Implementing the Witness protocol in Samba

Günther Deschner
<gd@samba.org>

(Red Hat / Samba Team)

About Samba and RedHat

- Currently 7 Samba Team members inside RedHat
- Creators and users of Samba technology for authentication and storage solutions
- Me: 11 years Samba Team member, 8 years RedHat (Samba Maintainer, Identity, Storage)

Agenda

- Witness?
- Failover in SMB1/SMB2
- Failover in SMB1/SMB2 with CTDB
- Failover in SMB3
- The Witness Protocol
- Roadmap for Witness support in Samba
- Further reading & Q/A

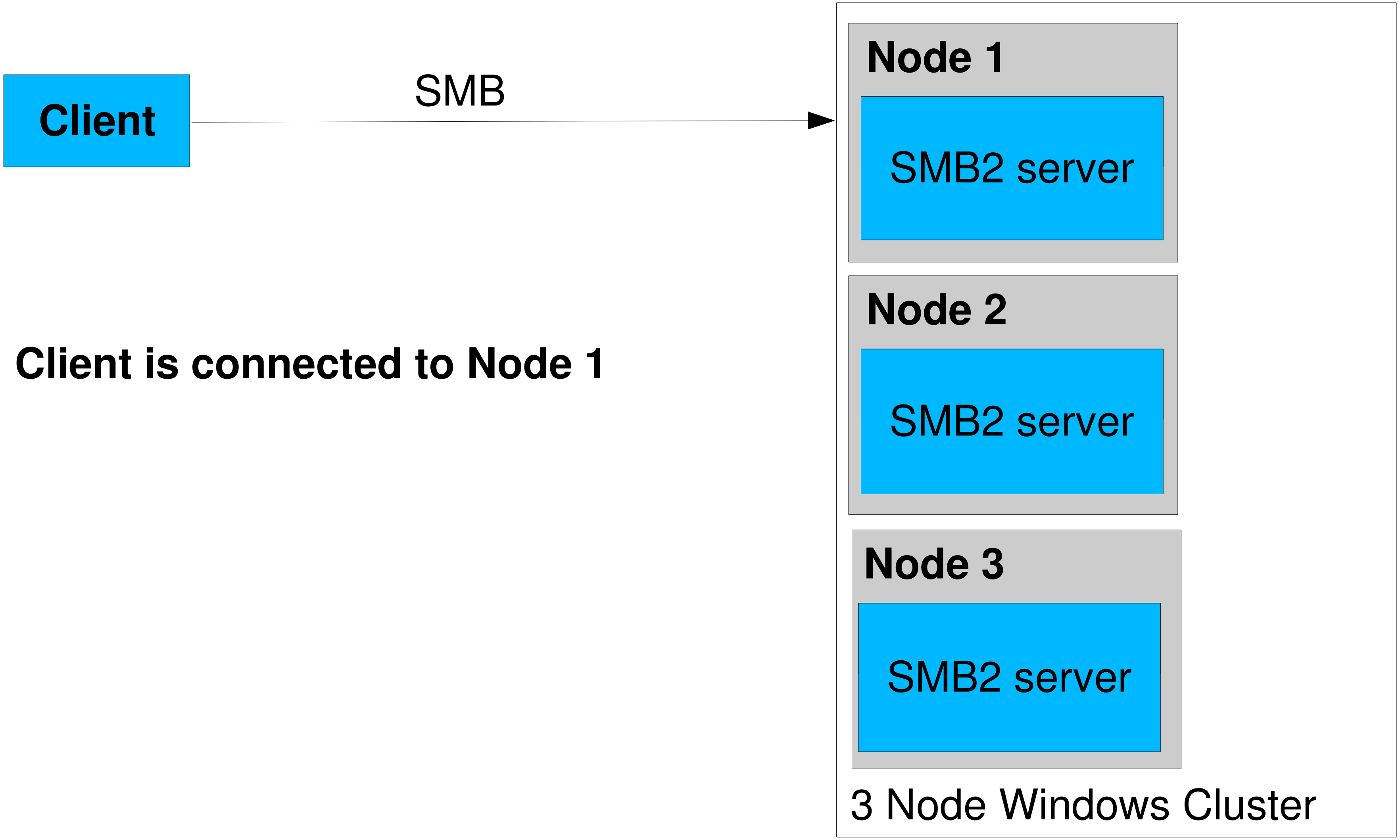
Witness ?

- New DCE/RPC Service to „witness“ availability of other services, in particular SMB3 connection
- Prompt and explicit notifications about failures in highly available systems
- Allows Continous Availability of SMB shares in clustered environments
- Controlled way of dealing with reconnects instead of detecting failures due to timeouts
- Available with SMB3

Failover in SMB1/SMB2

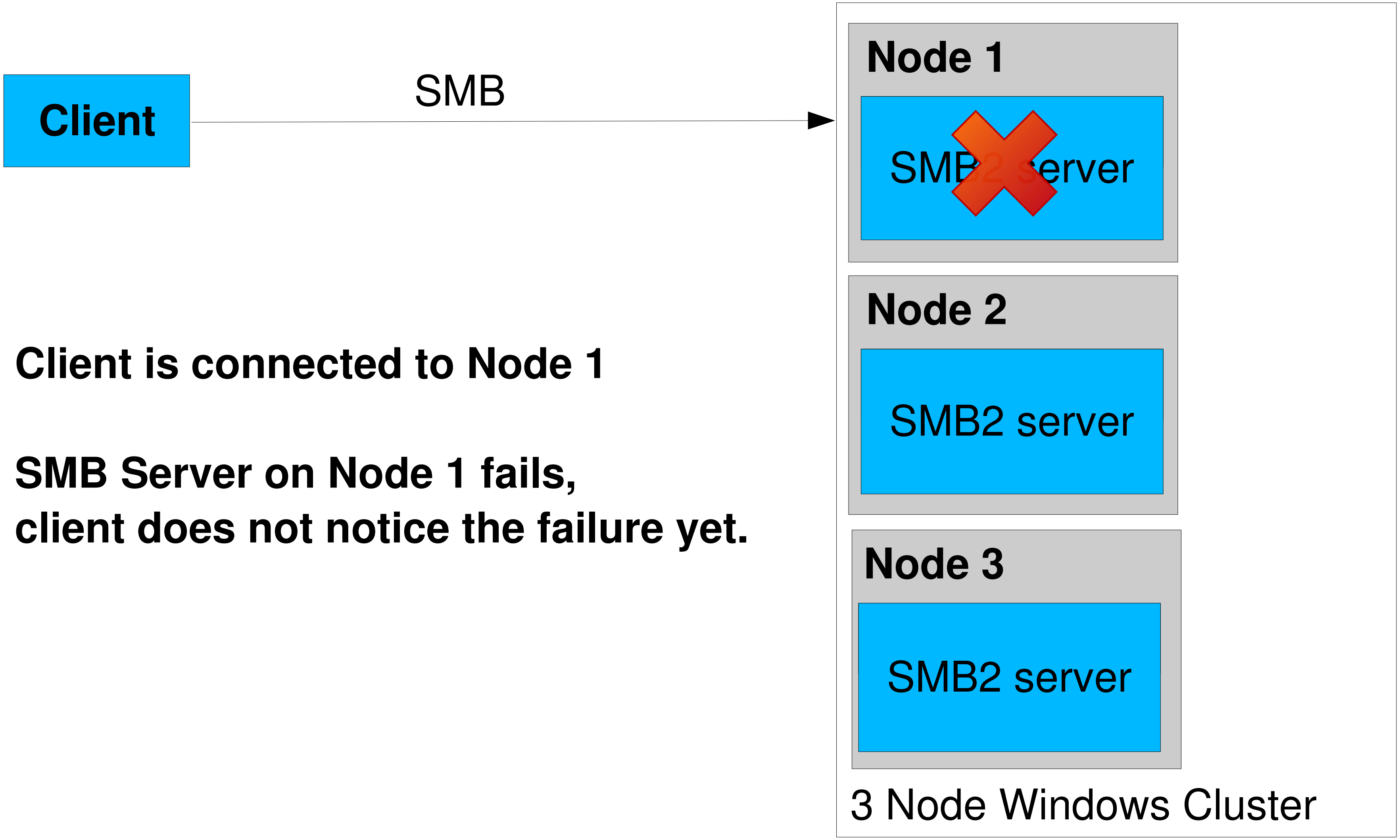
- Uncontrolled, clients detect unavailability by running into timeouts or by using keep alive mechanisms
- Clients reconnect after TCP/IP connection timeout
- Slow, unreliable, unpredictable
- Not all applications deal with stale connections good enough

Failover in SMB1/SMB2



Client is connected to Node 1

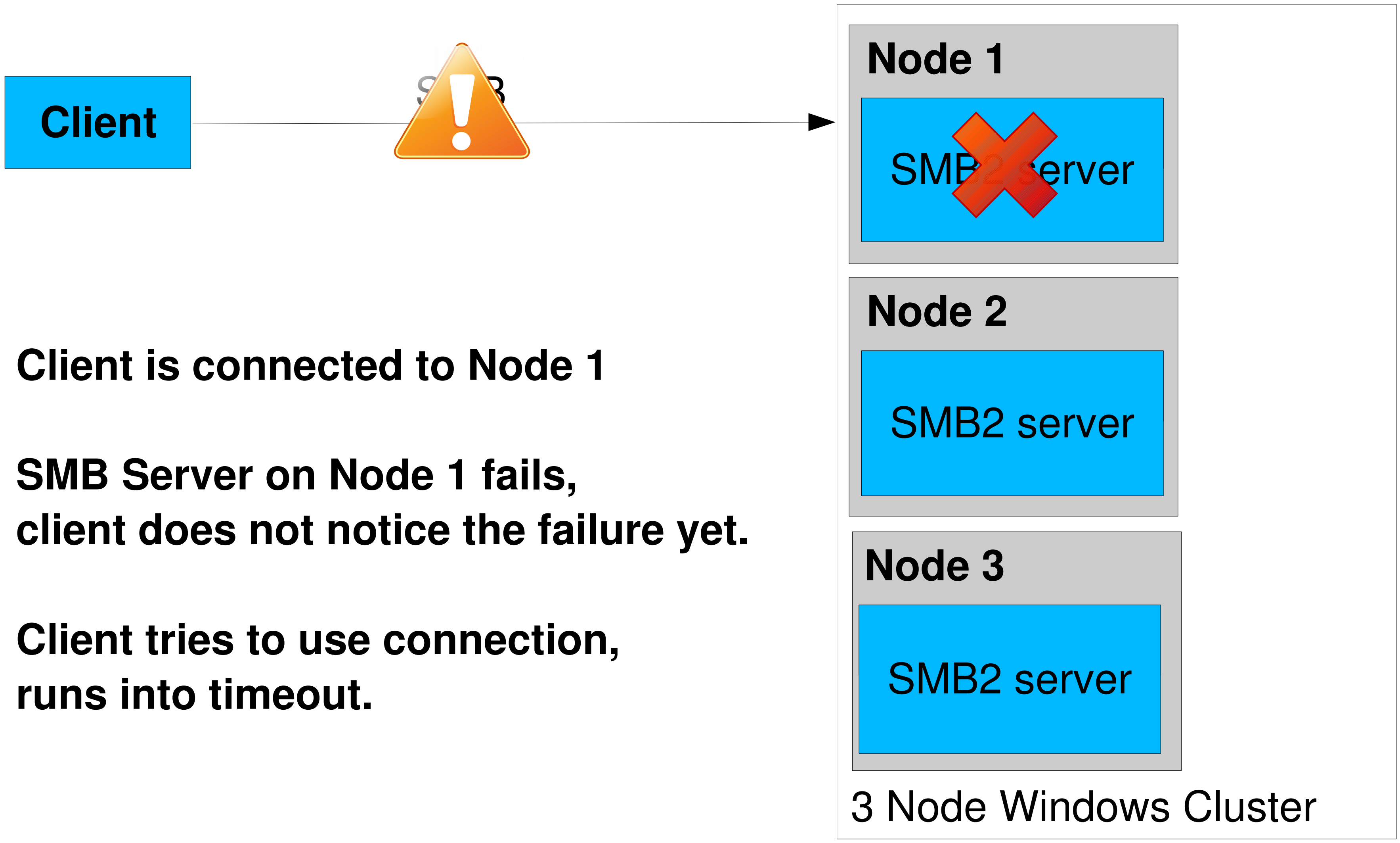
Failover in SMB1/SMB2



Client is connected to Node 1

SMB Server on Node 1 fails,
client does not notice the failure yet.

Failover in SMB1/SMB2

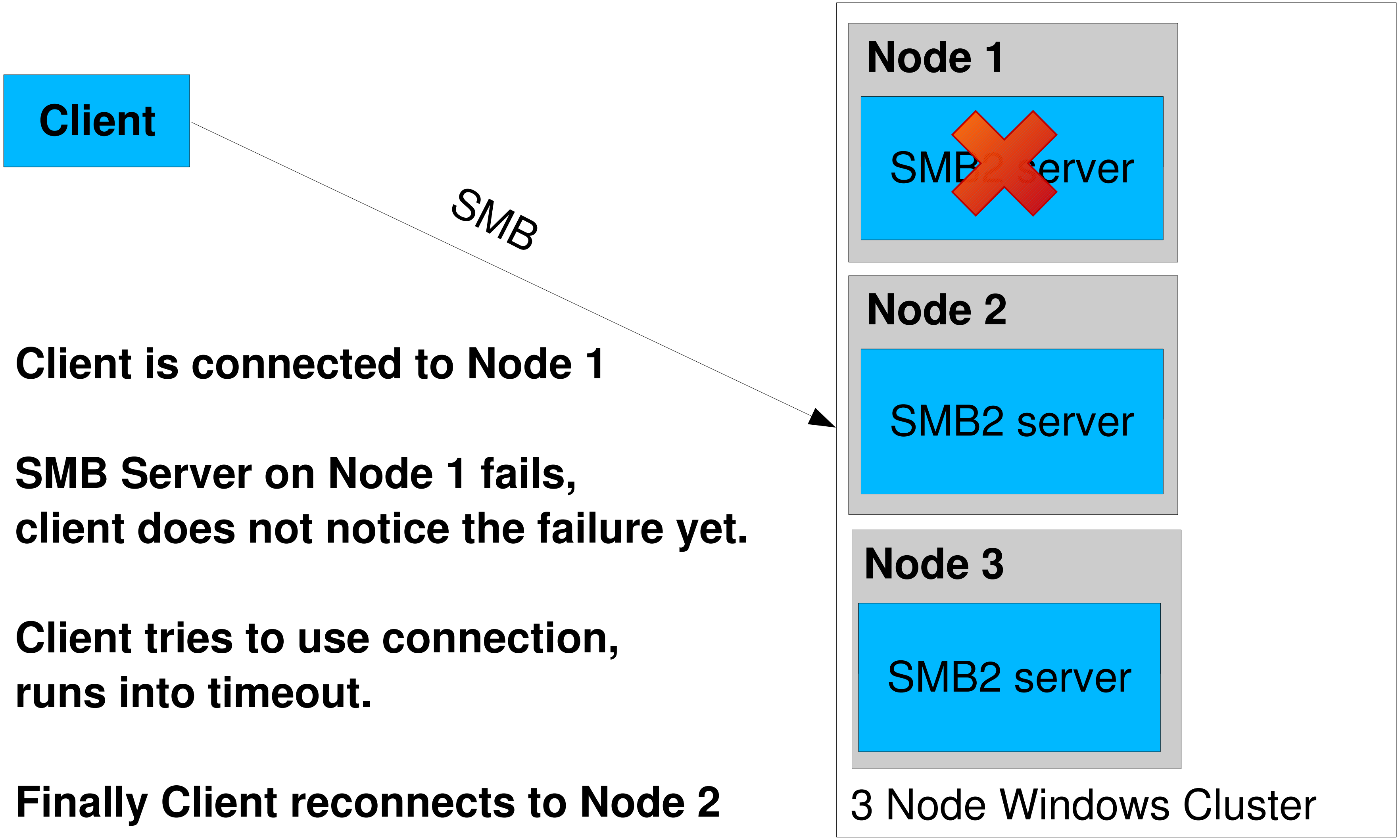


Client is connected to Node 1

SMB Server on Node 1 fails,
client does not notice the failure yet.

Client tries to use connection,
runs into timeout.

Failover in SMB1/SMB2



Client is connected to Node 1

SMB Server on Node 1 fails,
client does not notice the failure yet.

Client tries to use connection,
runs into timeout.

Finally Client reconnects to Node 2

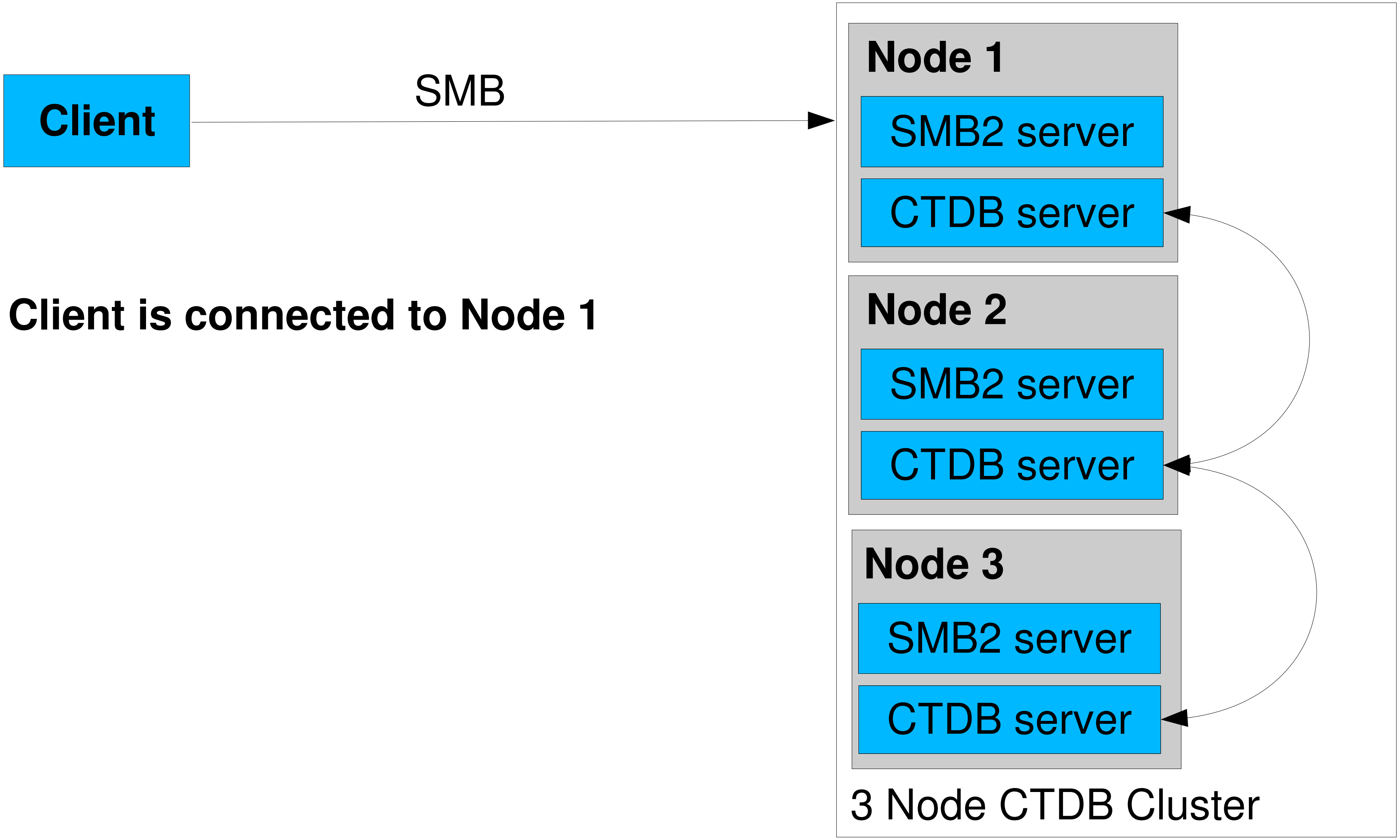
Failover in SMB1/SMB2 with CTDB

- In a Samba cluster with CTDB the cluster usually is aware of failures before the client is
- In case of failure CTDB can proactively route the clients to another node
- With CTDB the cluster coordinates the failover, not the client

Failover in SMB1/SMB2 with CTDB

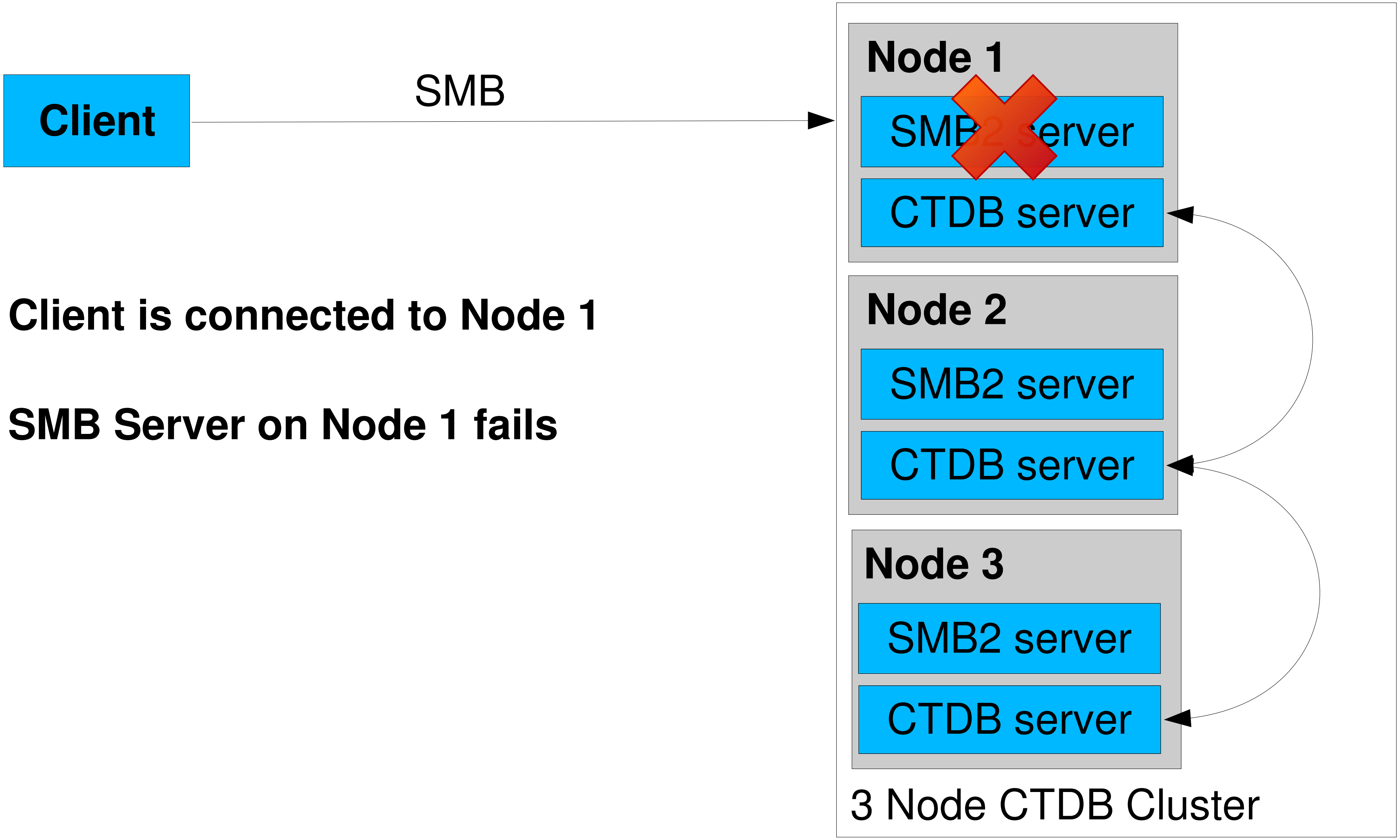
- CTDB uses Tickle ACKs to speedup recovery
- Tickle ACKs are TCP ACK packets with invalid sequence and acknowledge numbers
- They cause a TCP connection to be recognized as been disrupted, Client reconnects immediately
- The Tickle ACK mechanism has been discovered by Tridge in 2007 while working on CTDB
- The Cluster Resource Manager project pacemaker also provides a Tickle ACK implementation (as part of the portblock resource agent)

Failover in SMB1/SMB2 with CTDB



Client is connected to Node 1

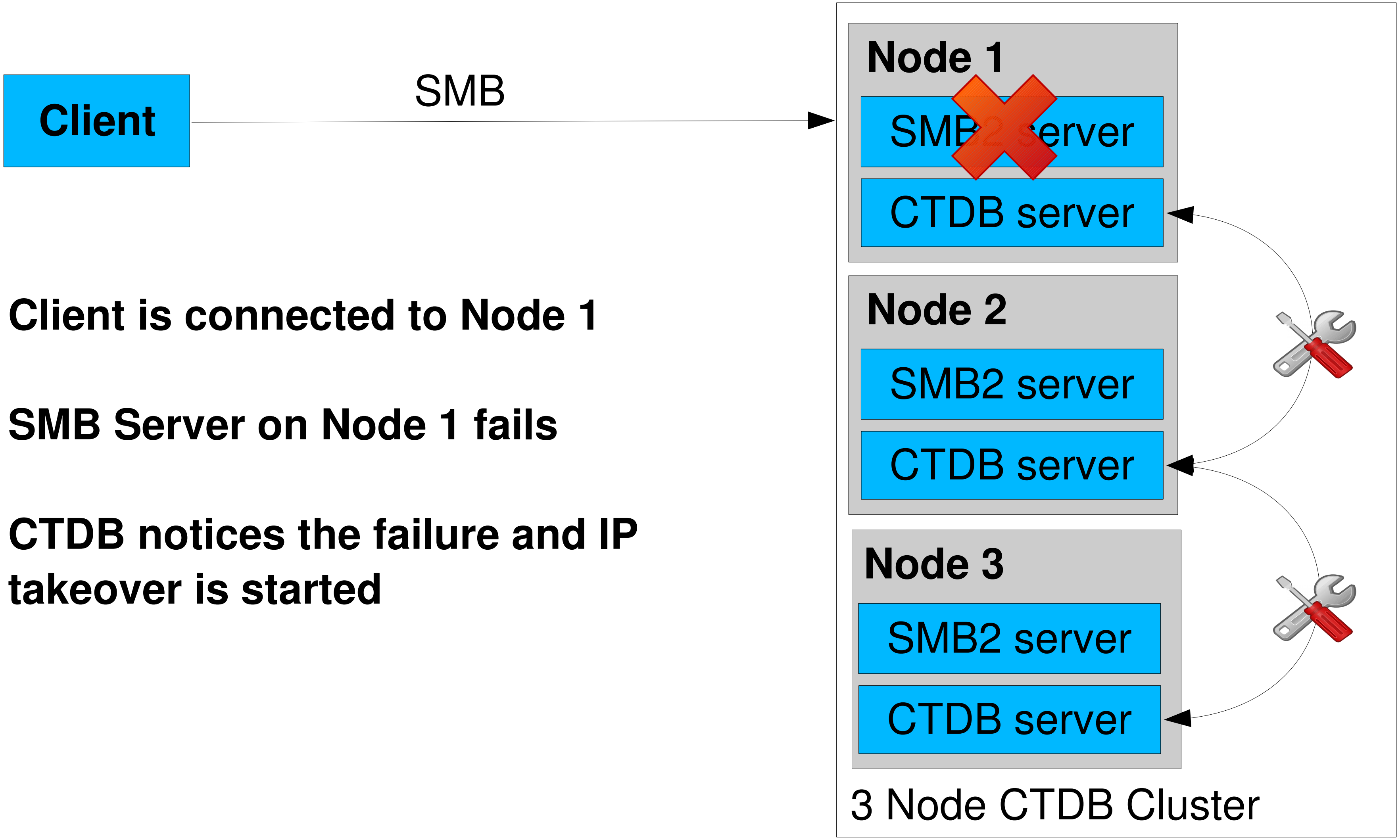
Failover in SMB1/SMB2 with CTDB



Client is connected to Node 1

SMB Server on Node 1 fails

Failover in SMB1/SMB2 with CTDB

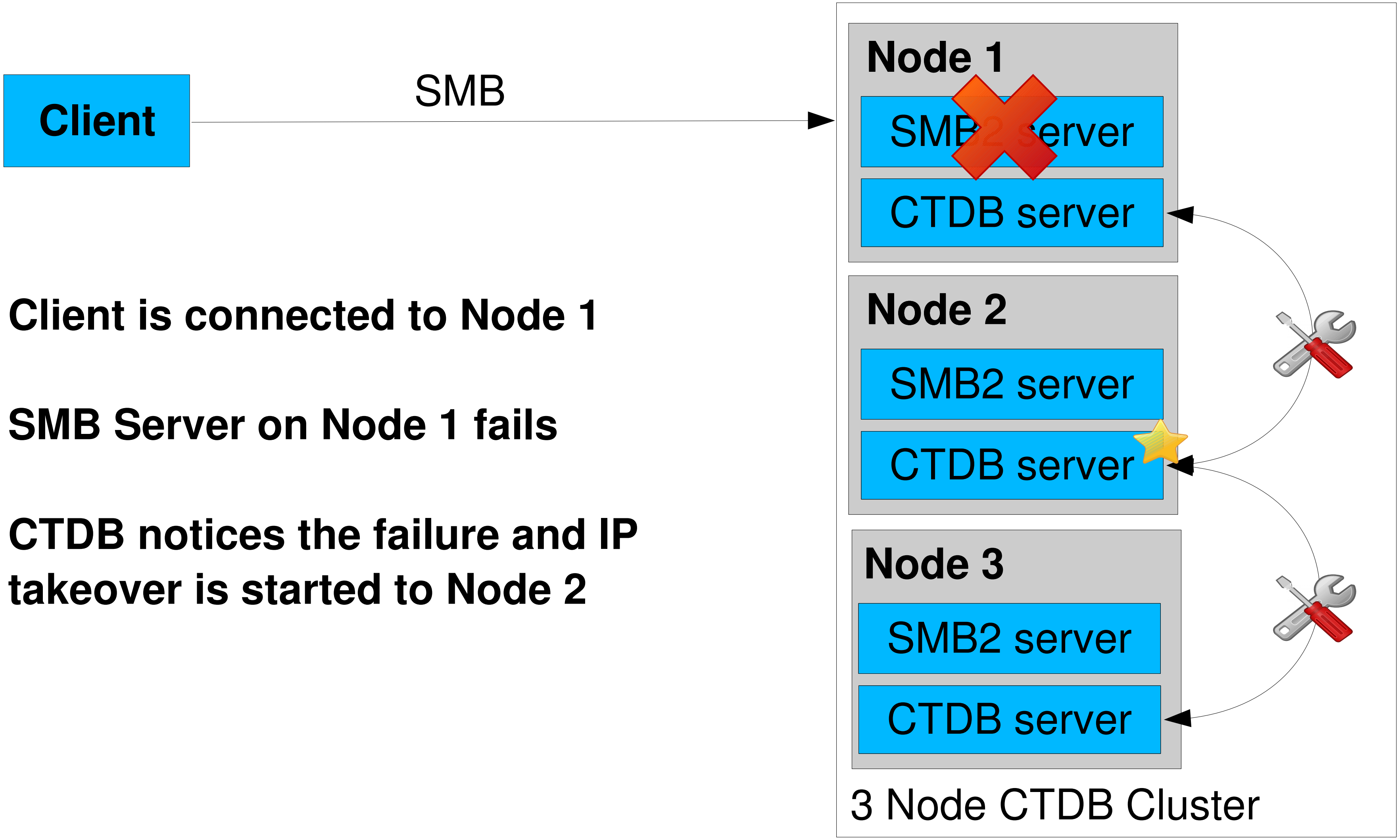


Client is connected to Node 1

SMB Server on Node 1 fails

CTDB notices the failure and IP takeover is started

Failover in SMB1/SMB2 with CTDB

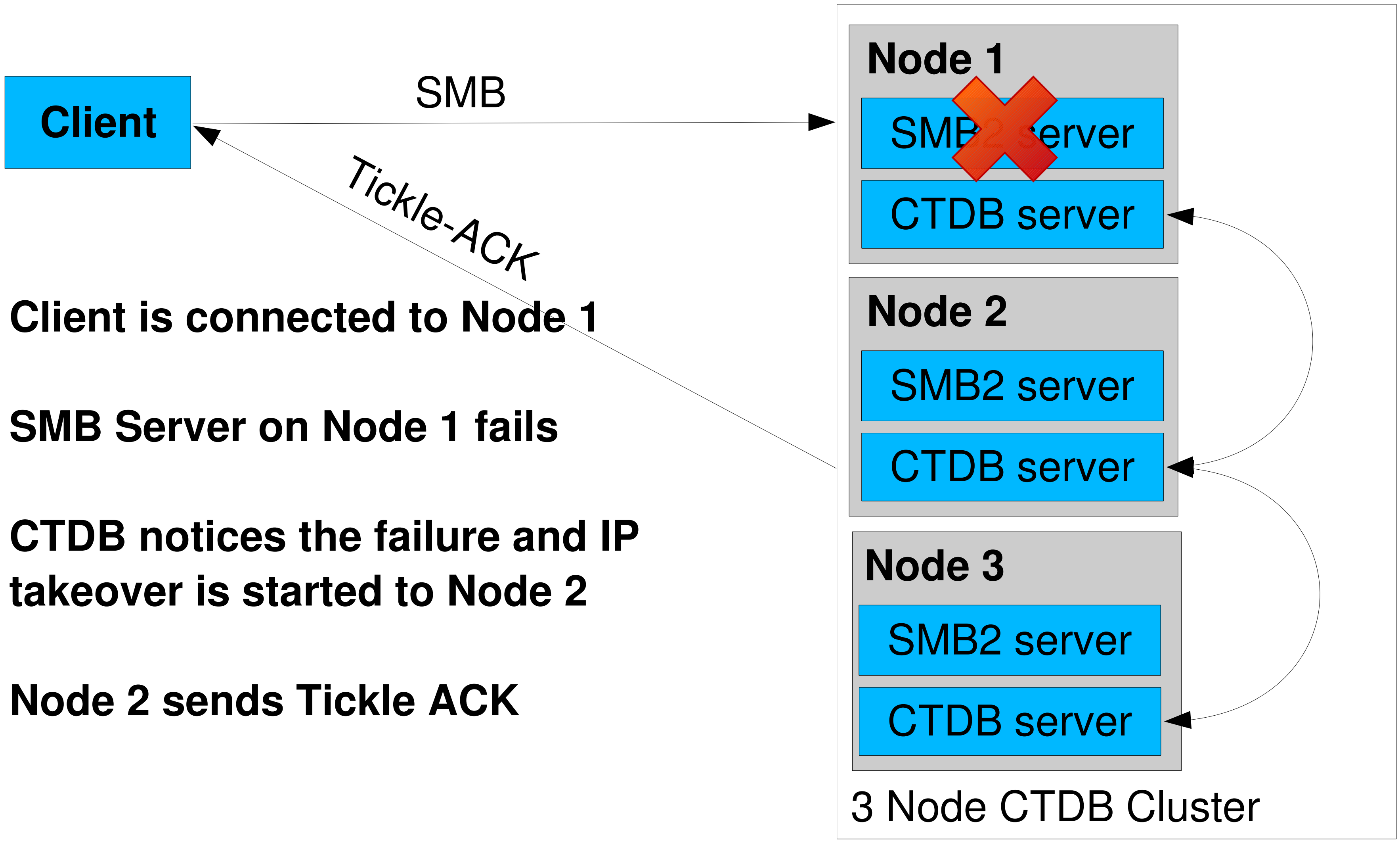


Client is connected to Node 1

SMB Server on Node 1 fails

CTDB notices the failure and IP takeover is started to Node 2

Failover in SMB1/SMB2 with CTDB



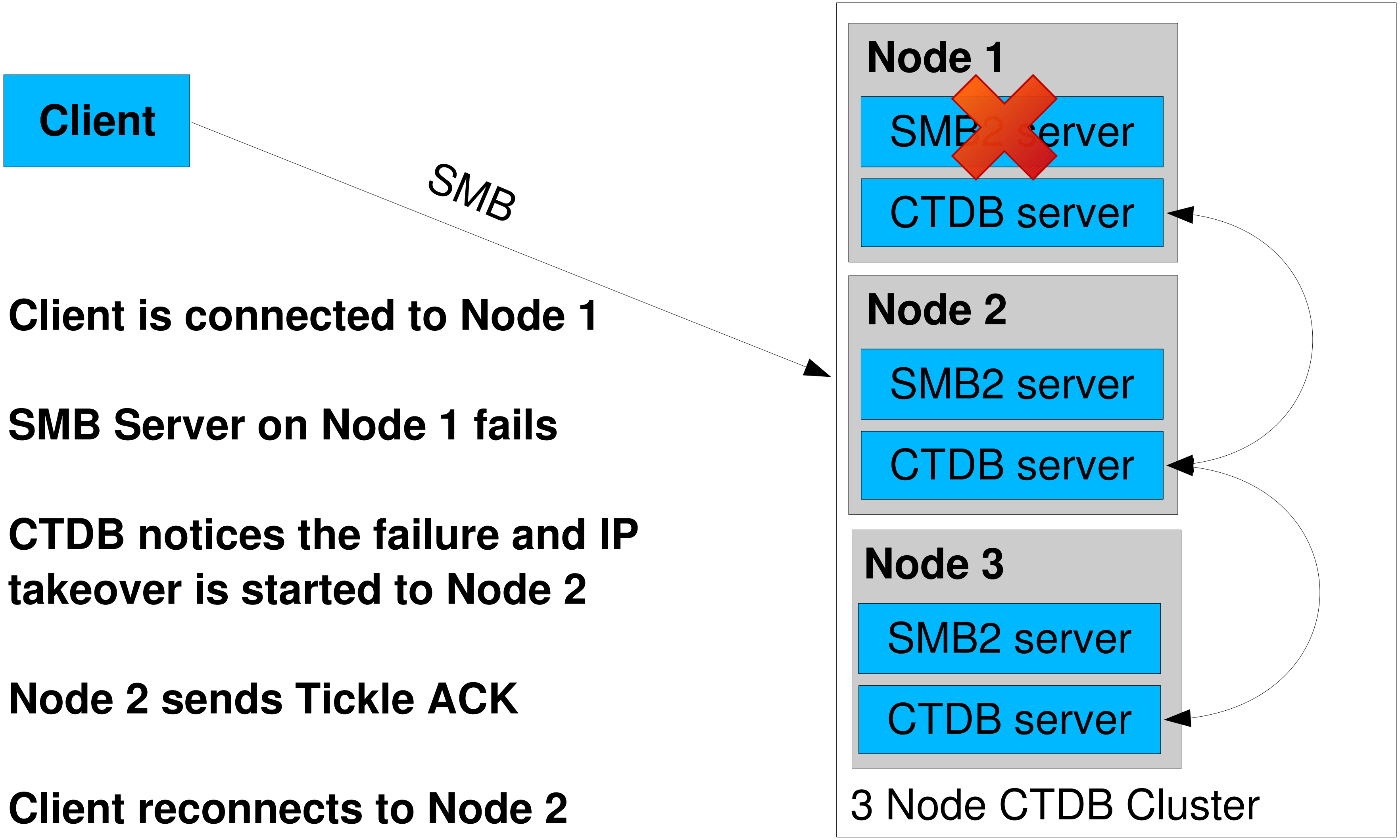
Client is connected to Node 1

SMB Server on Node 1 fails

CTDB notices the failure and IP takeover is started to Node 2

Node 2 sends Tickle ACK

Failover in SMB1/SMB2 with CTDB



Client is connected to Node 1

SMB Server on Node 1 fails

CTDB notices the failure and IP takeover is started to Node 2

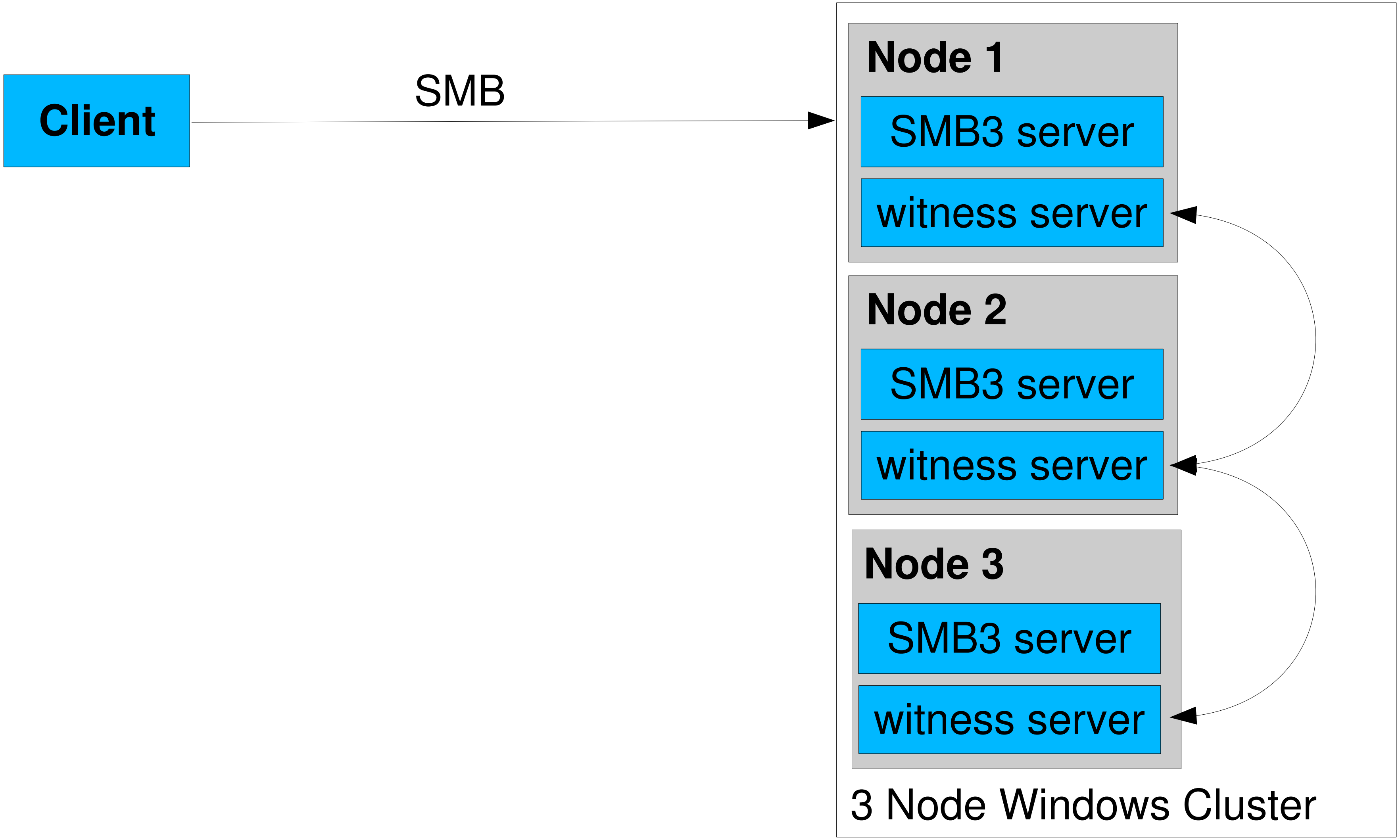
Node 2 sends Tickle ACK

Client reconnects to Node 2

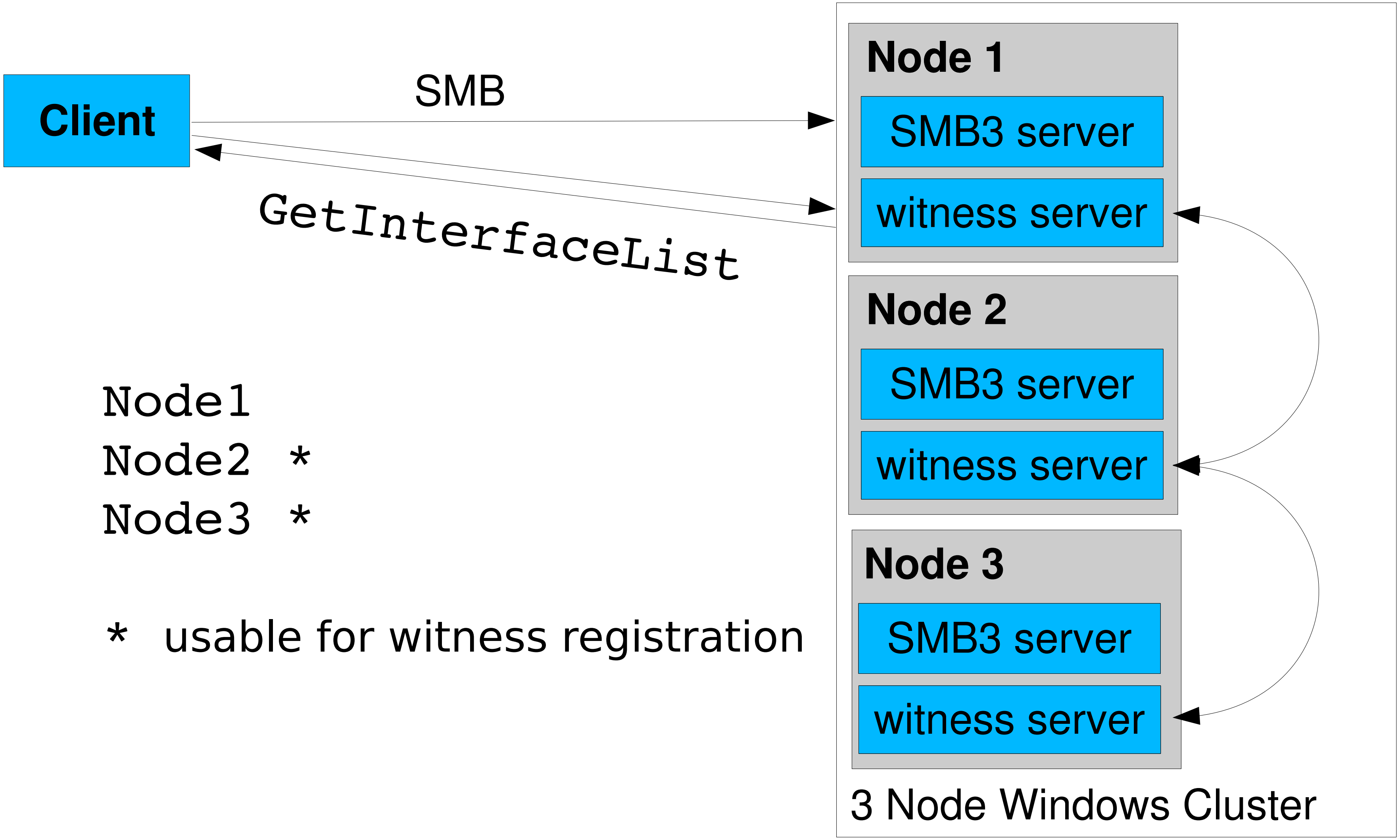
Failover in SMB3

- SMB3 provides new feature SMB Transparent Failover:
  - Persistent handles
  - Continous availability
  - Witness service
- Faster recovery from unplanned node failures
- Allow planned and controlled migration of clients to other Cluster nodes

Failover in SMB3



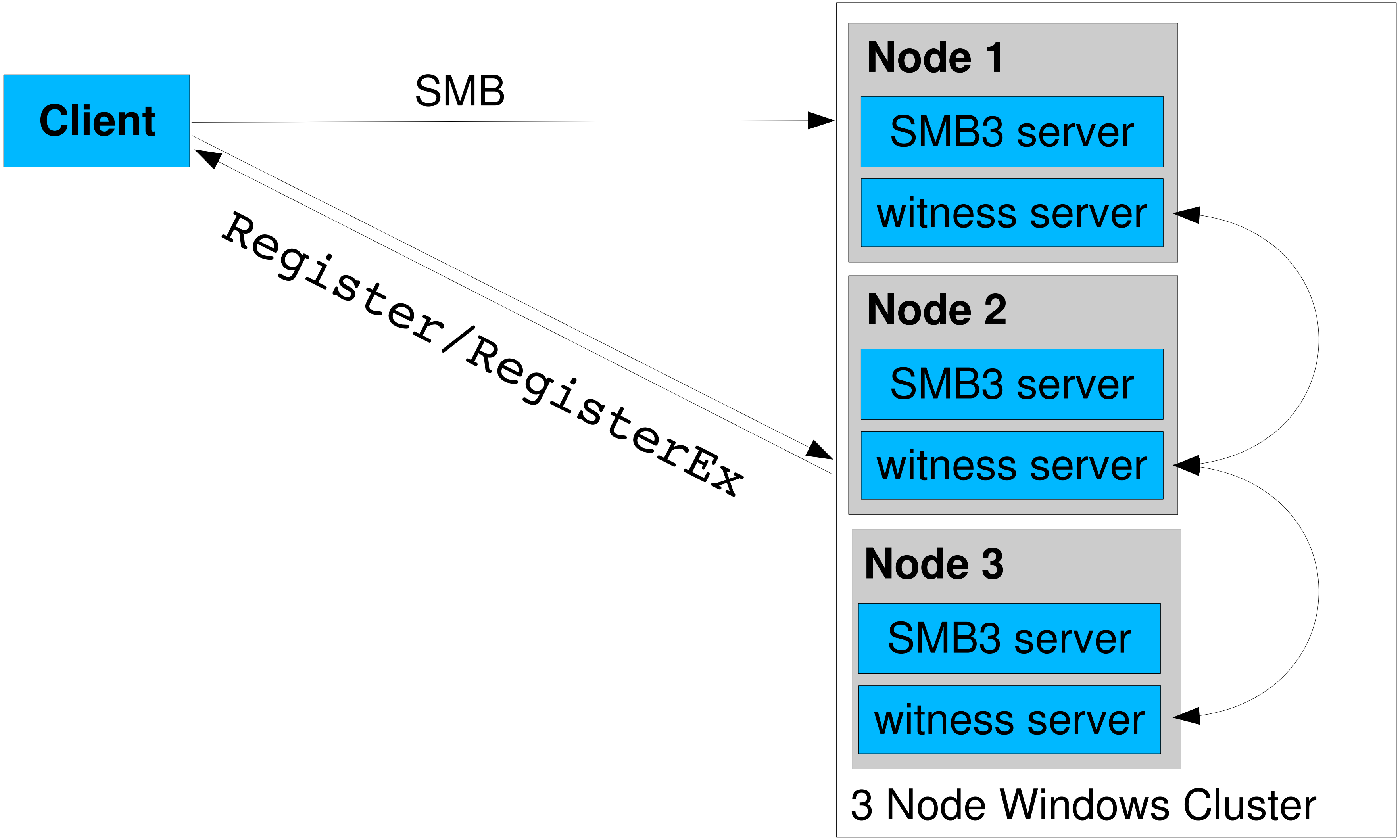
Failover in SMB3



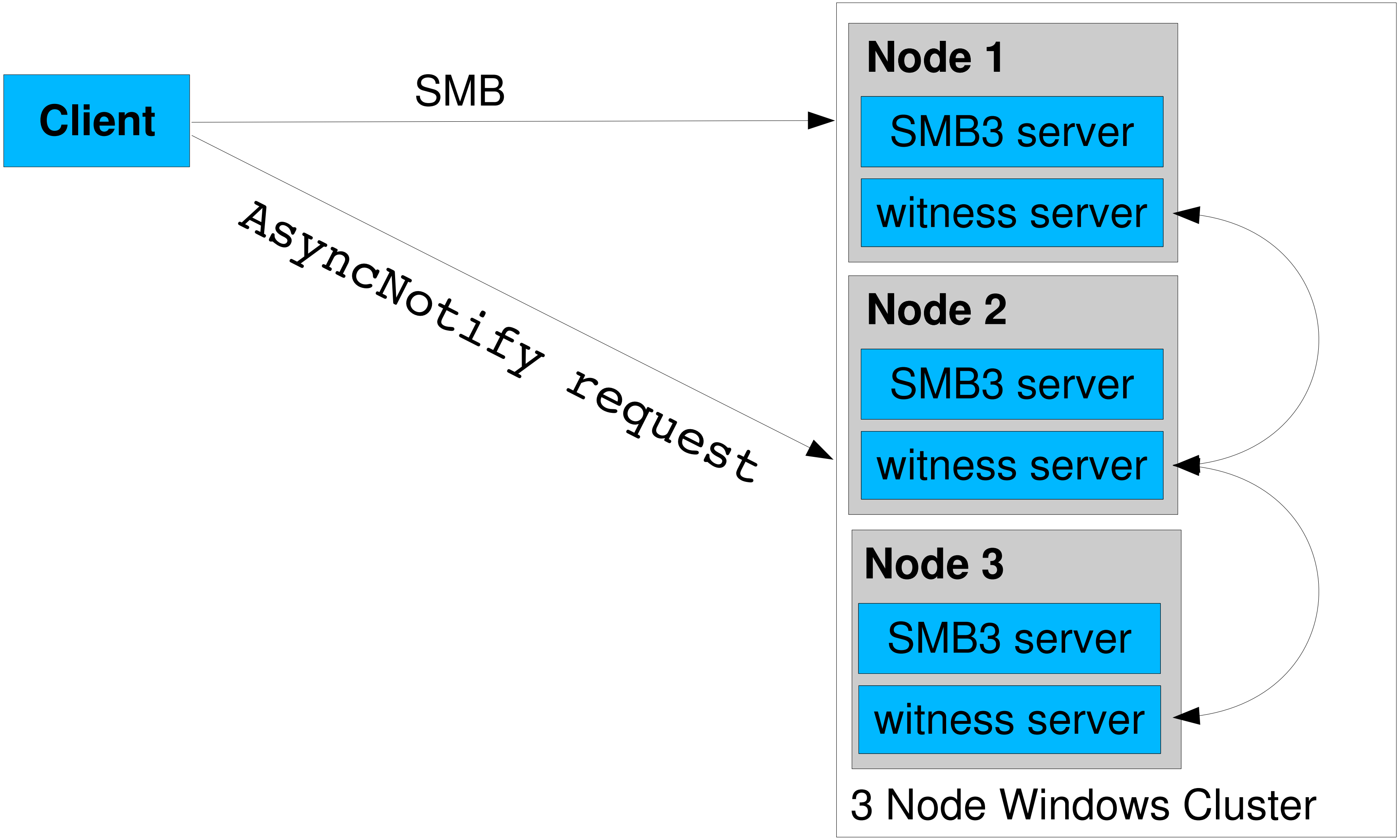
```
Node1
Node2  *
Node3  *

*  usable for witness registration
```

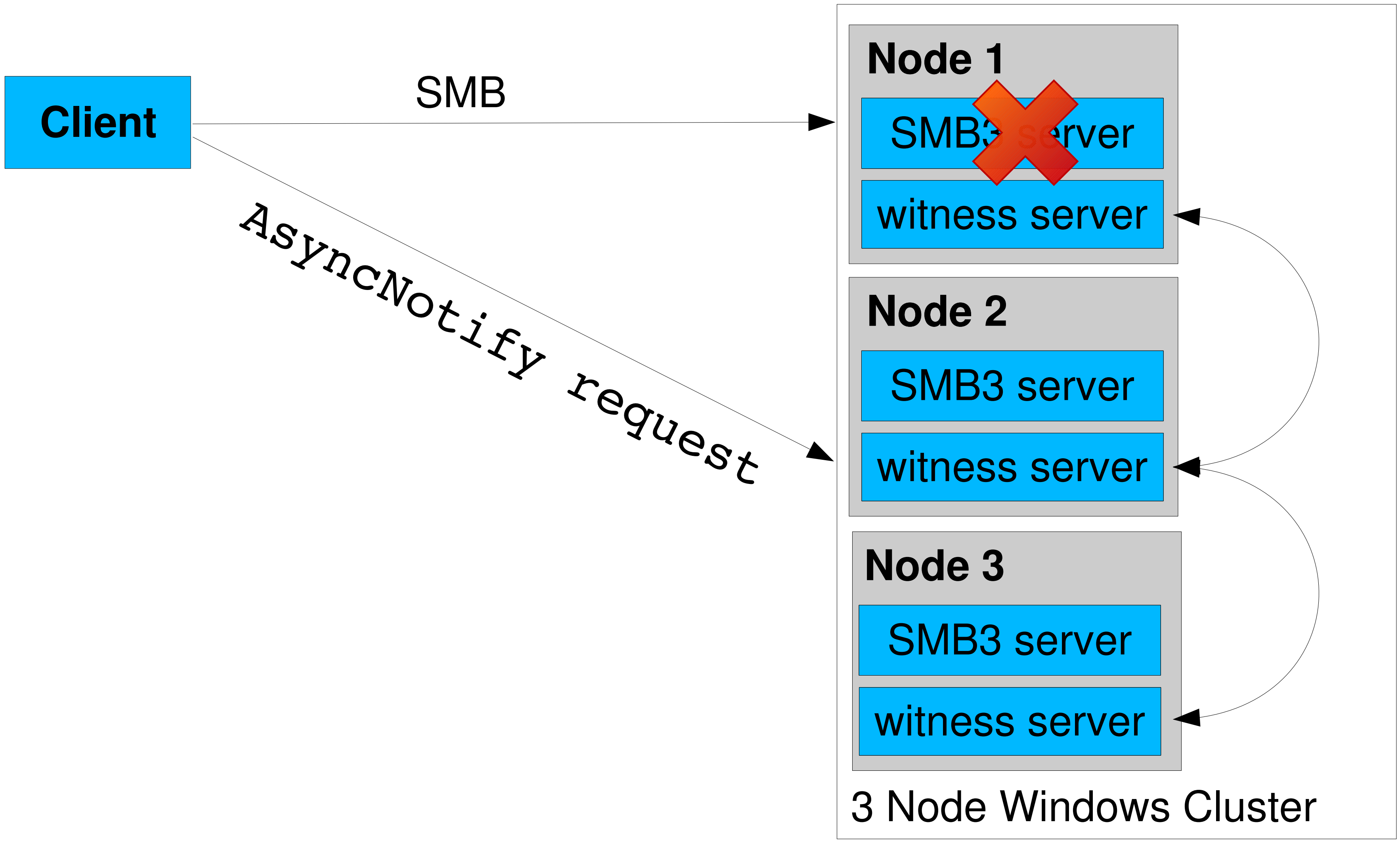
Failover in SMB3



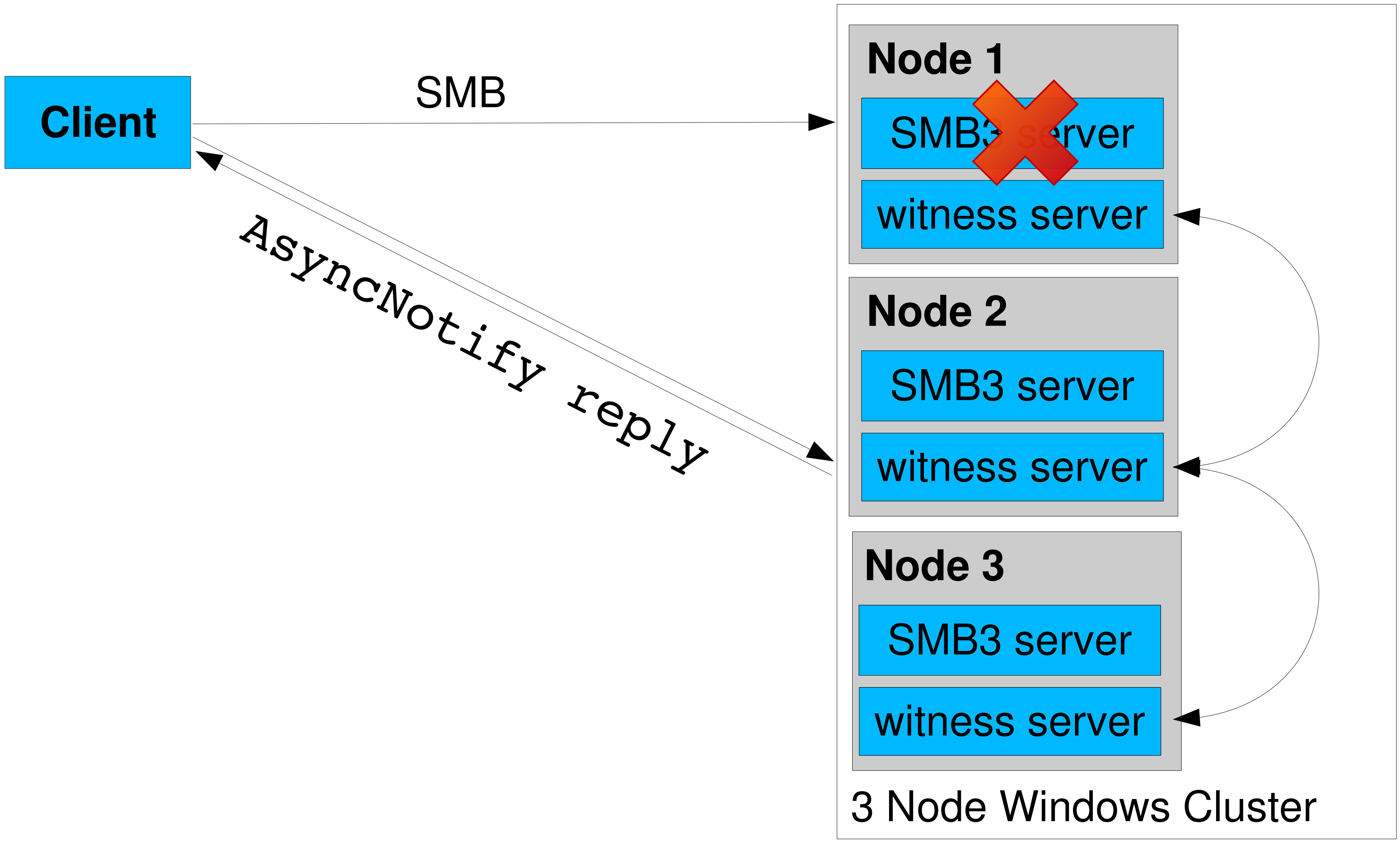
Failover in SMB3



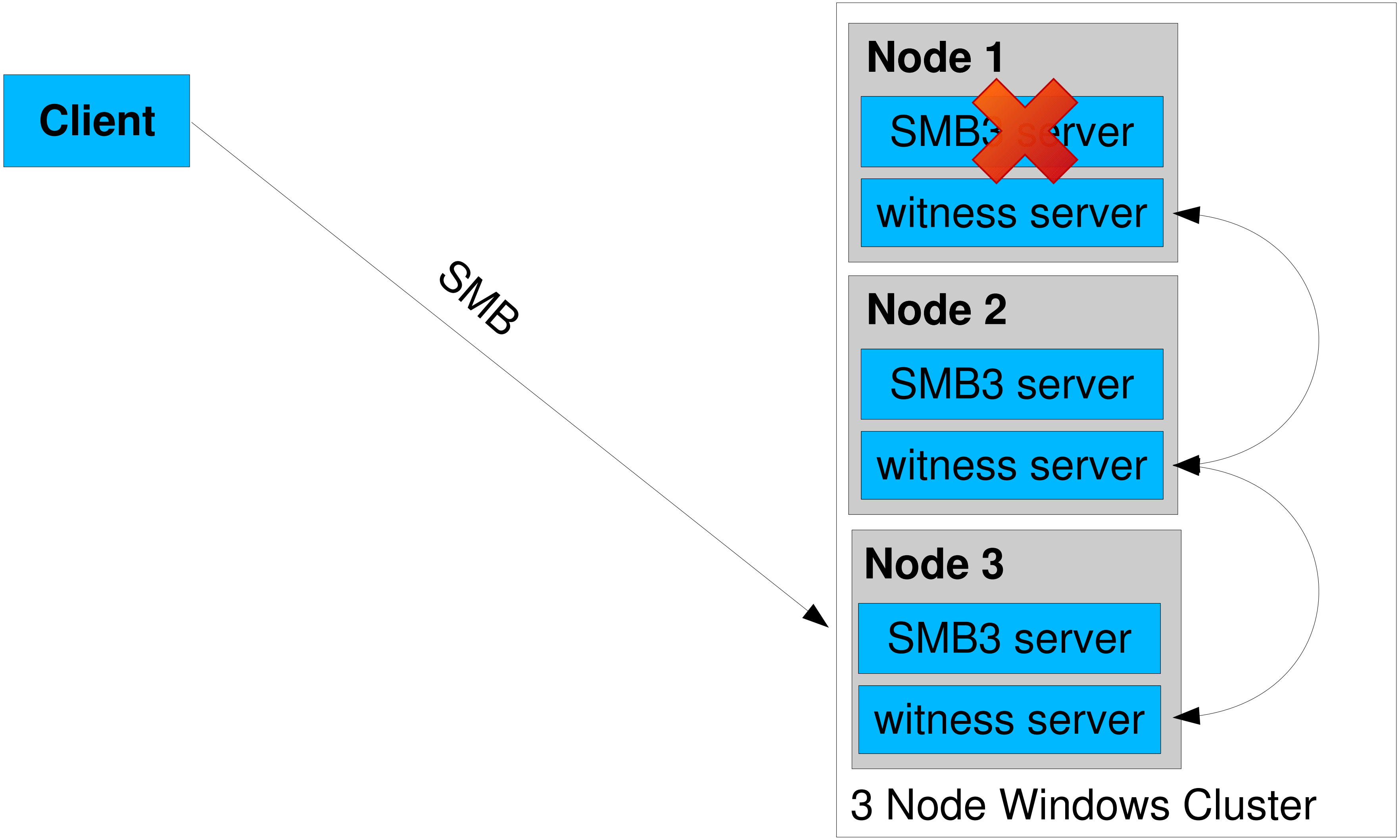
Failover in SMB3



Failover in SMB3



Failover in SMB3



Wait. So why a new protocol ?

- Witness is not only about failover when unexpected failures occur
- Witness allows to programmatically control the client
- Administrators can use witness to control the client use of server ressources (loadbalancing, planned server maintainence)

The witness interface

- Surprisingly short spec (only 47 pages)
- Version 1, SMB 3.0 (Windows 2012, Windows 8)
- Version 2, SMB 3.02 (Windows 2012 R2, Windows 8.1)
- Only 5 opcodes in the interface:
  - `_witness_GetInterfaceList`
  - `_witness_Register`
  - `_witness_Unregister`
  - `_witness_AsyncNotify`
  - `_witness_RegisterEx (witness version 2)`

GetInterfaceList

```
DWORD WitnessrGetInterfaceList(
   [in] handle_t Handle,
   [out] PWITNESS_INTERFACE_LIST * InterfaceList);
```

- Returns list of network interfaces with IPv4 and/or IPv6 addresses
- Each interface carries information about the interfaces version, state and whether it is a good candidate for witness use

Witness_InterfaceInfo

```
interfaces: struct witness_interfaceInfo
   group_name       : 'MTHELENA'
   version          : WITNESS_UNSPECIFIED_VERSION (-1)
   state            : WITNESS_STATE_AVAILABLE (1)
   ipv4             : 192.168.56.108
   ipv6             : ::
   flags            : 0x00000005 (5)
         1: WITNESS_INFO_IPv4_VALID
         0: WITNESS_INFO_Ipv6_VALID
         1: WITNESS_INFO_WITNESS_IF
```

Register

```
DWORD WitnessrRegister(
   [in] handle_t Handle,
   [out] PPCONTEXT_HANDLE ppContext,
   [in] ULONG Version,
   [in] [string] [unique] LPWSTR NetName,
   [in] [string] [unique] LPWSTR IpAddress,
   [in] [string] [unique] LPWSTR ClientComputerName);
```

- Only Wintess V1 can be used as version
- Registers client for notify events
- Registration is server-based (NetName) (not share-based)

UnRegister

```
DWORD WitnessrUnRegister(
   [in] handle_t Handle,
   [in] PCONTEXT_HANDLE pContext);
```

- Cleans up client registration

AsyncNotify

```
DWORD WitnessrAsyncNotify(
   [in] handle_t Handle,
   [in] PCONTEXT_HANDLE_SHARED pContext,
   [out] PRESP_ASYNC_NOTIFY * pResp);
```

- Asychronous call
- Clients send request and wait, and wait, and wait...
- Only in the event of a notification issued by the cluster the client receives a reply
- Witness keep-alive mechanism available
  in Witness v2 (SMB 3.02)

AsyncNotify call

- 4 different events are currently defined in the protocol:
- WITNESS_NOTIFY_RESOURCE_CHANGE
  - Notify about a resource change state (available, unavailable)
- WITNESS_NOTIFY_CLIENT_MOVE
  - Notify a connected client to move no another node
- WITNESS_NOTIFY_SHARE_MOVE (only v2)
  - Notify that a share has been moved to another node
- WITNESS_NOTIFY_IP_CHANGE (only v2)
  - Notify about an ip address change (online, offline)

RegisterEx

```
DWORD WitnessrRegisterEx(
   [in] handle_t Handle,
   [out] PPCONTEXT_HANDLE ppContext,
   [in] ULONG Version,
   [in] [string] [unique] LPWSTR NetName,
   [in] [string] [unique] LPWSTR ShareName,
   [in] [string] [unique] LPWSTR IpAddress,
   [in] [string] [unique] LPWSTR ClientComputerName,
   [in] ULONG Flags,
   [in] ULONG KeepAliveTimeout);
```

- Available with Windows 2012 R2 (Witness v2)
- Witness keepalive as client can define KeepAliveTimeout
- Server returns with ERROR_TIMEOUT after KeepAliveTimeout has expired (Windows 8.1 default 120 seconds)

RegisterEx

- Optional ShareName allows share notify instead of server notify
- Allows Asymetric Fileshares (SMB 3.02)

Roadmap for Witness support in Samba

- Early PoC implementation by Gregor Beck and Stefan Metzmacher from 2012
- Wireshark dissector for witness protocol (not upstream yet)
- Full IDL and torture tests in Samba Git repository upstream
- Witness Service is on Samba Roadmap as a funded project
- At RedHat José A. Rivera <jarrpa@samba.org> and me are working on a witness implementation
- Goal: Samba 4.3 should have a full witness implementation
- Some infrastructure requirements need to be resolved first

witness testing

- rpcclient witness command set
- smbtorture local.ndr.witness
  - Just tests correctness of the NDR marshalling/unmarshalling
- smbtorture rpc.witness
  - Test correctness of the DCE/RPC calls
- Fundamental problem: how to test a cluster ? How to test resource changes? How to test node failures ?
- Windows Failover Cluster Manager does resource changes with yet another DCE/RPC protocol

Sidetrack: clusapi

- Microsoft Cluster Management API
  - > 200 opcodes
  - > 600 pages protocol spec
  - Used by Microsoft Failover Cluster Manager
- purely DCE/RPC based interface (over ncacn_ip_tcp[seal])
- Samba now has IDL (for v3 of that protocol) and a torture test suite
- MS-CRMP
  Failover Cluster: Management API (ClusAPI) Protocol
- Some ideas to use this protocol as frontend for remote CTDB management

DCE/RPC requirements

- endpointmapper with ncacn_ip_tcp support
  - Available
- asynchronous DCE/RPC server
  - Currently two unfinished implementations:
    - David Disseldorp <ddiss@samba.org>
    - Stefan Metzmacher <metze@samba.org>
  - (also needed for MS-PAR and possibly other protocols)
- mgmt service (Remote DCE/RPC service management)
  - Two implementations available, none is published yet.

Relationship to SMB3 protocol

- Per share flag enables use of Witness Protocol
- MS-SMB2: “The specified share is present on a server configuration which provides monitoring of the availability of share through the Witness service specified in [MS-SWN]”
- SMB2 TREE_CONNECT Response Capability Flag: SMB2_SHARE_CAP_CLUSTER = 0x00000040
- Wintess support seems to be independent from SMB2_SHARE_CAP_SCALEOUT and SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY
- Currently for testing:
  - smbd:announce CLUSTER = yes

witnessd server

- Standalone binary, using new infrastructure invented for spoolssd
- Independent binary so any Samba server problem does not interfere with witness messaging
- Needs to register for at least 4 notification events (messaging)
- Configuration and possibly Server State store
- Very close integration with ctdb:
  - CTDB maintains all available cluster state information
  - CTDB already has mechanisms to communicate failures between the nodes
  - CTDB could easily reuse tickle-ack hooks for witness notifications

witness client

- Management tasks of witness server:
  - listing of active, connected clients
  - Manually move Clients to other nodes
  - Move share to other node
  - (similar to SmbWitnessClient PowerShell cmdlet)
- Allow third parties to benefit from witness infrastructure as a consumer of witness notifications:
  - CIFS Kernel module
  - smbclient
  - libsmbclient

Further reading

- Microsoft Protocol Documentation:
  - MS-SWN: Service Witness Protocol
  - MS-SMB2: Server Message Block (SMB) Protocol Versions 2 and 3
  - MS-CMRP: Failover Cluster Management Protocol
- SMB 2.x and SMB 3.0 Timeouts in Windows http://blogs.msdn.com/b/openspecification/archive/2013/03/27/smb-2-x-and-smb-3-0-timeouts-in-windows.aspx
- Samba Wiki https://wiki.samba.org/index.php/Samba3/SMB2#Witness_Notification_Protocol

Questions and answers

- Mail gd@samba.org
- gd at #samba-technical on irc.freenode.net
- https://git.samba.org/?p=gd/samba/.git;a=shortlog;h=refs/heads/master-witness-ok

Thank you for your attention!