SDC 20
BY Developers FOR Developers
Storage Developer Conference
September 22-23, 2020

Please kindly go somewhere else

Methods and strategies to move SMB2 clients' connections to other cluster nodes non-disruptively

Rafal Szczesniak
Jeremy Hitt

Dell/EMC – Isilon Storage Division

Even highly available storage needs maintenance

- Storage, especially enterprise class, is expected to be always on
- We still need (from time to time)
  - Fix hardware failures
  - Upgrade software
- There is never a good time for system restart
- In scale-out clusters, let's try and migrate the connections and then restart the node
- If we do it right, hopefully no one will have noticed…

Existing methods

The DNS

Oplocks and leases

Disconnecting

Reconnecting

Takeaways and nice-to-haves

Existing methods in SMB3

- Witness service
  - Considered a part of Continuously Available (CA) shares scenario
  - SMB2_SHARE_CAP_CLUSTER (TreeConnect) should start the client
  - Not very common – only Windows implements it
- Share Redirection (TreeConnect error response)
  - SMB 3.1.1 feature
  - Moves the connection closer to where data lives
  - Can be disruptive – calling application is involved on the client (MS-SMB2)
- They both only work at the share level

What if you don’t have them?

- Many SMB3 implementations don’t support continuous availability
- There’s still plenty of SMB2 clients out there
- SMB1… well, we just don’t care anymore

What if you don’t have them? (contd.)

- We’d really like to tell the clients where to go, but there’s no good way of doing that in SMB2
- We want to make the transition non-disruptive
- We certainly don’t want to lose any data
- We don’t want users asking administrator “What just happened?”

Existing methods

The DNS

Oplocks and leases

Disconnecting

Reconnecting

Takeaways and nice-to-haves

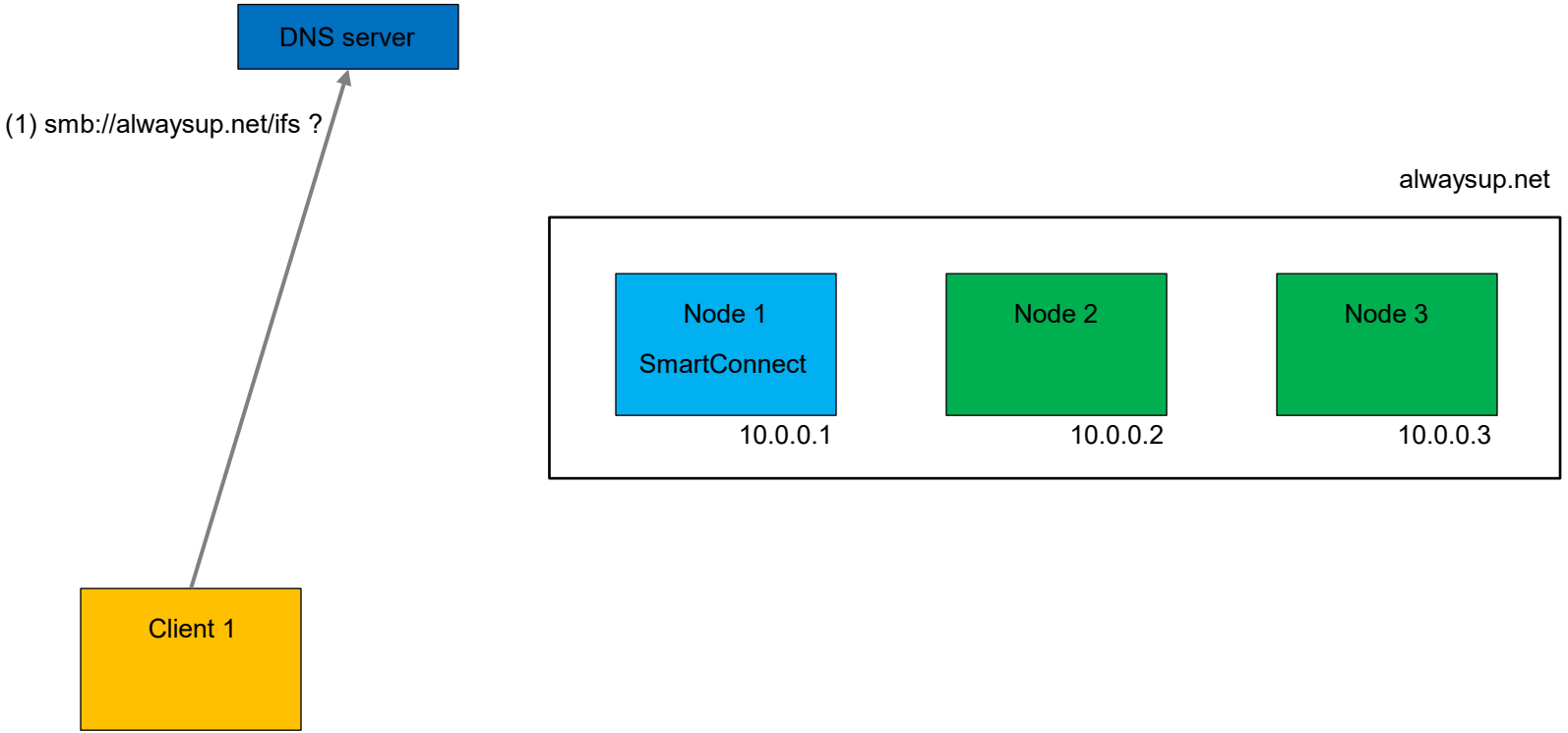
Let’s start with the DNS

- It all begins with the name resolution
- If connected directly to a node’s IP address, we’d have to resort to the CA-level methods

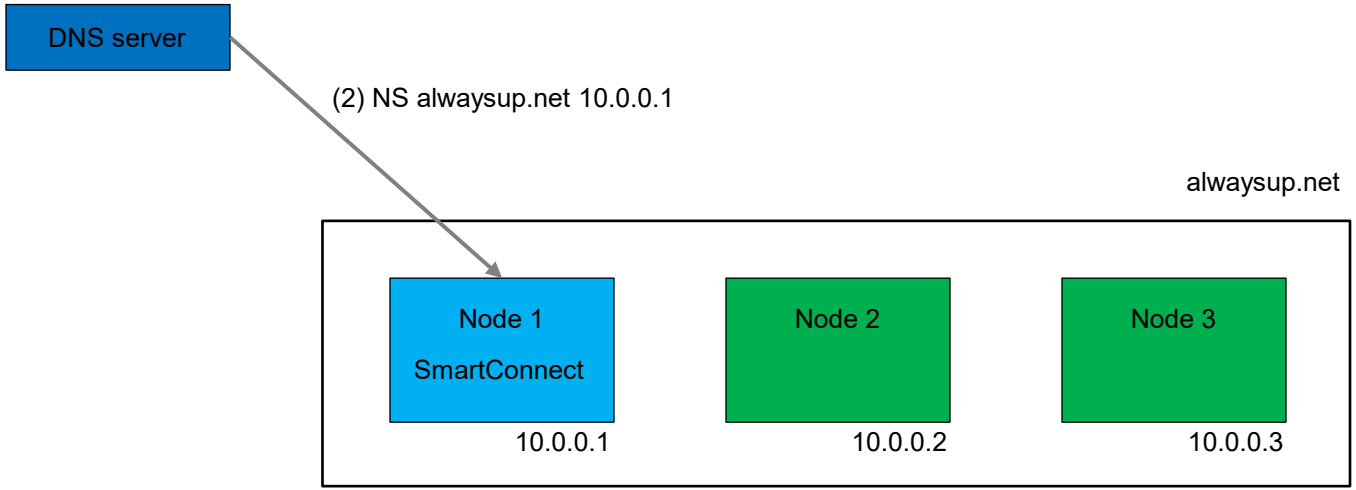
The role of DNS-resolving node

- OneFS clusters run a service called SmartConnect (price tag: 1 IP address)
- One node assumes the duty of resolving smb://cluster.your.domain requests
- May respond with a different IP every time for load balancing
- Can also stop handing out given node's addresses if requested
  - New clients will not reach it unless they know the IP

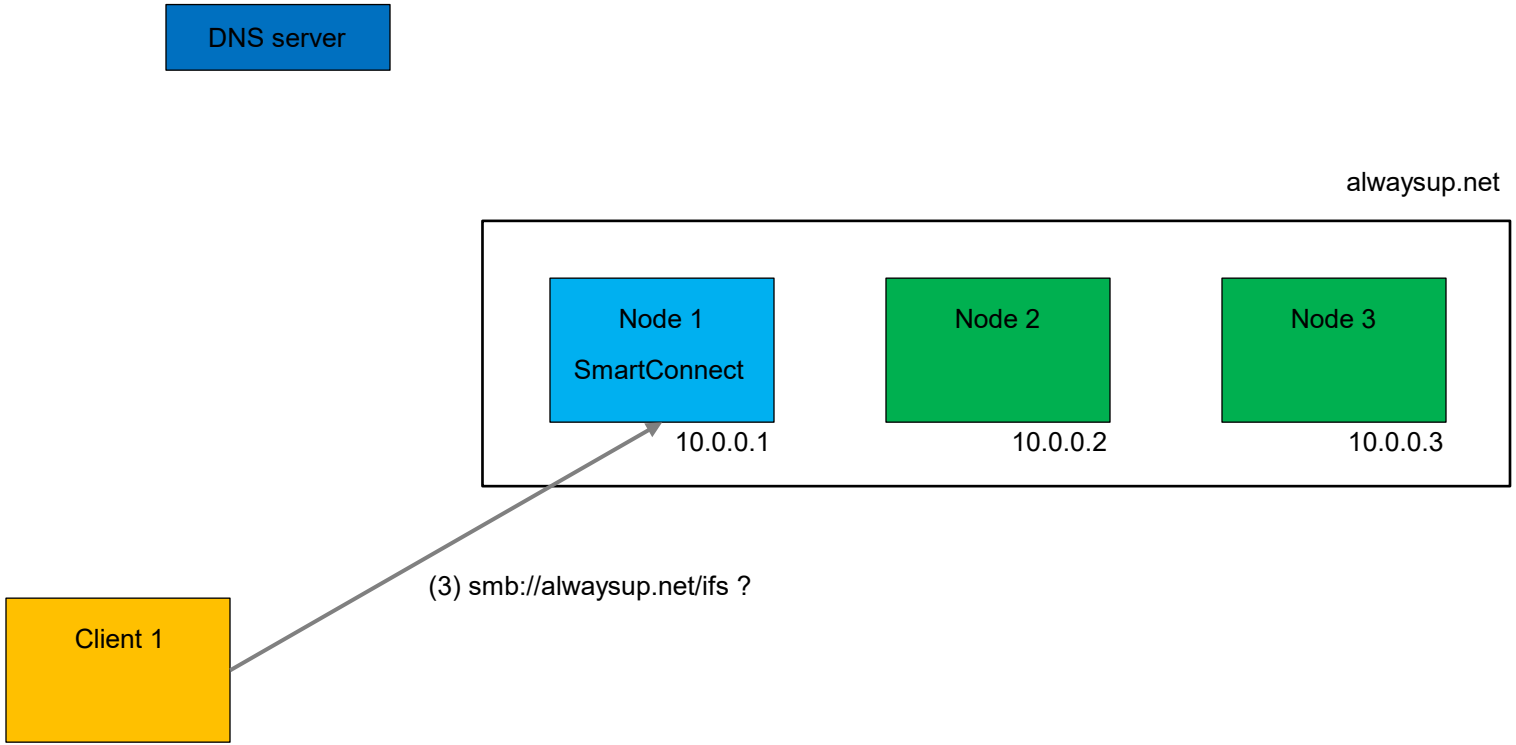
The role of DNS-resolving node



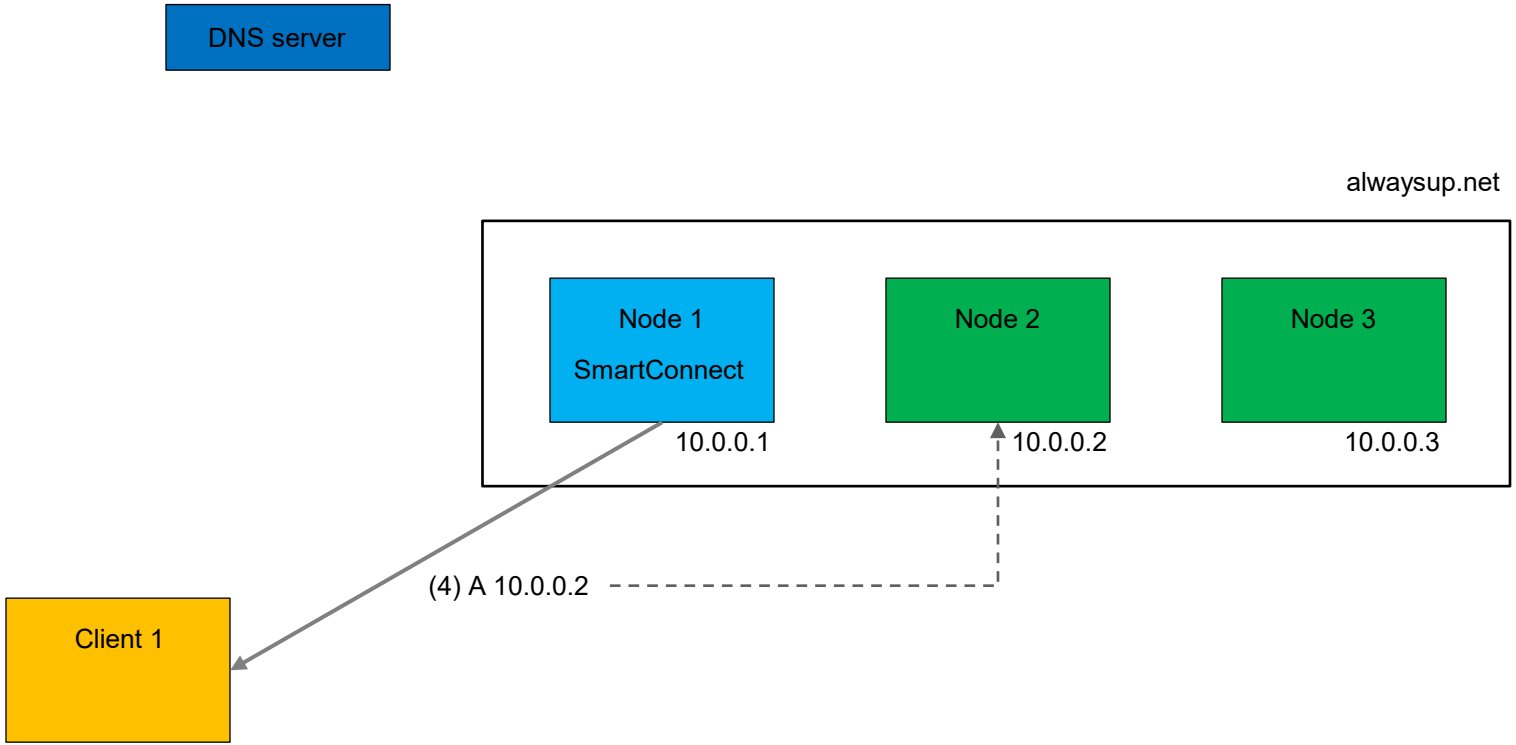
The role of DNS-resolving node



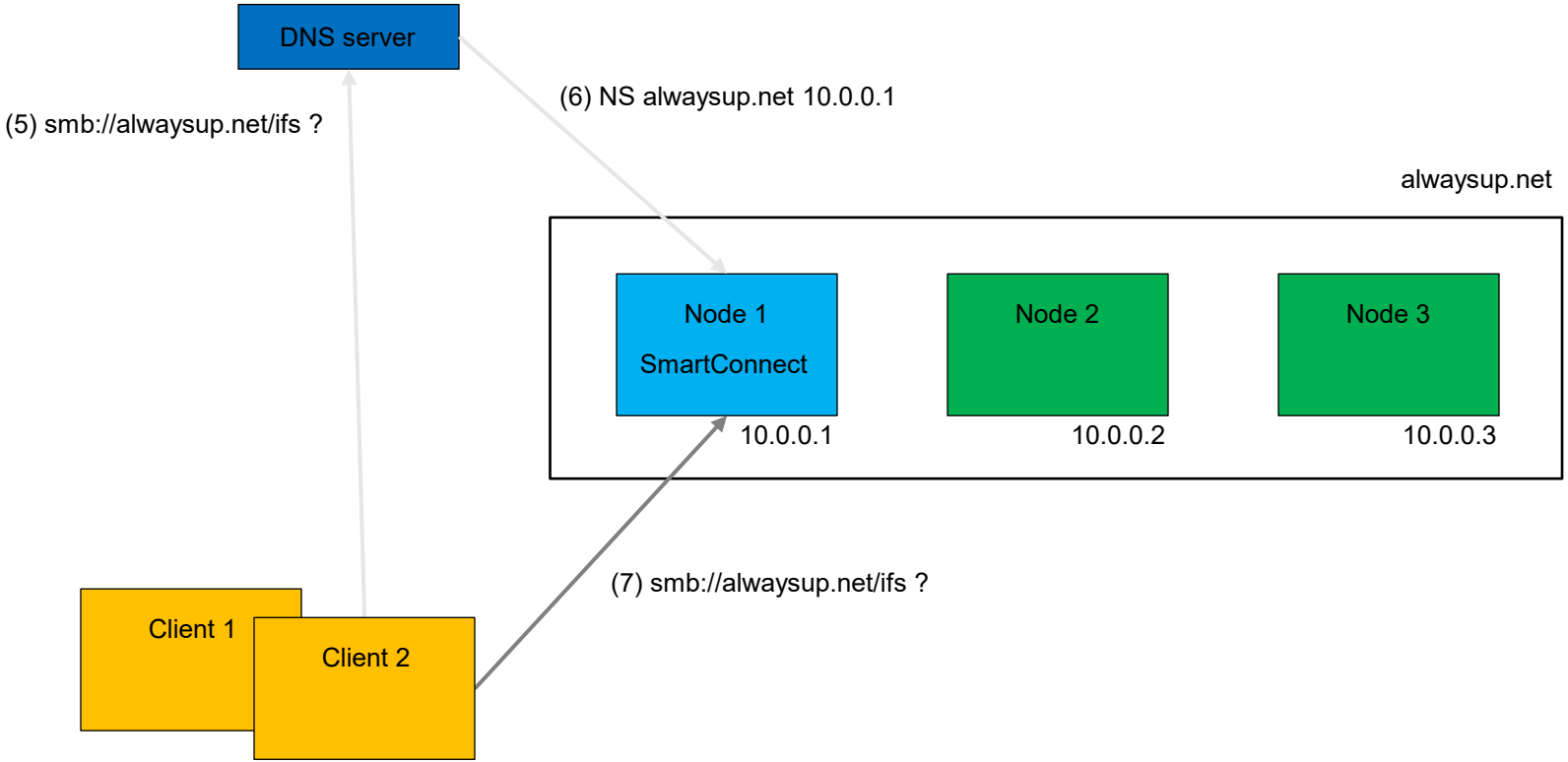
The role of DNS-resolving node



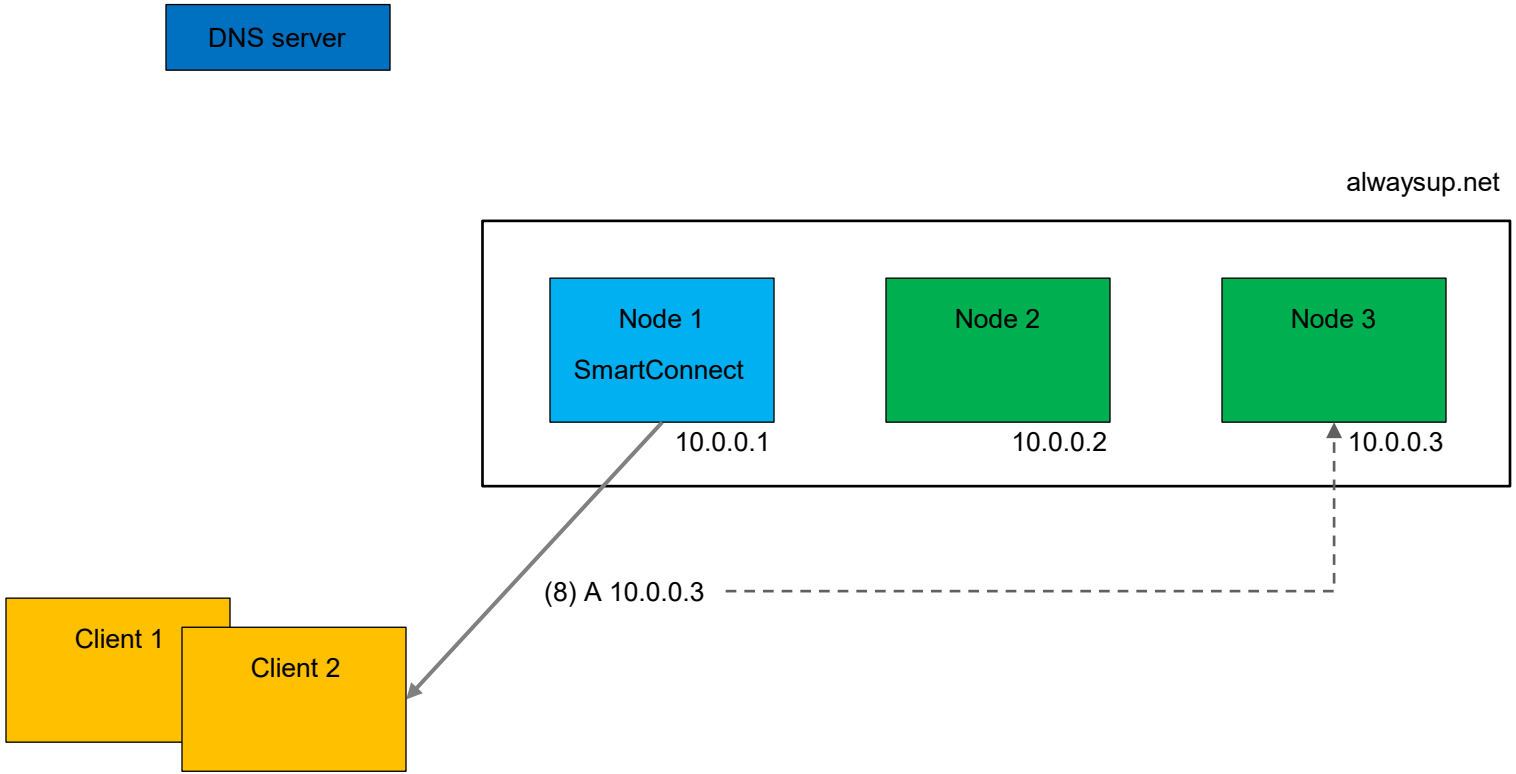
The role of DNS-resolving node



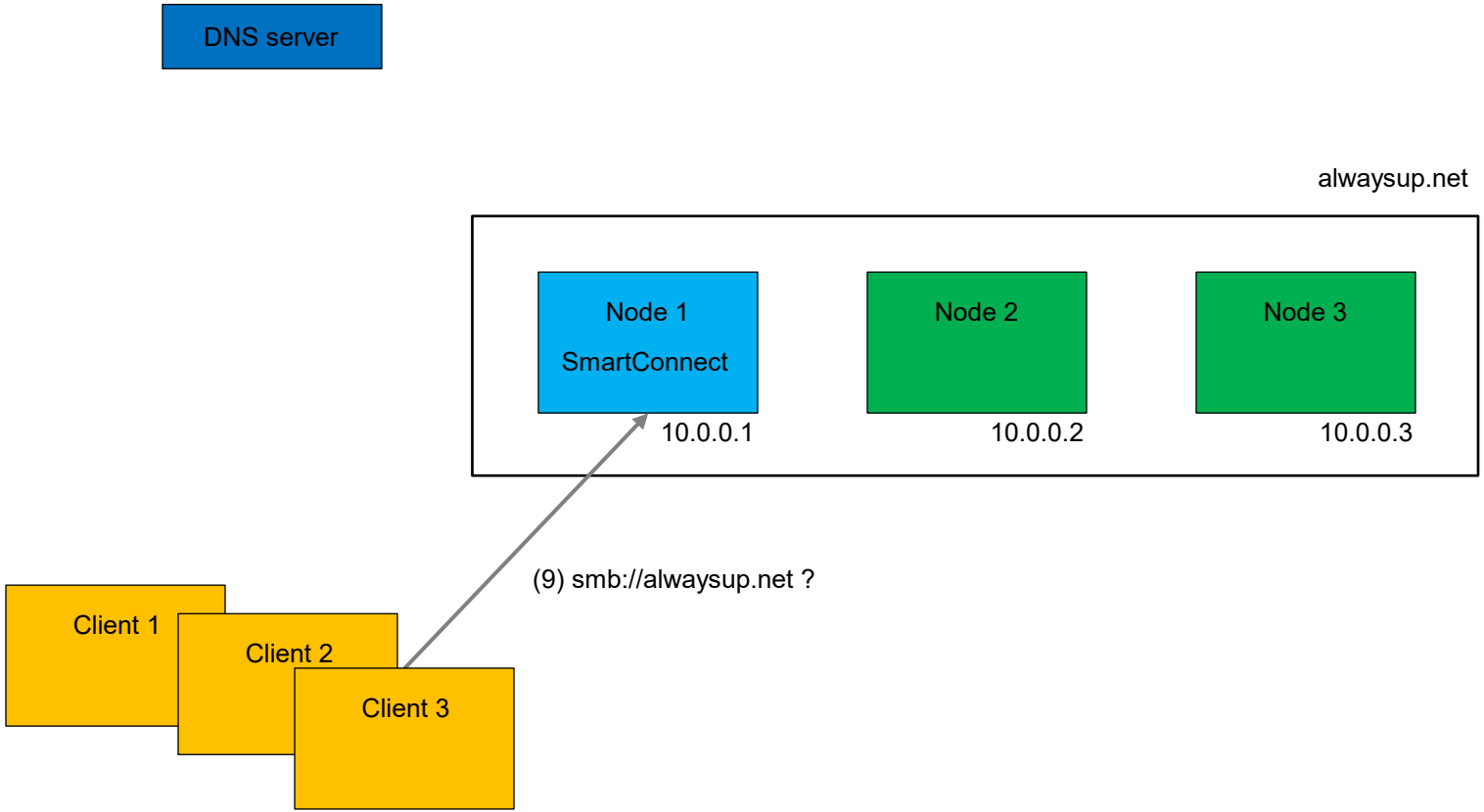
The role of DNS-resolving node



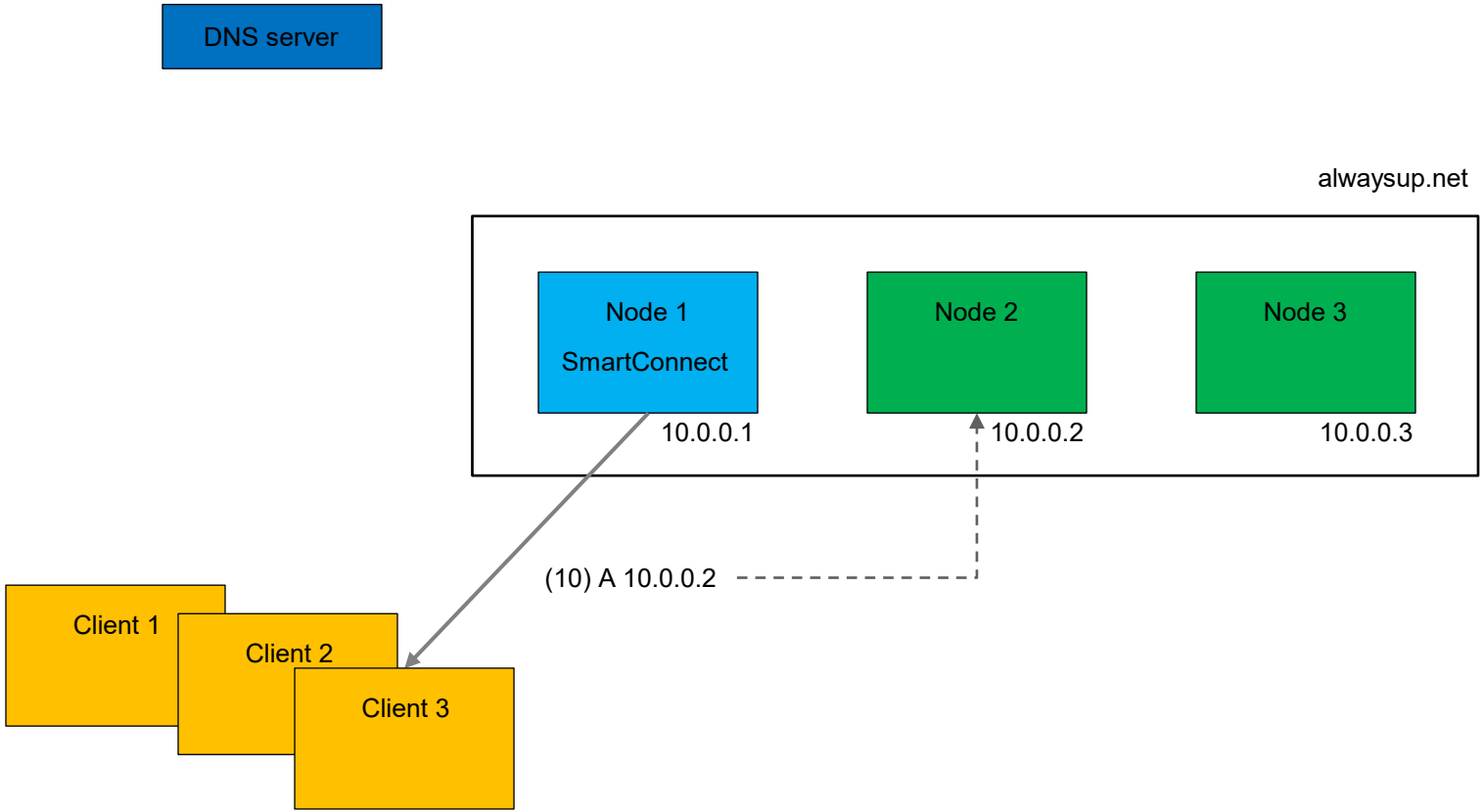
The role of DNS-resolving node



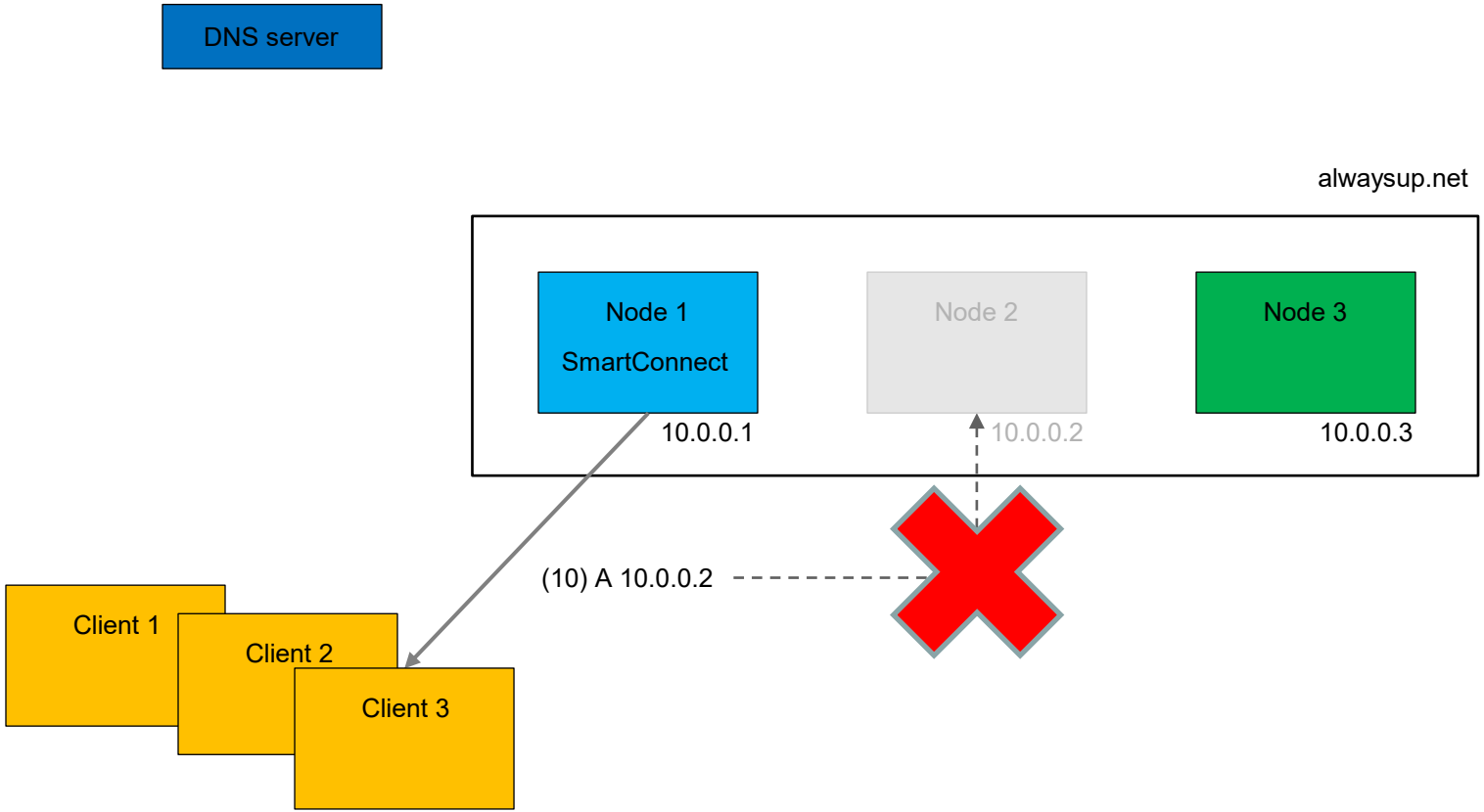
The role of DNS-resolving node



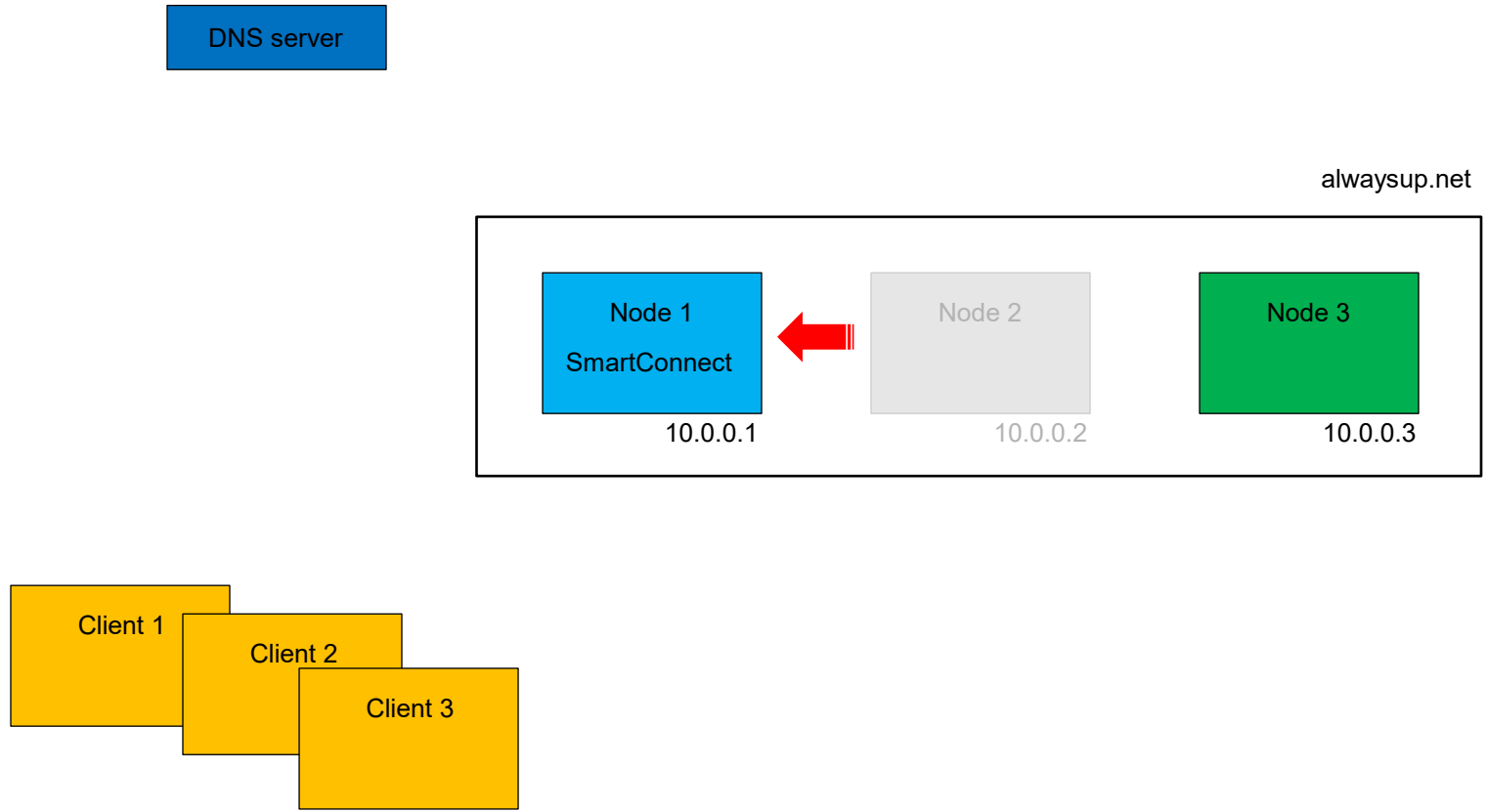
The role of DNS-resolving node



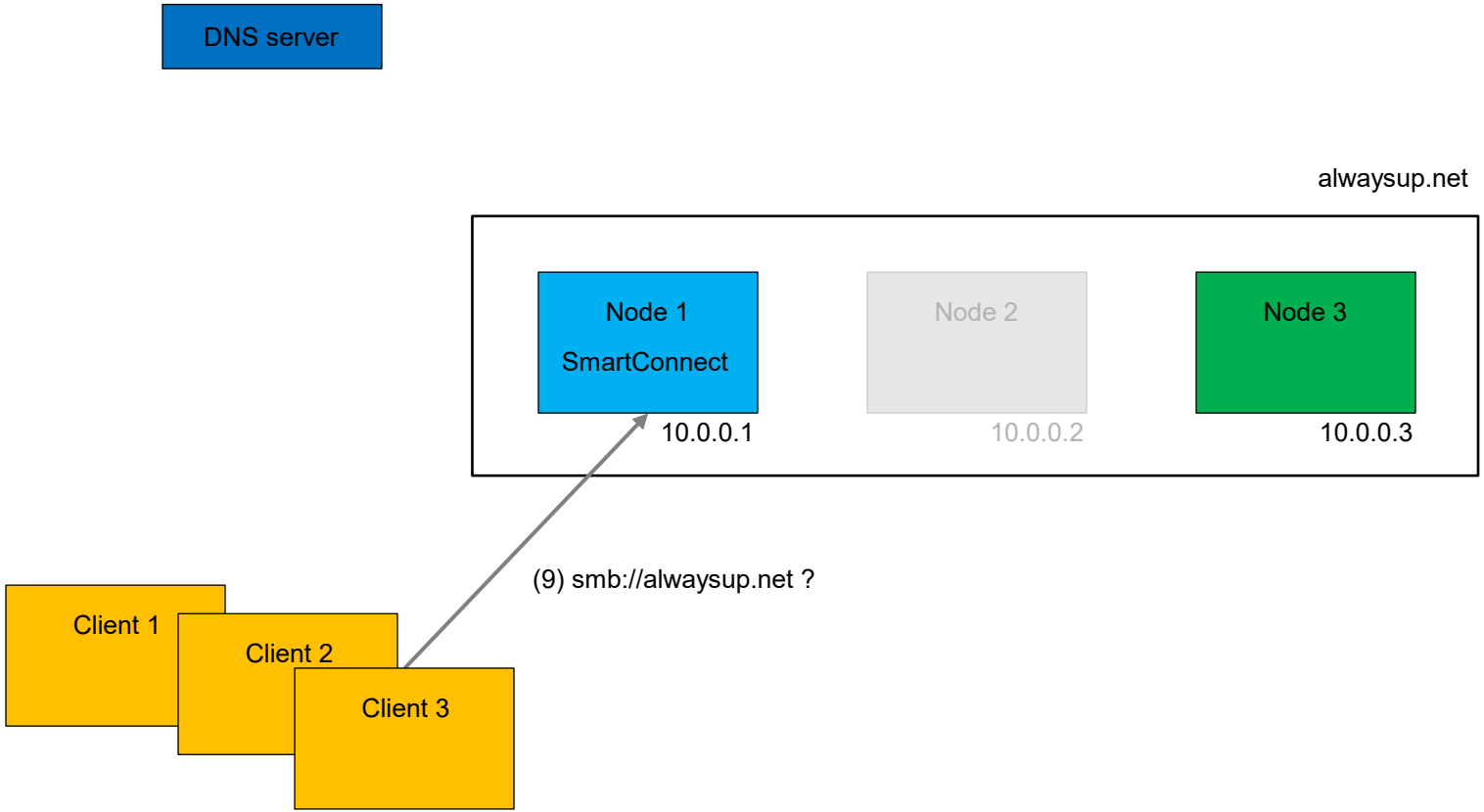
The role of DNS-resolving node



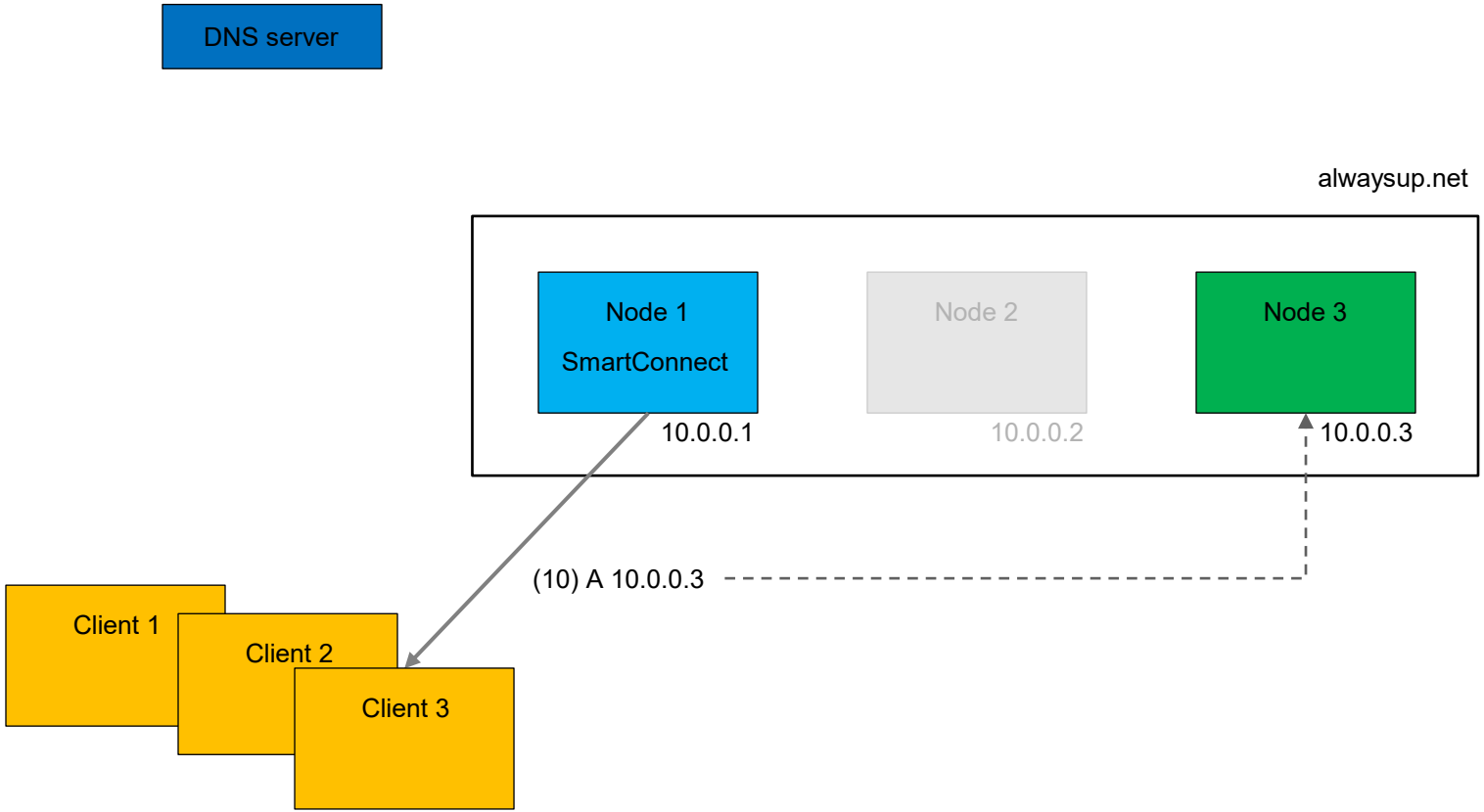
The role of DNS-resolving node



The role of DNS-resolving node



The role of DNS-resolving node



Existing methods

The DNS

Oplocks and leases

Disconnecting

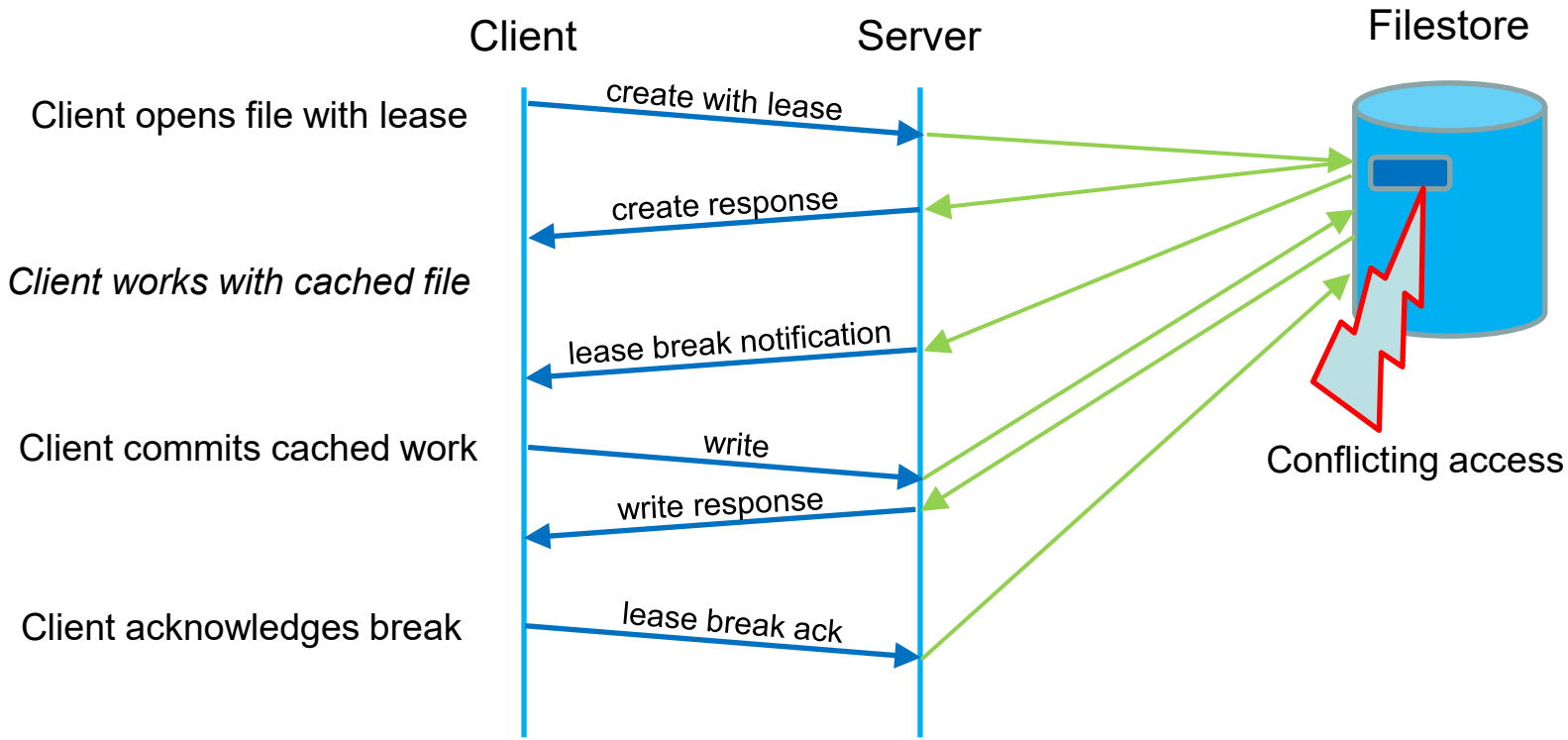
Reconnecting

Takeaways and nice-to-haves

Do not lose data – oplocks and leases

- We know we're going to disconnect the client, so we don't want them to cache
- We need to be certain that clients write the data directly to the node
- Any data loss is not a non-disruptive experience
- Don't allow clients to take out new oplocks/leases

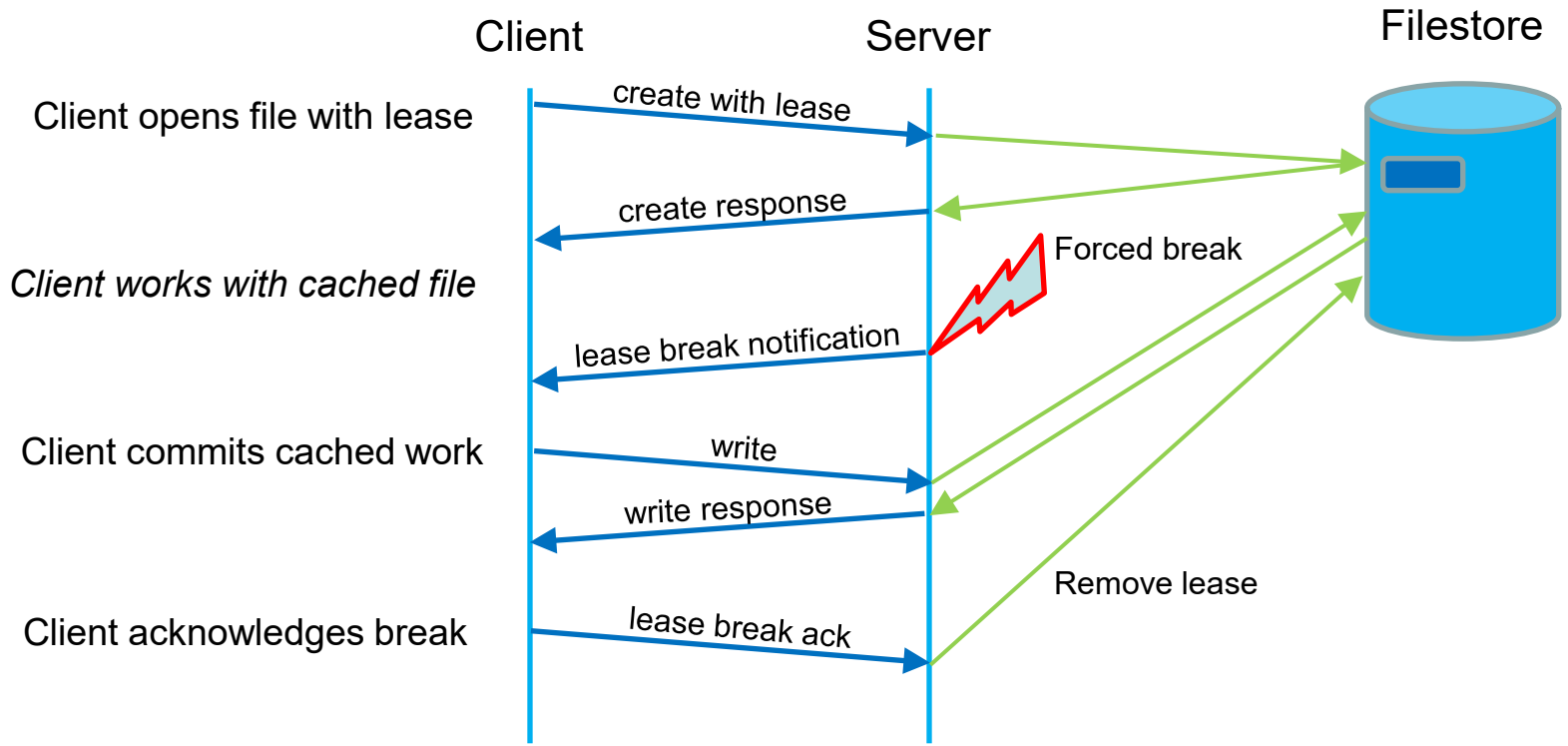
Normal lease operation



Self-induced lease breaks

- Oplocks/Leases are broken and downgraded when someone else is accessing the same file
- Clients can no longer pretend they have exclusive access
- This time it's the server who breaks and downgrades

Draining lease operation



Existing methods

The DNS

Oplocks and leases

Disconnecting

Reconnecting

Takeaways and nice-to-haves

Clients can reconnect

- Virtually all clients have the ability to reconnect when the connection drops
- Legacy from the days when networks were less reliable?
- In many cases, the end user doesn't even notice
- Ideally, when reconnecting, the client can connect to another node

What is the right moment?

- **Not too early (and not too late)**
- SMB2 connection has to be fully established:
  - Negotiated
  - Authenticated
  - Share connected

What is the right moment? (contd.)

Disconnection after receiving Negotiate request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Windows 7	Occasional errors (> 80%)	40%	No success (0%)
Windows 8	No errors (100%)	60%	80%
Windows 8.1	No errors (100%)	No errors (100%)	80%
Windows 10	Occasional errors (> 80%)	No errors (100%)	No errors (100%)

1. Windows 8 and 8.1 are slightly slower, Windows 10 does a good job.

What is the right moment? (contd.)

Disconnection after receiving Negotiate request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Linux 4.15.0 (Ubuntu 16)	No success (0%)	No success (0%)	No success (0%)
Linux 5.4.0 (Ubuntu 20)	No success (0%)	No success (0%)	No success (0%)
macOS 10.14.16 (Mojave)	No success (0%)	No success (0%)	No success (0%)
macOS 10.15.5 (Catalina)	No success (0%)	No success (0%)	No success (0%)

1. Linux errors: “Host is down (112) / No such file or directory (2)”
2. macOS sends KeepAlive requests (0x0d) when 10 s delay is reached.

What is the right moment? (contd.)

Disconnection after receiving SessionSetup request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Windows 7	No errors (100%)	No errors (100%)	No errors (100%)
Windows 8	No errors (100%)	No errors (100%)	No errors (100%)
Windows 8.1	No errors (100%)	No errors (100%)	No errors (100%)
Windows 10	No errors (100%)	No errors (100%)	No errors (100%)

1. In all cases, Windows 8.1 reacts the fastest with Windows 10 closely following.

What is the right moment? (contd.)

Disconnection after receiving SessionSetup request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Linux 4.15.0 (Ubuntu 16)	No success (0%)	No success (0%)	No success (0%)
Linux 5.4.0 (Ubuntu 20)	No success (0%)	No success (0%)	No success (0%)
macOS 10.14.16 (Mojave)	No success (0%)	No success (0%)	No success (0%)
macOS 10.15.5 (Catalina)	No success (0%)	No success (0%)	No success (0%)

1. Linux errors: “Resource temporarily unavailable (11) / No such file or directory (2)”
2. macOS sends KeepAlive requests (0x0d) when 10 s delay is reached.

What is the right moment? (contd.)

Disconnection after receiving TreeConnect request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Windows 7	No errors (100%)	No errors (100%)	No errors (100%)
Windows 8	No errors (100%)	No errors (100%)	No errors (100%)
Windows 8.1	No errors (100%)	No errors (100%)	No errors (100%)
Windows 10	No errors (100%)	No errors (100%)	No errors (100%)

1. In all cases, Windows 8.1 reacts the fastest with Windows 10 closely following.

What is the right moment? (contd.)

Disconnection after receiving TreeConnect request – success rate at reconnecting

Delay (ms)	1000	5000	10000
Linux 4.15.0 (Ubuntu 16)	No success (0%)	No success (0%)	No success (0%)
Linux 5.4.0 (Ubuntu 20)	No success (0%)	No success (0%)	No success (0%)
macOS 10.14.16 (Mojave)	Never stops trying	Never stops trying	Never stops trying
macOS 10.15.5 (Catalina)	Never stops trying	Never stops trying	Never stops trying

1. Linux errors: “Resource temporarily unavailable (11) / No such file or directory (2)”
2. macOS will keep trying even after the user cancels connection (no longer interested).

Extend and pretend

- If connection drops after share has been connected, it can be completely rebuilt
- Even after a few files have been opened clients can handle it
- It helps a little to slow down artificially
  - If responses are delayed the client remains patient but don't do much
  - This gives us more time to "clean up"

Don’t wait too long

- Larger data transfers are hard to break and get away with that
- It helps to wait until there’s as few reads and writes as possible (ideally none, even for a moment)
- We have to accept that the fortunate moment may never come

Existing methods

The DNS

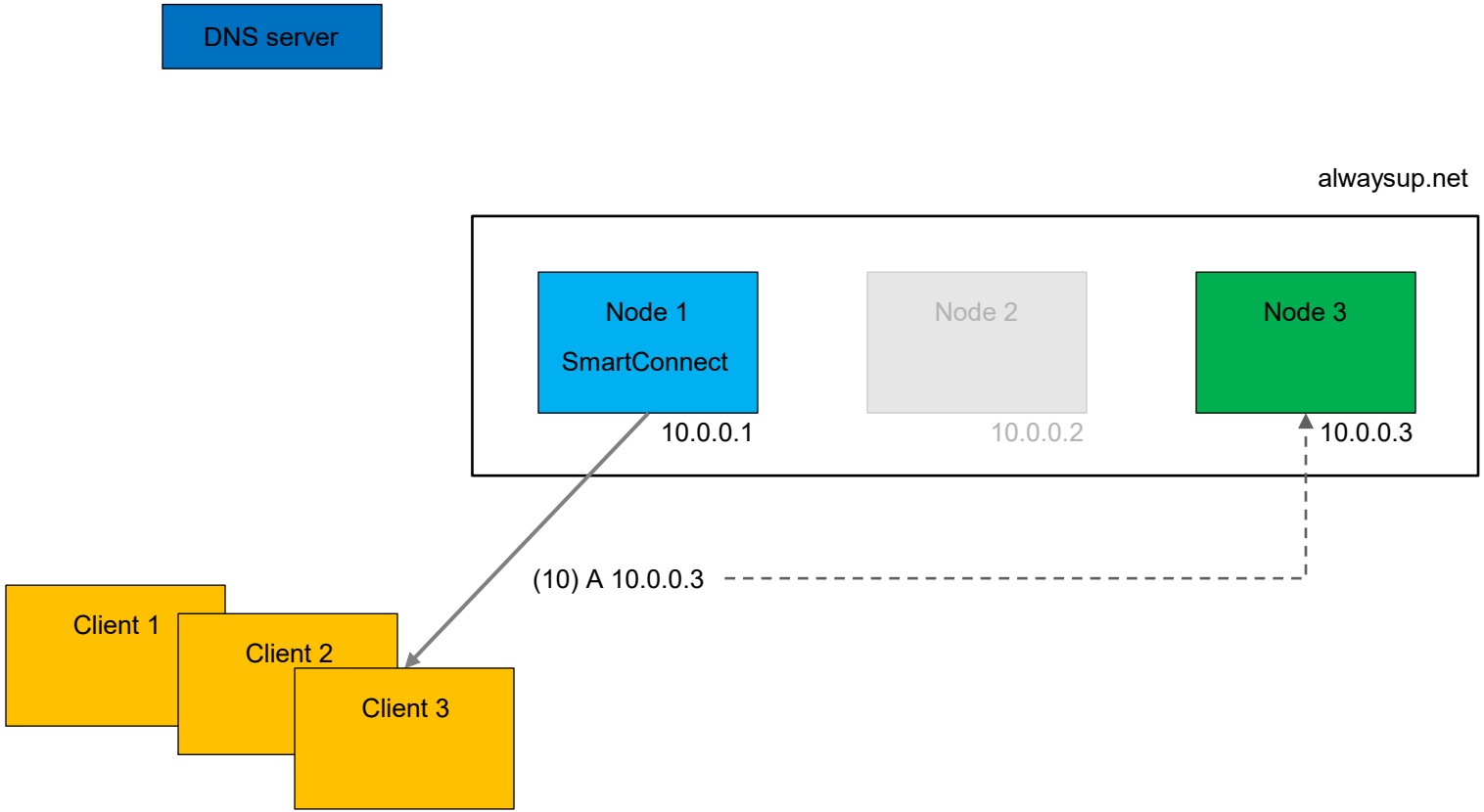
Oplocks and leases

Disconnecting

Reconnecting

Takeaways and nice-to-haves

DNS and reconnections



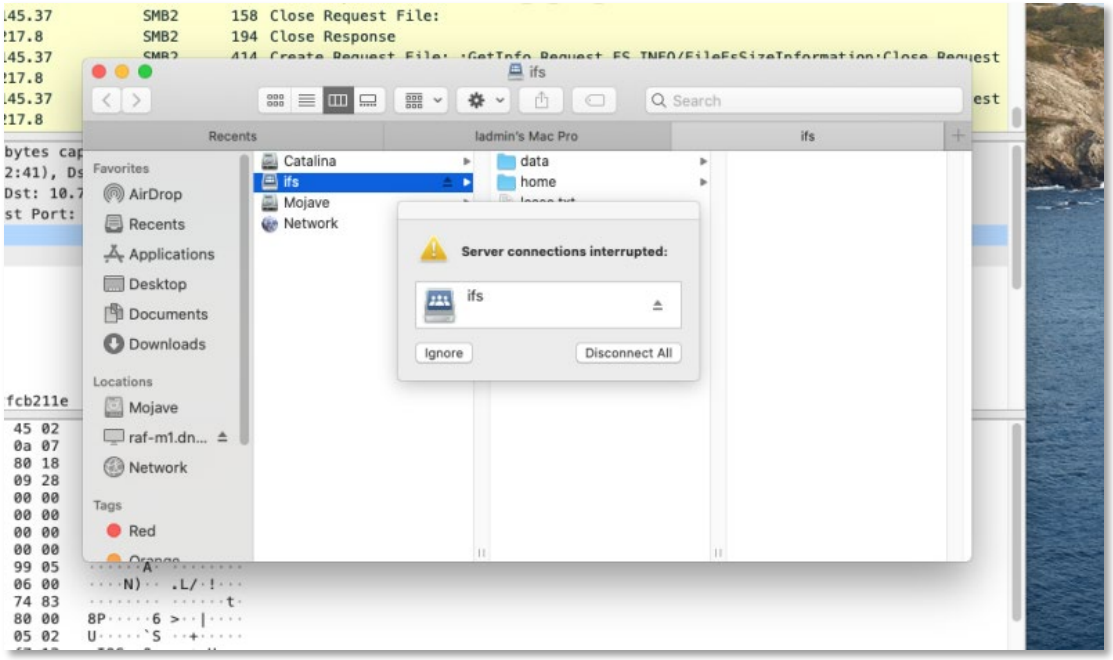
DNS and reconnections (contd.)

- Ideally, we want the client to DNS-resolve again before reconnecting
  - If another node's IP is given, the client will "go away"
  - Windows does a really good job here
- DNS-caching clients may require more patience
- It may take several rounds of disconnections

Gloves off

- Some clients just won't go if they can keep connecting over TCP
  - Likely because a resolved and connected (at least once) server is considered alive
- We have to stop accepting at port 445 to drop the stubborn ones
- Some versions of Linux do get the message (and DNS-resolve)
- macOS does not

Gloves off (contd.)



Ignoring the disconnection **can be a workaround** (not completely non-disruptive, though).

Existing methods

The DNS

Oplocks and leases

Disconnecting

Reconnecting

Takeaways and nice-to-haves

Key takeaways

- If your connection keeps dropping, the server may be trying to get rid of you – DNS-resolve and try again!
- Possible hints:
  - Your connection has slowed down (longer response times)
  - You requested oplocks/leases but you didn't get them
  - …or perhaps your server is being passive-aggressive? ☺

Wishlist

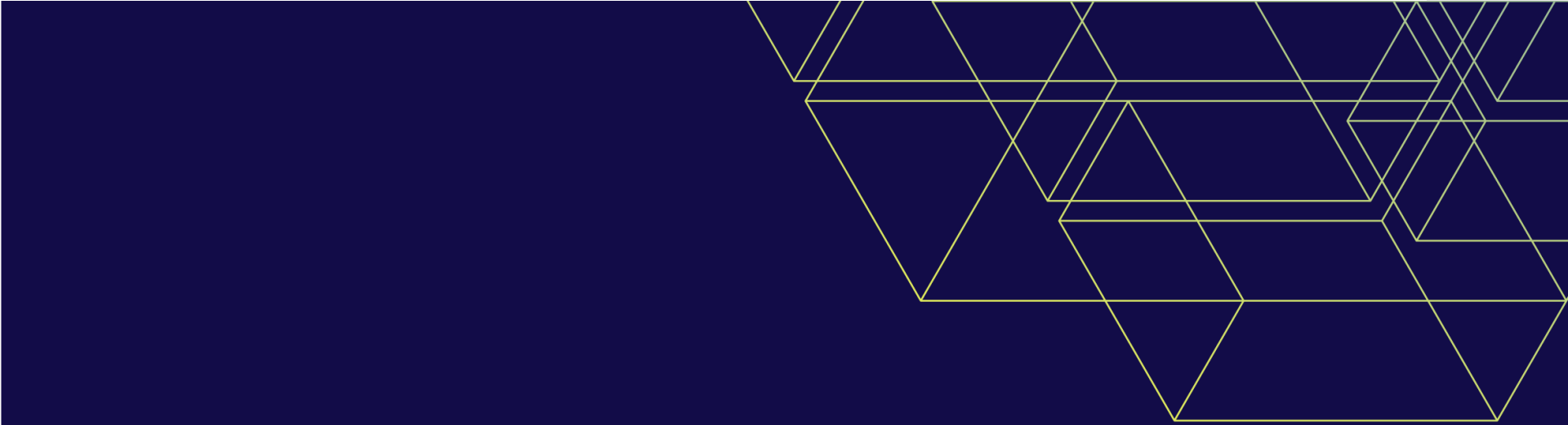
- Could Witness be made a Negotiate-level capability (not per-share)?
- STATUS_REDIRECT_DUE_TO_SHUTDOWN?

Thank you!

Questions?

Rafal Szczesniak
rafal.szczesniak@dell.com
@emc.com

Jeremy Hitt
jeremy.hitt@dell.com
@emc.com



Please take a moment
to rate this session.

Your feedback matters to us.