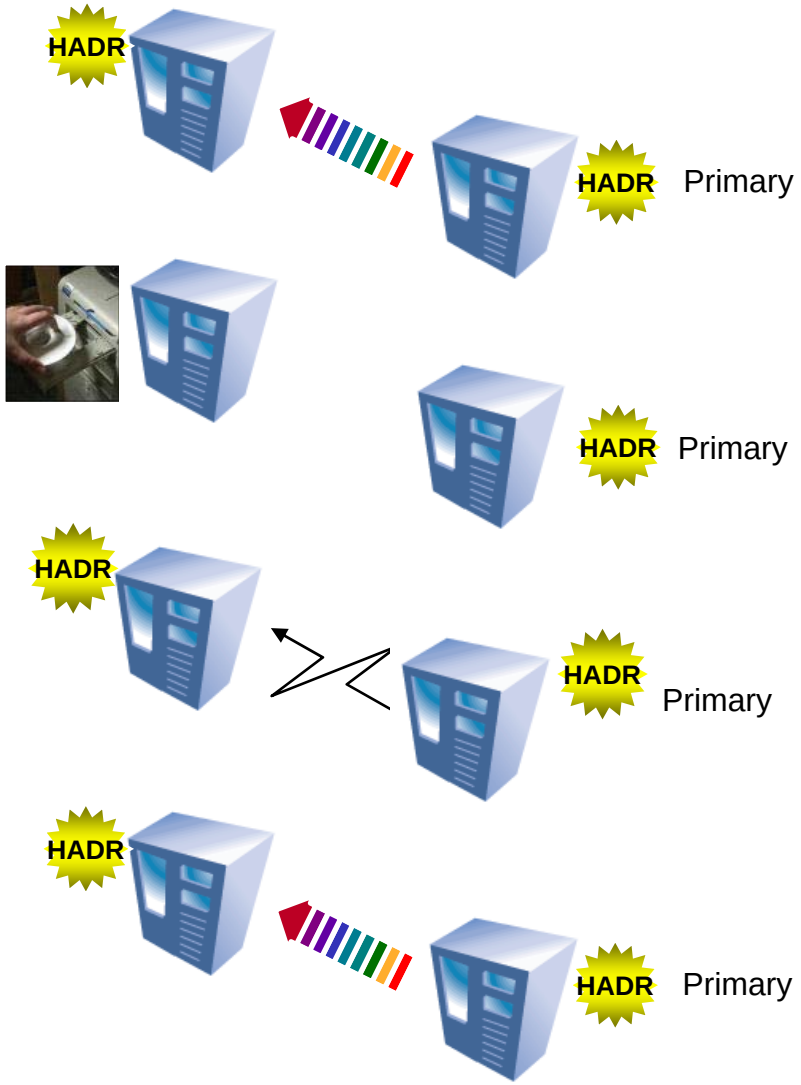
Normal Takeover Example

- HADR in peer state
  - Deactivate HADR on the Standby
- Upgrade the standby
- Start the standby again
  - Let it catch-up with primary
- Issue a normal TAKEOVER
  - The primary and standby change roles



Normal Takeover Example

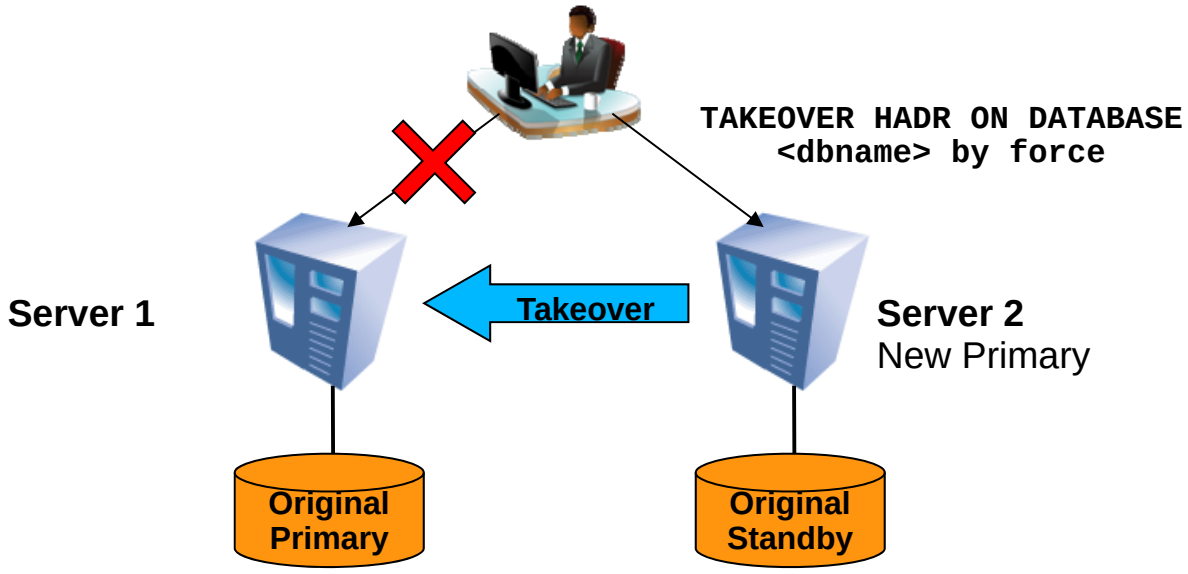
- Suspend the new standby
- Upgrade the new standby
- Reactivate the new standby
  - Let it catch-up with primary
- Optionally, TAKEOVER again
  - The primary and standby play their original roles



By Force Takeover Example

1) Standby sends a notice asking the primary to shut itself down
2) Standby does NOT wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down (must be within peer window)
3) Standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

Note: The option “by force peer window only” only succeeds within the peer window



Automatic Failover with Cluster Manager

- Provides quorum between the primary and standby nodes to prevent split-brain scenario
- SA MP is the default cluster manager in an IBM DB2 server clustered environment on AIX, Linux, and Solaris SPARC operating systems
- Other cluster management software supported are :
  - AIX: High Availability Cluster Multi-Processing (HACMP)
  - Linux: Tivoli System Automation
  - Microsoft: Microsoft Cluster Server
  - Solaris: Sun Cluster or VERITAS Cluster Server
  - Hewlett-Packard: Multi-Computer/ServiceGuard



Tivoli System Automation for Multi-platforms

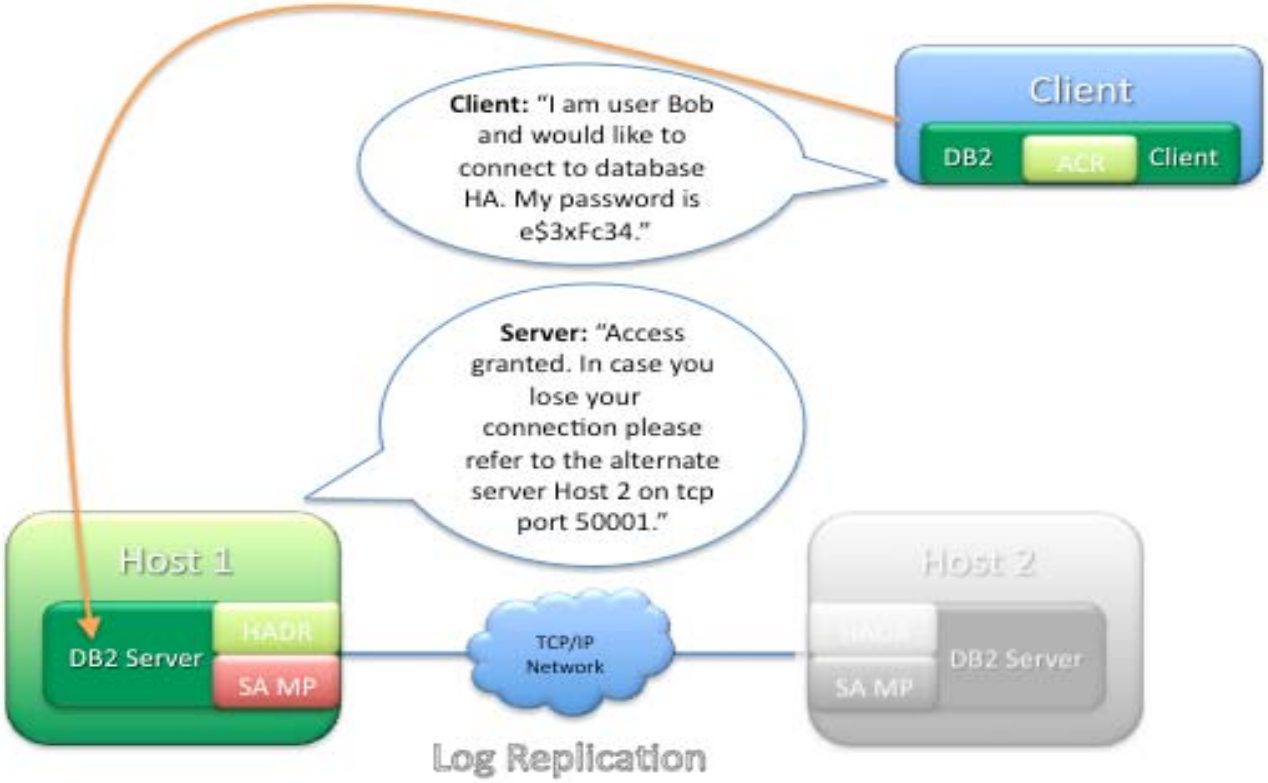
- DB2 is integrated with Tivoli System Automation for Multi platforms (SA MP) for node failure detection and automatic failover
- Easily set up using the db2haicu wizard
- Supported Operating Systems for SA MP
  - AIX 5.3 and 6.1
  - Linux
    - Red Hat RHEL 5 and 6
    - SUSE SLES 10 & 11
    - Windows 2003 and 2008 Standard / Enterprise Edition(32 & 64 bit)
    - Microsoft Cluster Services can be used for HADR failover
  - Solaris 10 on SPARC (64-bit)

Automatic Client Reroute (ACR)

- Redirects client applications from primary database to standby database immediately after a failover operation is performed
- Transparent to the application
- Applications can recover from a loss of communications with minimal interruption
- Configuration:
  - Set up HADR using the **Set Up HADR Databases Wizard**:
    - Automatic Client Reroute is enabled by **default**
  - Set up HADR manually:
    - Enable the Automatic Client Reroute feature by executing the **UPDATE ALTERNATE SERVER FOR DATABASE** command

Automatic Client Reroute (ACR)

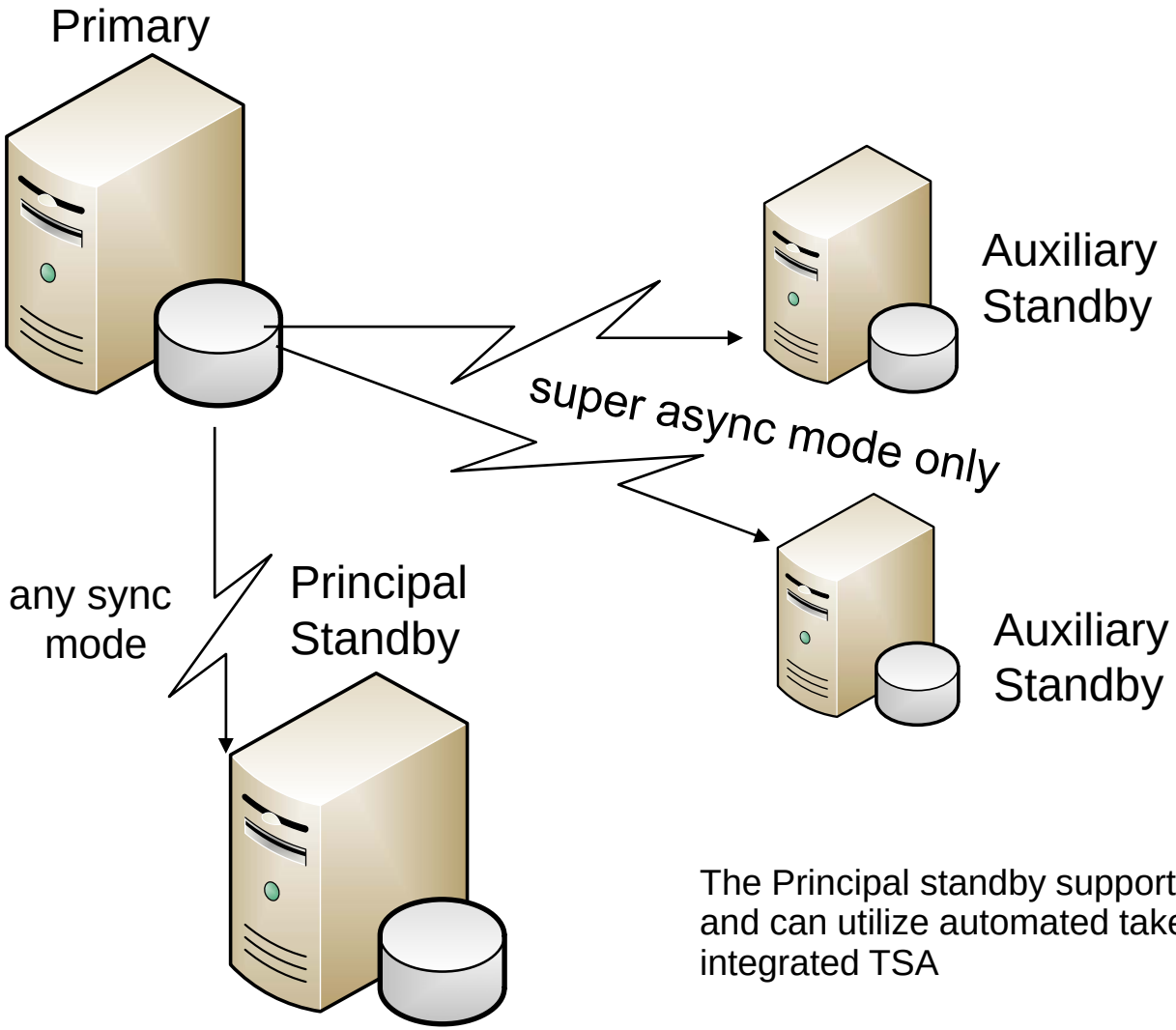
- Reroute is transparent to the application
  - Automatically reconnects to the alternate server
- Application must be able to catch the exception
  - SQL30108 “A connection failed but has been re-established. …”
- Also works with normal databases or with DPF, SQL Replication, etc.



Configuring a Clustered Environment using db2haicu

- DB2 High Availability Instance Configuration Utility (db2haicu)
- Text based utility to configure and administer your highly available databases in a clustered environment
- db2haicu collects information about database instance, cluster environment, and cluster manager by querying your system
- More information is supplied through interactive mode or through XML input file

Multiple Standby Overview



Support for up to two(2) Auxiliary Standbys (AS)
AS supports super async mode only
No automated takeover supported
Always feed from the current primary
Can be added dynamically

The Principal standby supports any sync mode and can utilize automated takeover with the integrated TSA

Multiple Standby Considerations

- You can have a maximum of **three** standby databases:
  - One principal standby
  - Two auxiliary standbys
- Only the principal standby supports all the HADR synchronization modes
  - All auxiliary standbys will be in **SUPERASYNC** mode
- IBM Tivoli System Automation for Multi platforms (SA MP) support applies only between the primary HADR database and its principal standby
- You must set the new **hadr_target_list** database configuration parameter on all HADR databases to enable the multiple standby mode.
- **HADR_TARGET_LIST**
  - Specifies a list of up to three target host:port pairs that act as HADR standby databases.
  - For a primary server, it determines which hosts act as standbys for that primary
  - For a standby server, it identifies the standby hosts to be used if this standby takes over as the new HADR primary database

Automatic Reconfiguration

- Reconfiguration after HADR service is started
  - Values for **hadr_remote_host, hadr_remote_inst,** and **hadr_remote_svc** configuration parameters are automatically reset when HADR starts if you did not correctly set them.

- Reconfiguration during and after a takeover
  - Values for the **hadr_remote_host, hadr_remote_inst,** and **hadr_remote_svc** configuration parameters are updated automatically on all the databases that are part of the new setup.
  - Any database that is not a valid standby for the new primary is not updated.

Takeover Behavior in Multiple Standby Environment

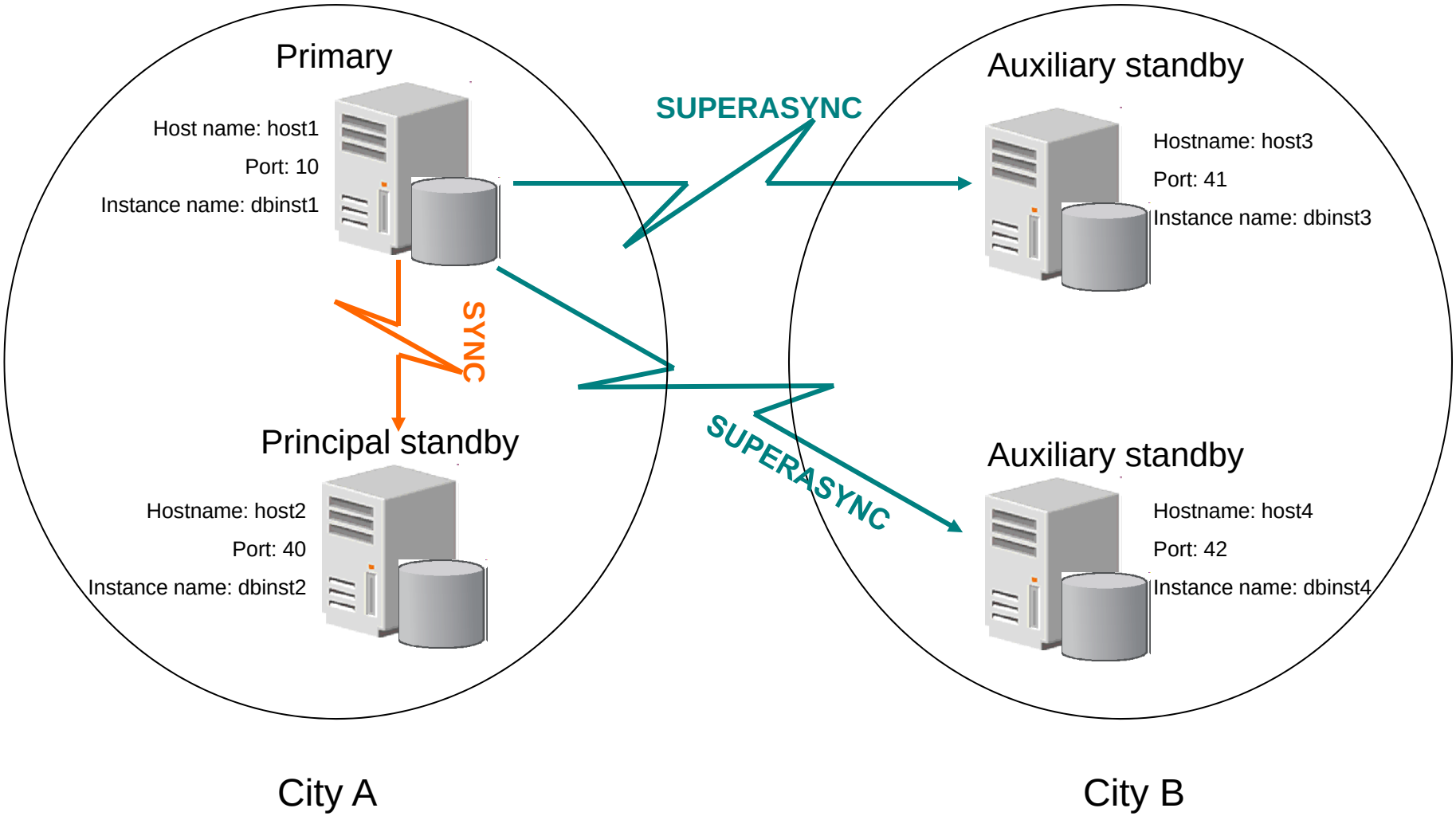
- Two types of takeovers as in traditional HADR
  - Role Switch (non-forced takeover)
    - Can be performed only when the primary is available and it switches the role of primary and standby. Provides zero data loss guarantee
  - Failover
    - Can be performed when the primary is not available. It is commonly used in primary failure cases to make the standby the new primary.

- Both types of takeover are supported in multiple standby mode, and any of the standby databases can take over as the primary.

If a standby is not included in the new primary's target list, it is considered to be orphaned and cannot connect to the new primary.

- After a takeover, DB2 auto-redirects and makes necessary configuration changes (host/service/instance name of new primary) on the standbys that are in the new primary's target list (and vice versa)

HADR Multiple Standby Example



Configuration of Each Host

Configuration parameter	Host1	Host2	Host3	Host4
Hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host1:10 | host2:40 | host4:42	host1:10 | host2:40 | host3:41
Hadr_remote_host	host2	host1	host1	host1
Hadr_remote_svc	40	10	10	10
Hadr_remote_inst	dbinst2	dbinst1	dbinst1	dbinst1
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	superasync	superasync
Effective Hadr_syncmode	N/A	sync	superasync	superasync

Initial Setup

- Step 1: Create a backup image
  - On host1 (primary)

```
DB2 BACKUP DB HADR_DB to /nfs/db_backup
```

- Step 2: Initialize the standbys
  - On each of host2, host3, and host 4

```
DB2 RESTORE DB HADR_DB from /nfs/db_backup
```

- Step 3: Update the configuration
  - the **HADR_TARGET_LIST** should be set to the following on all nodes:

```
principalHost:Principalsvc | auxhost1:auxsvc1 |  auxhost2:auxsvc2
```

- The following are not required to be set in a multiple standby environment as they will be automatically set
  - hadr_remote_host
  - hadr_remote_svc
  - hadr_remote_inst

Initial Setup

- On host1 (the primary)

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST
host2:40|host3:41|host4:42
HADR_REMOTE_HOST host2
HADR_REMOTE_SVC 40
HADR_LOCAL_HOST host1
HADR_LOCAL_SVC 10
HADR_SYNCMODE sync
HADR_REMOTE_INST db2inst2"
```

- On host2 (principal standby)

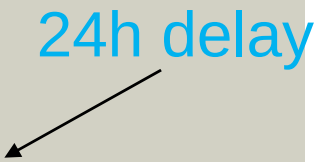
```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST
host1:10|host3:41|host4:42
HADR_REMOTE_HOST host1
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host2
HADR_LOCAL_SVC 40
HADR_SYNCMODE sync
HADR_REMOTE_INST db2inst1"
```

- On host3 (auxiliary standby)

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST
host1:10|host2:40|host4:42
HADR_REMOTE_HOST host1
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host3
HADR_LOCAL_SVC 41
HADR_SYNCMODE superasync
HADR_REMOTE_INST db2inst1“
```

- On host4 (auxiliary standby)

```
DB2 "UPDATE DB CFG FOR hadr_db USING
HADR_TARGET_LIST
host1:10|host2:40|host3:41
HADR_REMOTE_HOST host1
HADR_REMOTE_SVC 10
HADR_LOCAL_HOST host4
HADR_LOCAL_SVC 42
HADR_SYNCMODE superasync
HADR_REMOTE_INST db2inst1
HADR_REPLAY_DELAY 86400"
```



Starting HADR

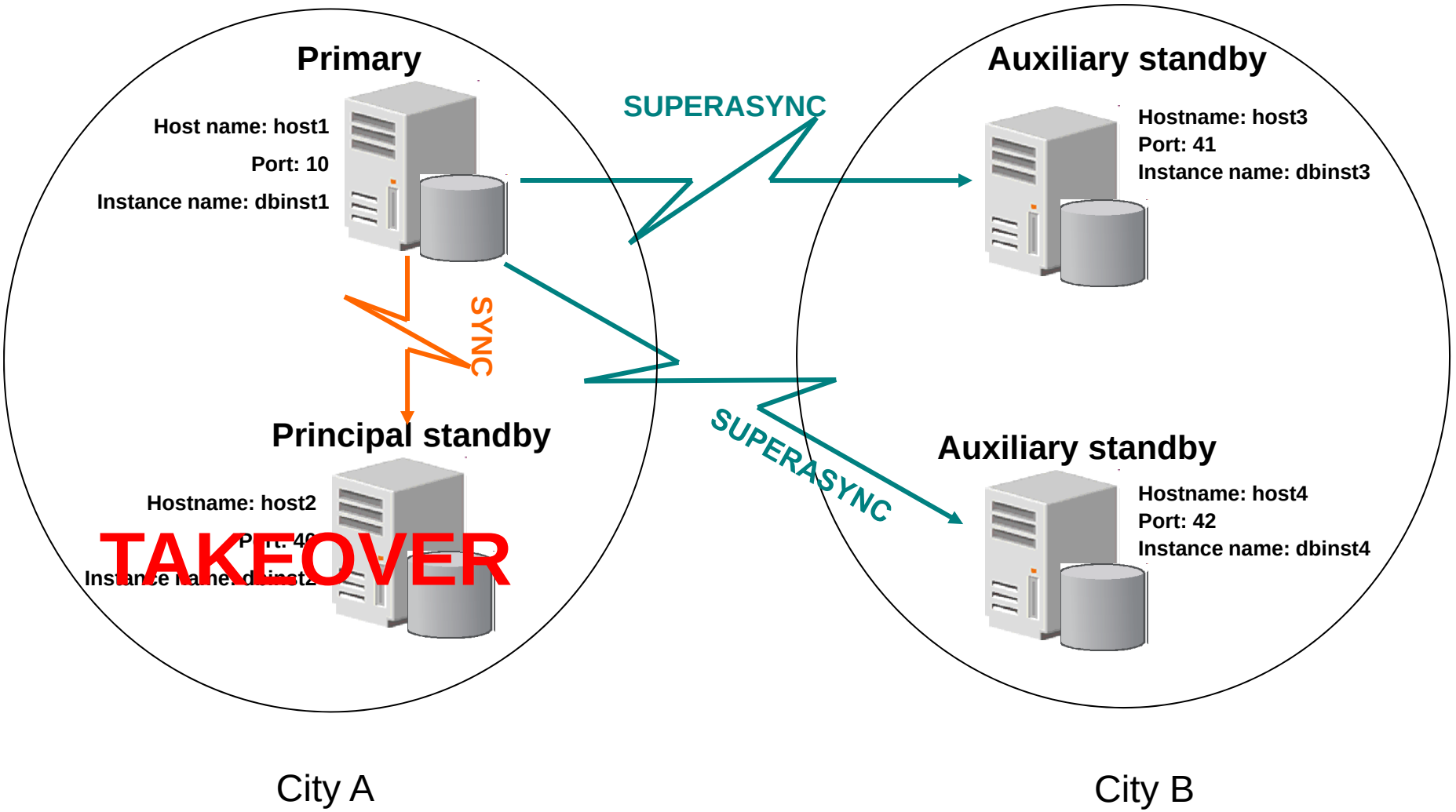
- The DBA starts the standby databases first, by issuing the following command on each of host2, host3, and host 4

```
DB2 START HADR ON DB hadr_db AS STANDBY
```

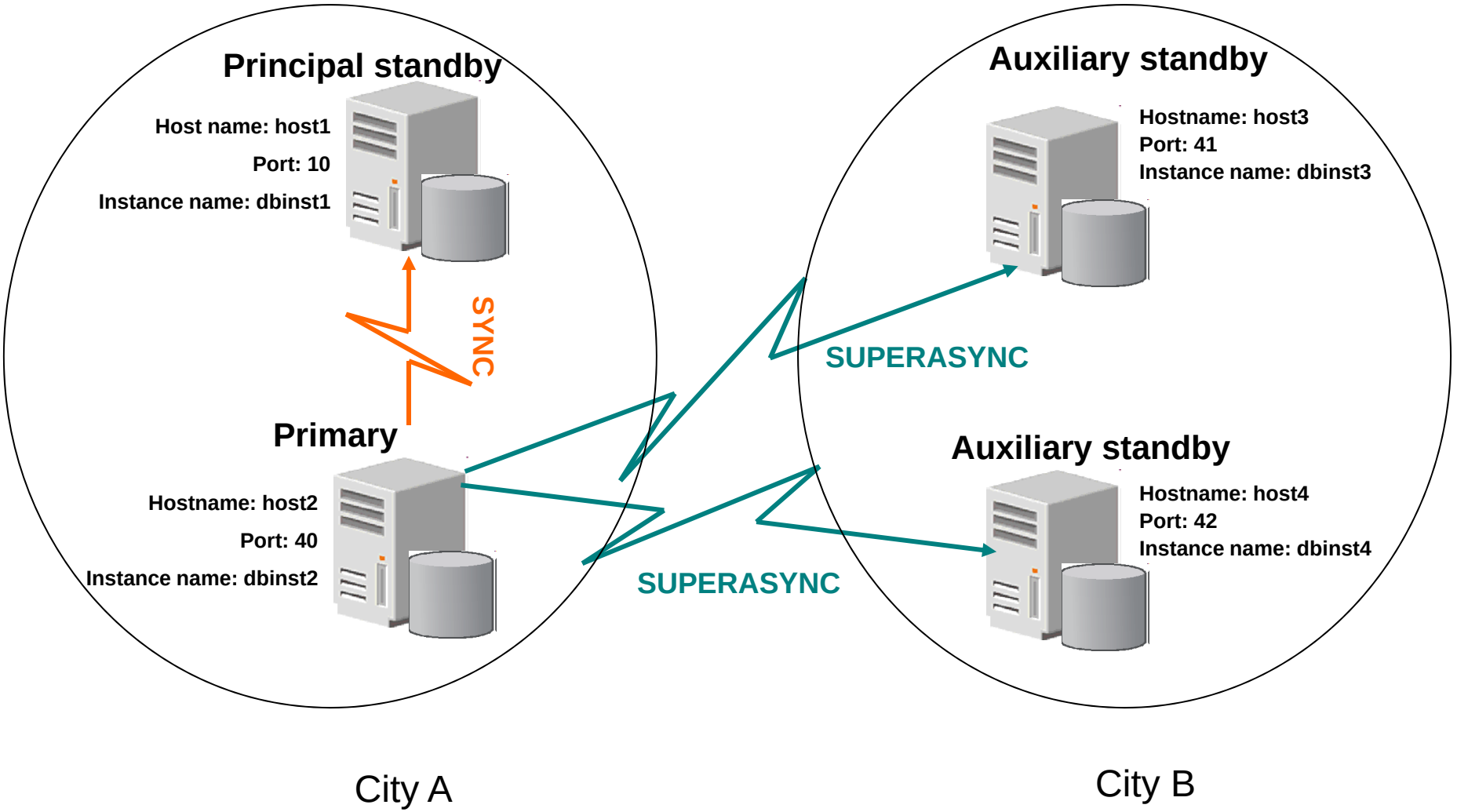
- Next, the DBA starts HADR on the primary database, on host1

```
DB2 START HADR ON DB hadr_db AS PRIMARY
```

HADR Multiple Standby Example



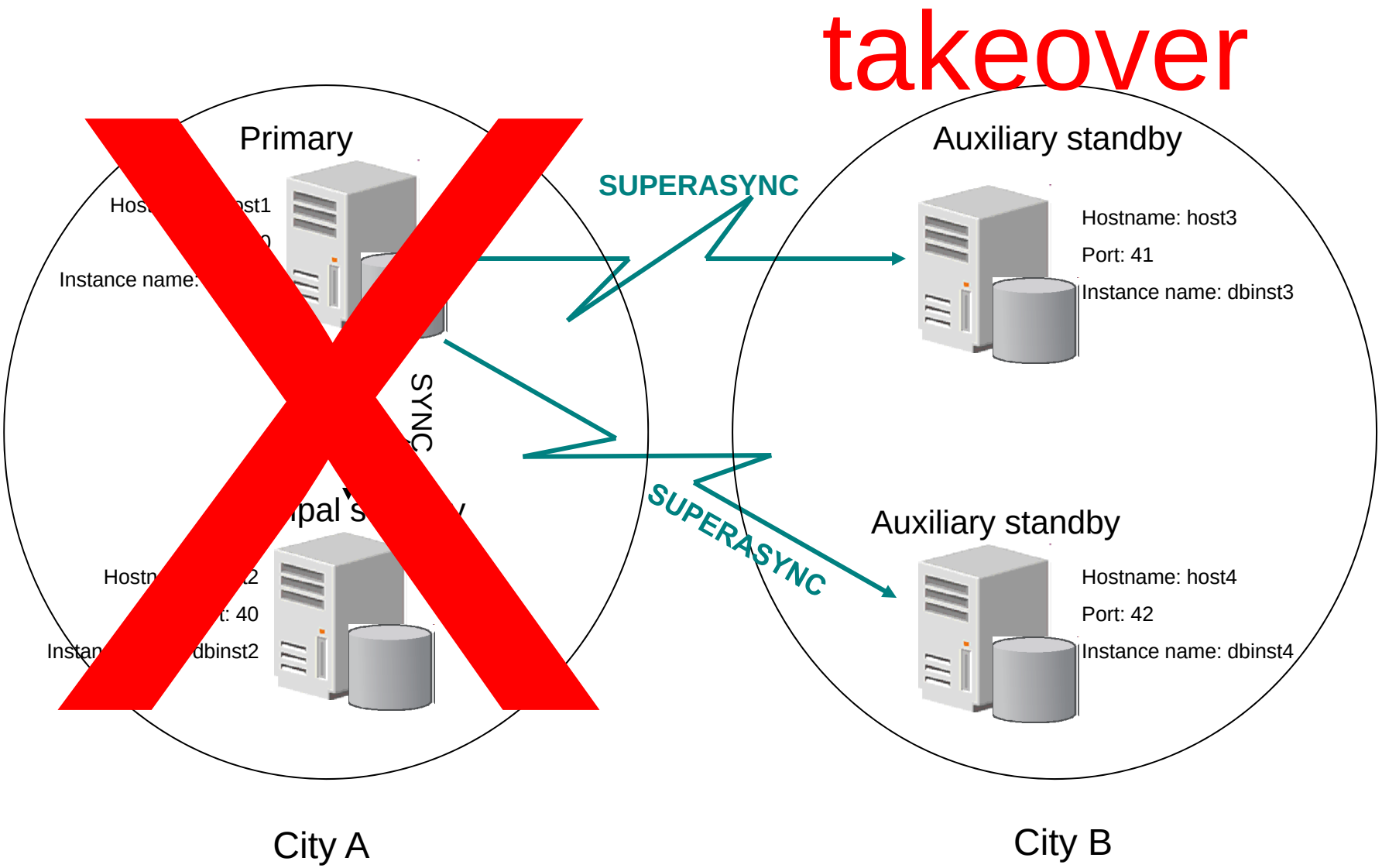
HADR – Principal Standby Takeover



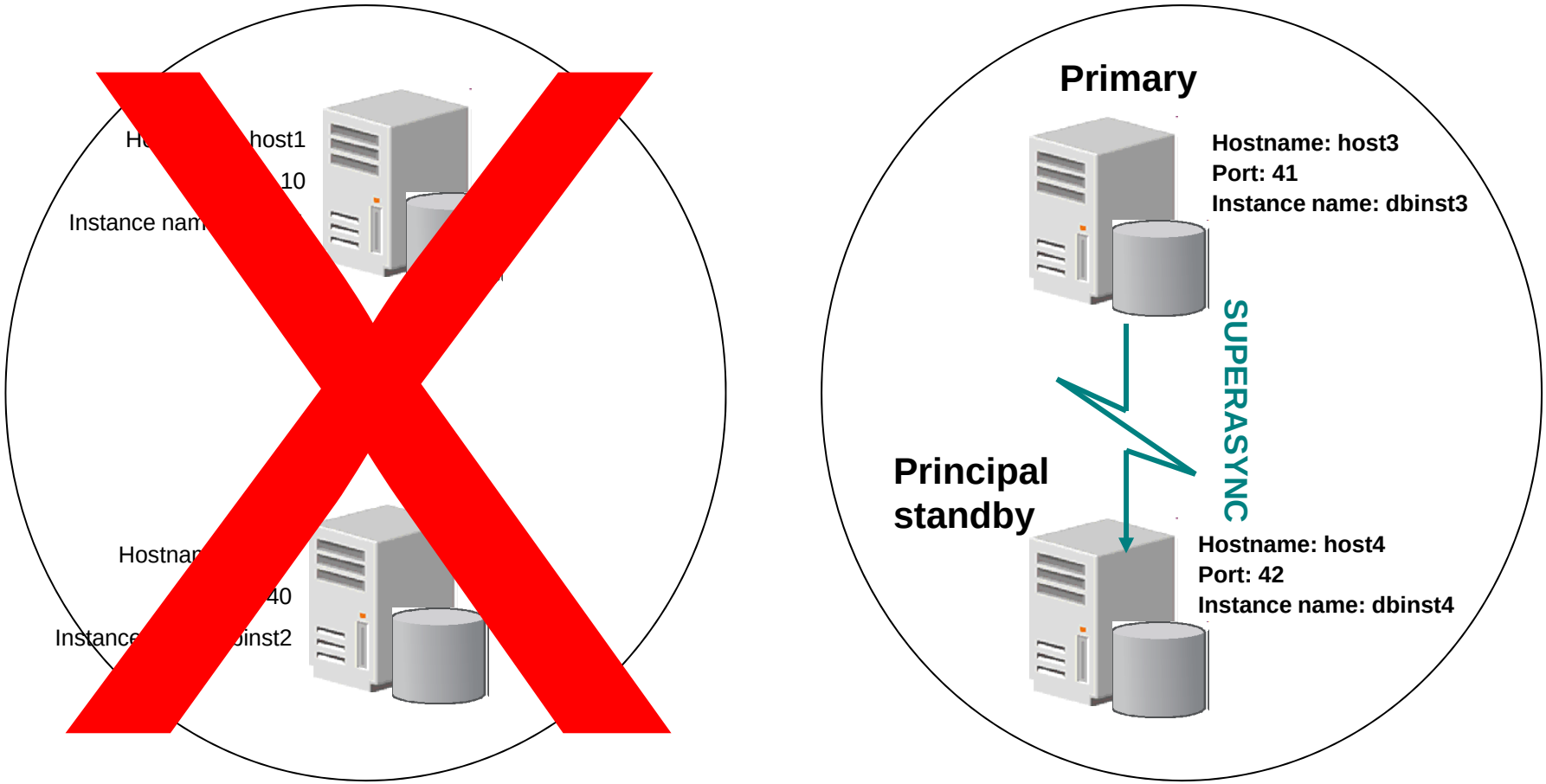
After Issuing Takeover on host2 (Auto-reconfiguration)

Configuration parameter	Host1	Host2	Host3	Host4
Hadr_target_list	**host2:40 | host3:41 | host4:42**	**host1:10 | host3:41 | host4:42**	**host1:10 | host2:40 | host4:42**	**host1:10 | host2:40 | host3:41**
Hadr_remote_host	**host2**	**host1**	**host2**	**host2**
Hadr_remote_svc	**40**	**10**	**40**	**40**
Hadr_remote_inst	**dbinst2**	**dbinst1**	**dbinst2**	**dbinst2**
Hadr_local_host	**host1**	**host2**	**host3**	**host4**
Hadr_local_svc	**10**	**40**	**41**	**42**
Operational Hadr_syncmode	**sync**	**sync**	**superasync**	**superasync**
Effective Hadr_syncmode	**sync**	**N/A**	**superasync**	**superasync**

Host 1 and 2 are Down



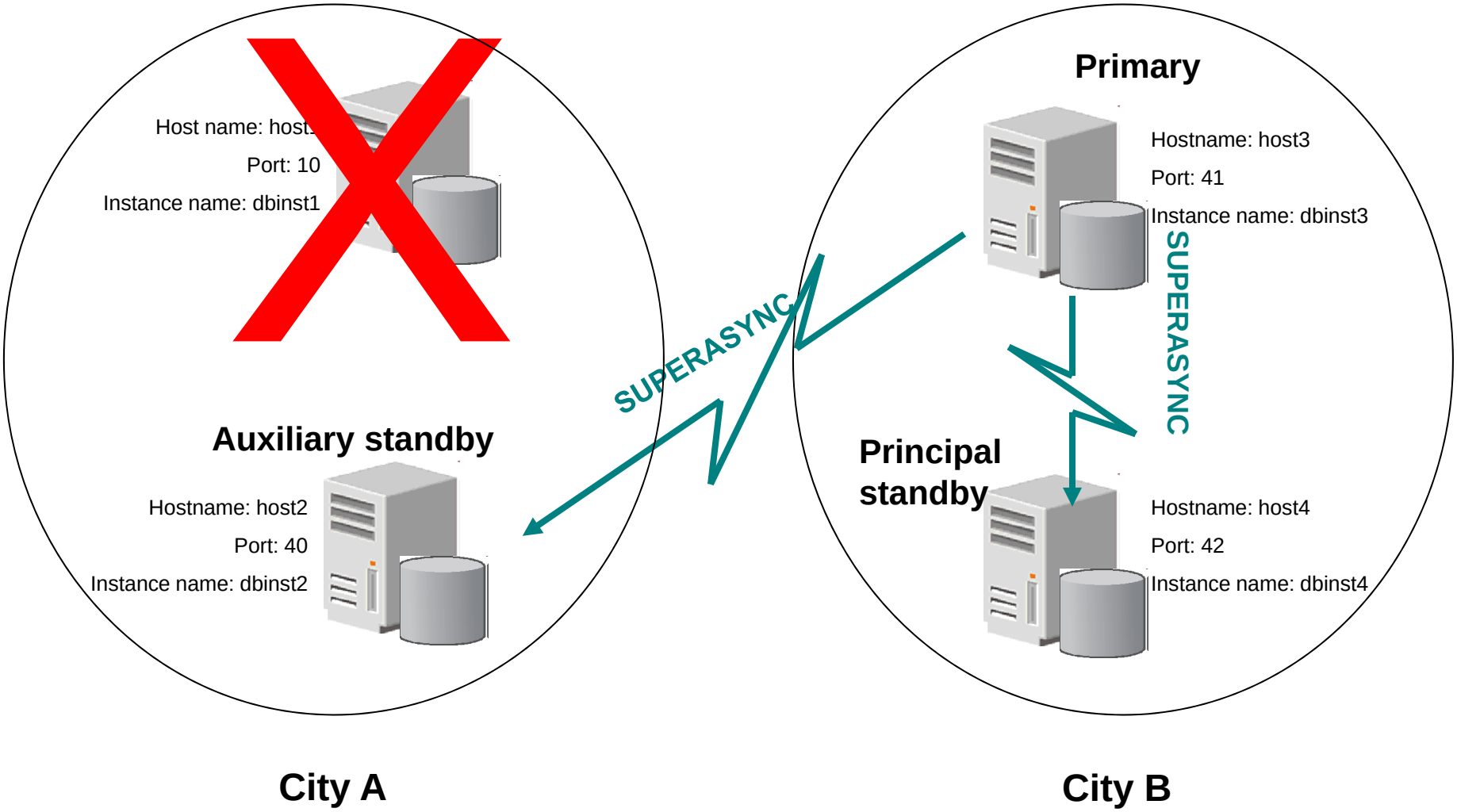
After Issuing Takeover on host3 (Host 1,2 are still down)



After Issuing Takeover on host3 (Host 1,2 are Down)

Configuration parameter	Host1	Host2	Host3	Host4
Hadr_target_list	**host2:40 | host3:41 | host4:42**	**host1:10 | host3:41 | host4:42**	**host1:10 | host2:40 | host4:42**	**host1:10 | host2:40 | host3:41**
Hadr_remote_host	**host2**	**host1**	**host4**	**host3**
Hadr_remote_svc	**40**	**10**	**42**	**41**
Hadr_remote_inst	**dbinst2**	**dbinst1**	**dbinst4**	**dbinst3**
Hadr_local_host	**host1**	**host2**	**host3**	**host4**
Hadr_local_svc	**10**	**40**	**41**	**42**
Operational Hadr_syncmode	**sync**	**sync**	**superasync**	**superasync**
Effective Hadr_syncmode	**N/A**	**sync**	**n/a**	**superasync**

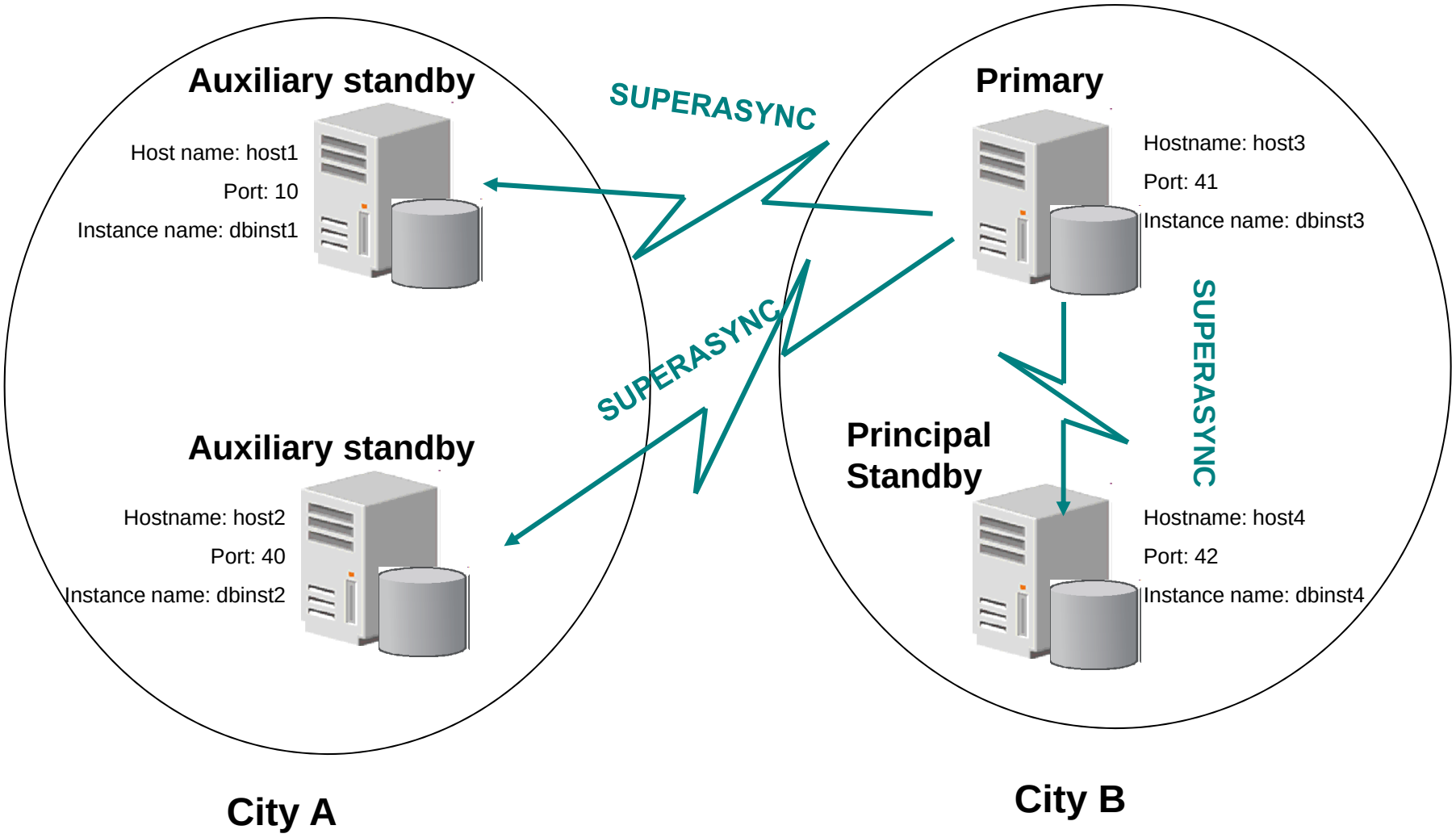
Host 2 is Back Online



Host 2 is Back Online

Configuration parameter	Host1	Host2	Host3	Host4
Hadr_target_list	**host2:40 | host3:41 | host4:42**	**host1:10 | host3:41 | host4:42**	**host1:10 | host2:40 | host4:42**	**host1:10 | host2:40 | host3:41**
Hadr_remote_host	**host2**	**host3**	**host4**	**host3**
Hadr_remote_svc	**40**	**41**	**42**	**41**
Hadr_remote_inst	**dbinst2**	**dbinst3**	**dbinst4**	**dbinst3**
Hadr_local_host	**host1**	**host2**	**host3**	**host4**
Hadr_local_svc	**10**	**40**	**41**	**42**
Operational Hadr_syncmode	**sync**	**sync**	**superasync**	**superasync**
Effective Hadr_syncmode	**N/A**	**superasync**	**n/a**	**superasync**

Host 1 is Back Online



Host 1 is Back Online

Configuration parameter	Host1	Host2	Host3	Host4
Hadr_target_list	**host2:40 | host3:41 | host4:42**	**host1:10 | host3:41 | host4:42**	**host1:10 | host2:40 | host4:42**	**host1:10 | host2:40 | host3:41**
Hadr_remote_host	**host3**	**host3**	**host4**	**host3**
Hadr_remote_svc	**41**	**41**	**42**	**41**
Hadr_remote_inst	**dbinst3**	**dbinst3**	**dbinst4**	**dbinst3**
Hadr_local_host	**host1**	**host2**	**host3**	**host4**
Hadr_local_svc	**10**	**40**	**41**	**42**
Operational Hadr_syncmode	**sync**	**sync**	**superasync**	**superasync**
Effective Hadr_syncmode	**superasync**	**superasync**	**n/a**	**superasync**

Monitoring HADR

- There are a number of methods to monitor the status of HADR databases:
  - db2pd utility and GET SNAPSHOT FOR DATABASE command
    - From the primary = information about the primary and all standbys
    - From a standby = information about that standby and the primary

- To monitor the current role of the database is indicated by the database configuration parameter **hadr_db_role**
- db2pd has been changed to display one entry for each primary-standby pair

```
db2pd -db HADRDB -hadr
```

Summary

- HADR solution is based on a pair (primary and standby) of databases. In case of failure, the standby database can take over the workload
- HADR Reads-on-Standby feature runs concurrent read-only workloads offloading reporting, DSS/BI workloads to standby with minimal impact
- DB2 10 introduces
  - **HADR Multiple-Standby** improving your ability to protect your data while still keeping it highly available.
  - **HADR log spooling** allows the standby to spool log records arriving from the primary
  - **HADR Delayed Replay** can allow a standby database to intentionally remain far behind the primary at all times

The next steps…

The Next Steps…

- Complete the online quiz for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module and select the quiz
- Provide feedback on the module
  - Log onto SKI, go to “My Learning” page
  - Find the module and select the “Leave Feedback” button to leave your comments

Questions?
askdata@ca.ibm.com