Implementing Witness service for various cluster failover scenarios

Rafal Szczesniak
EMC/Isilon

Long time ago vs. now

- SMB1 – no high availability at all

Long time ago vs. now

- SMB1 – no high availability at all
- SMB2 – durable and resilient handles (file opens)

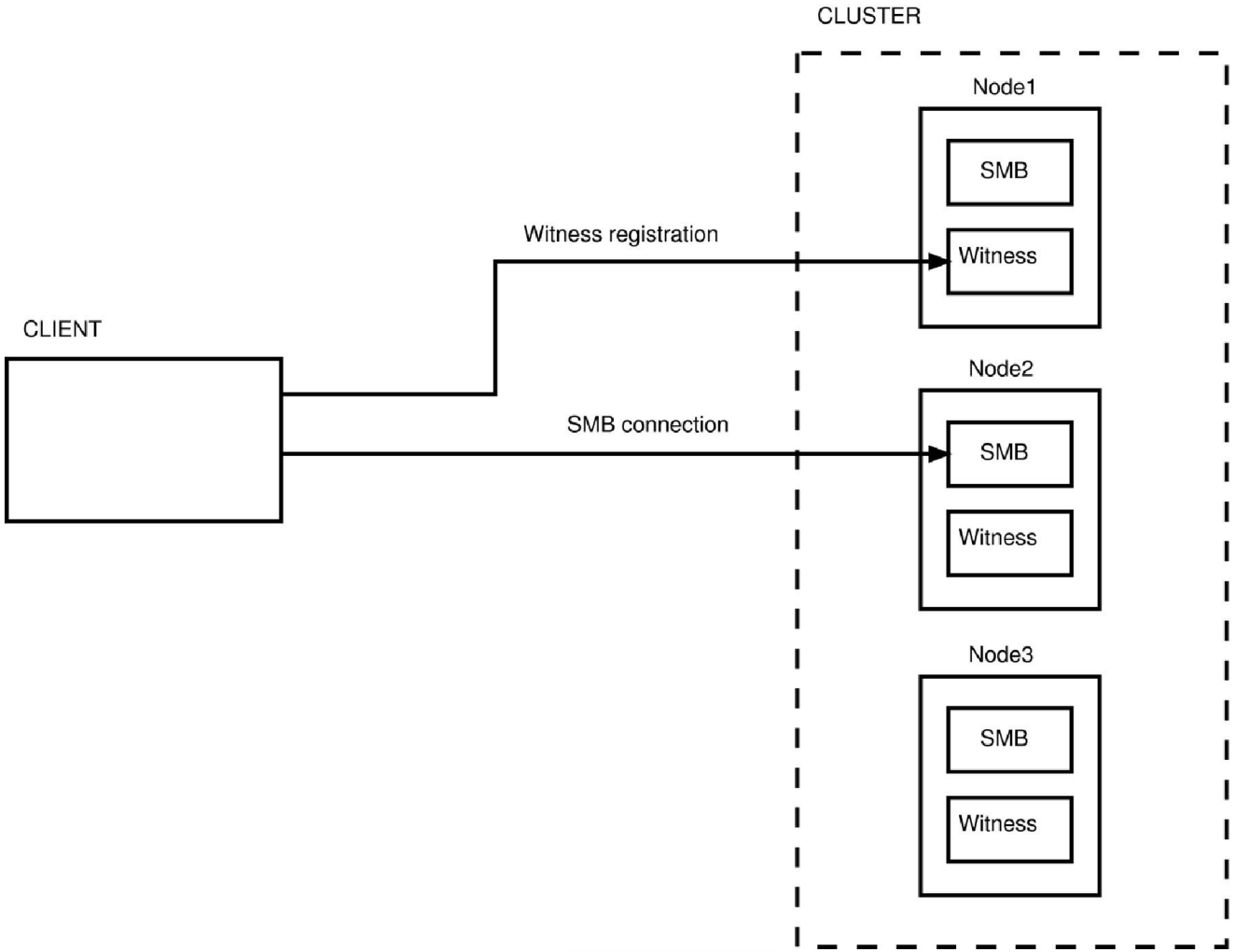
Long time ago vs. now

- SMB1 – no high availability at all
- SMB2 – durable and resilient handles (file opens)
- SMB3 – persistent handles, multi-channel and Witness

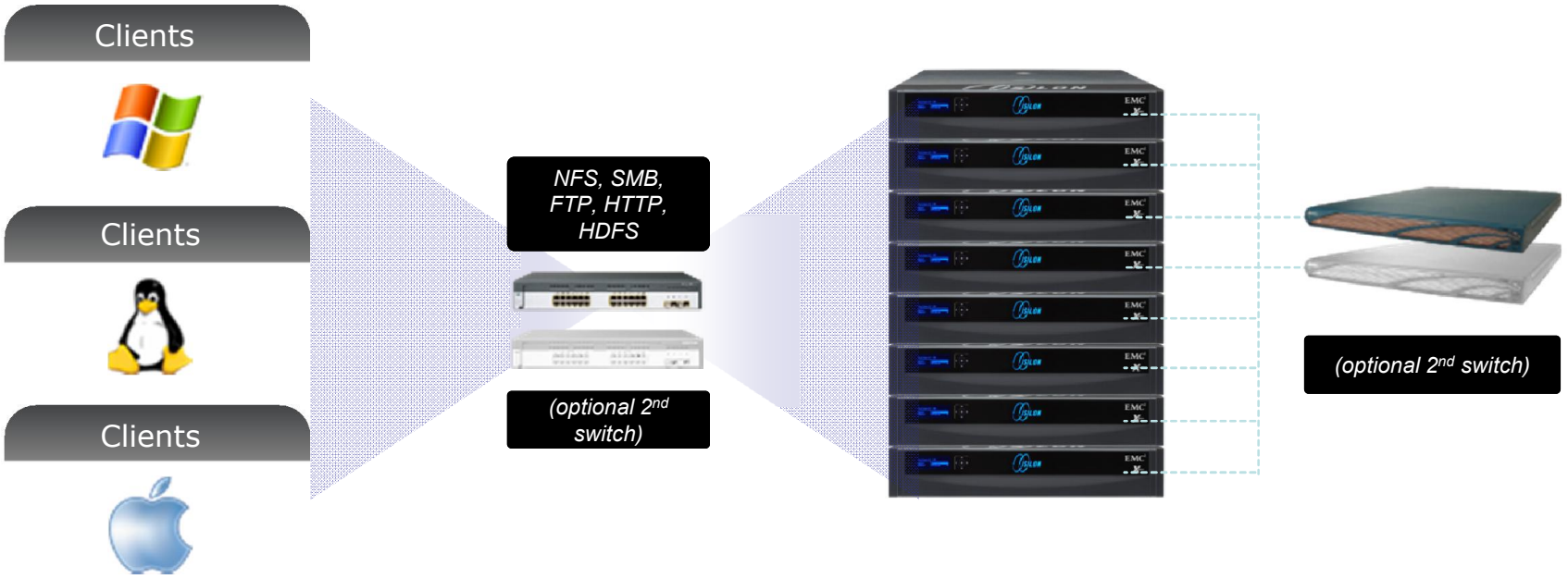
What is Witness?

- DCE/RPC interface (see [MS-SWN])
- Service providing early detection of connection failures instead of relying on TCP timeouts
- Means of (partial) control over client connections

What is Witness?

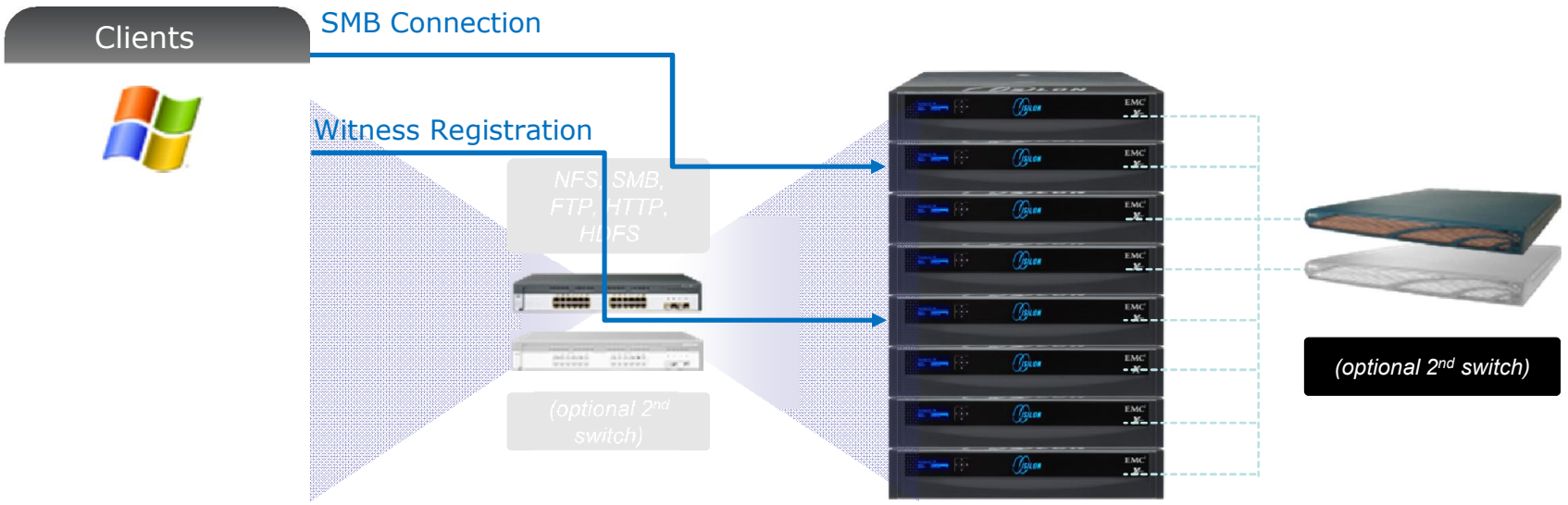


OneFS cluster



Client/Application Layer	Ethernet Layer	Isilon IQ Storage Layer	Intracluster Communication Infiniband

Witness Service in OneFS cluster



Client/Application Layer	Ethernet Layer	Isilon IQ Storage Layer	Intracluster Communication Infiniband

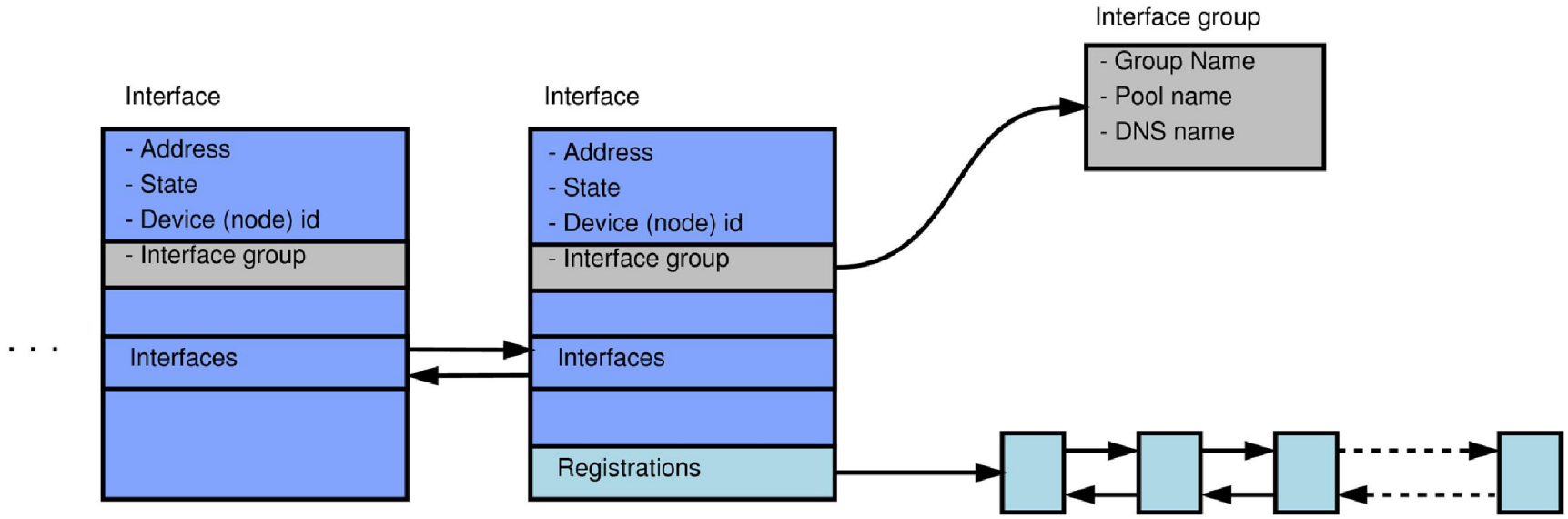
Interfaces and Groups

- Interface group as an abstraction of cluster nodes' network interfaces
- Usually the same as OneFS *address pool*
- Separate groups for separate OneFS *Access Zones*

Caching the state of interfaces

- Requesting the interface information from the system all the time can be expensive
- The interface state does not change so often
- We can cache the information for as long as we need it

Caching the state of interfaces



Caching on-demand

- The internal list of interfaces is propagated when needed
- The number of interfaces can be substantial, especially in a cluster with multiple Access Zones
- Updating a large cache could be expensive too, so it's easier to keep track of only those interfaces the clients ask about

Resource monitor

- Thin layer providing access to the cluster “resources”
- The only resources monitored (at the moment): *Interface*, *Interface Group*
- Allows querying the current information
- Allows subscribing for events and unsubscribing when the server is no longer interested in updates

What does the availability mean?

- Network interface failures

What does the availability mean?

- Network interface failures
- Server process crashes or deadlocks

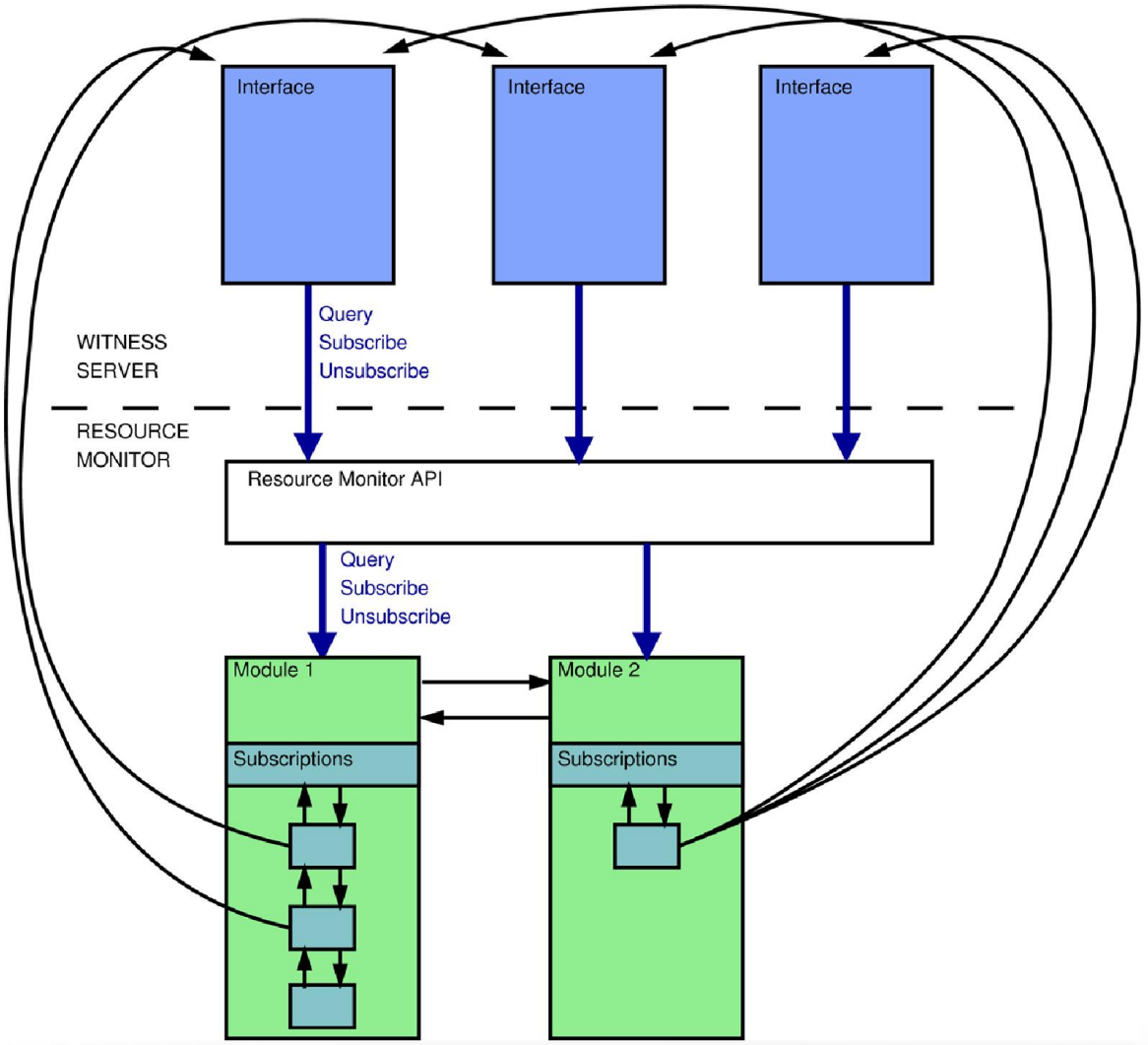
What does the availability mean?

- Network interface failures
- Server process crashes or deadlocks
- System crashes

Resource monitor modules and events

- Individual modules can keep track of all sorts of things independently
- Subscribing certain (or any) changes enables the module to submit events to *Interface* or *Interface Group*
- Witness server has the authority to filter the events and make its own decisions on how the clients should be notified

Resource monitor modules



Resource events

- Virtually any change happening to a subscribed resource can generate an event
- Examples of events to watch for:
  - *Interface* state change to unavailable
  - New interface added to an *Interface Group*
- Submitted events are “pre-treated” by the server before they are used to generate client notifications

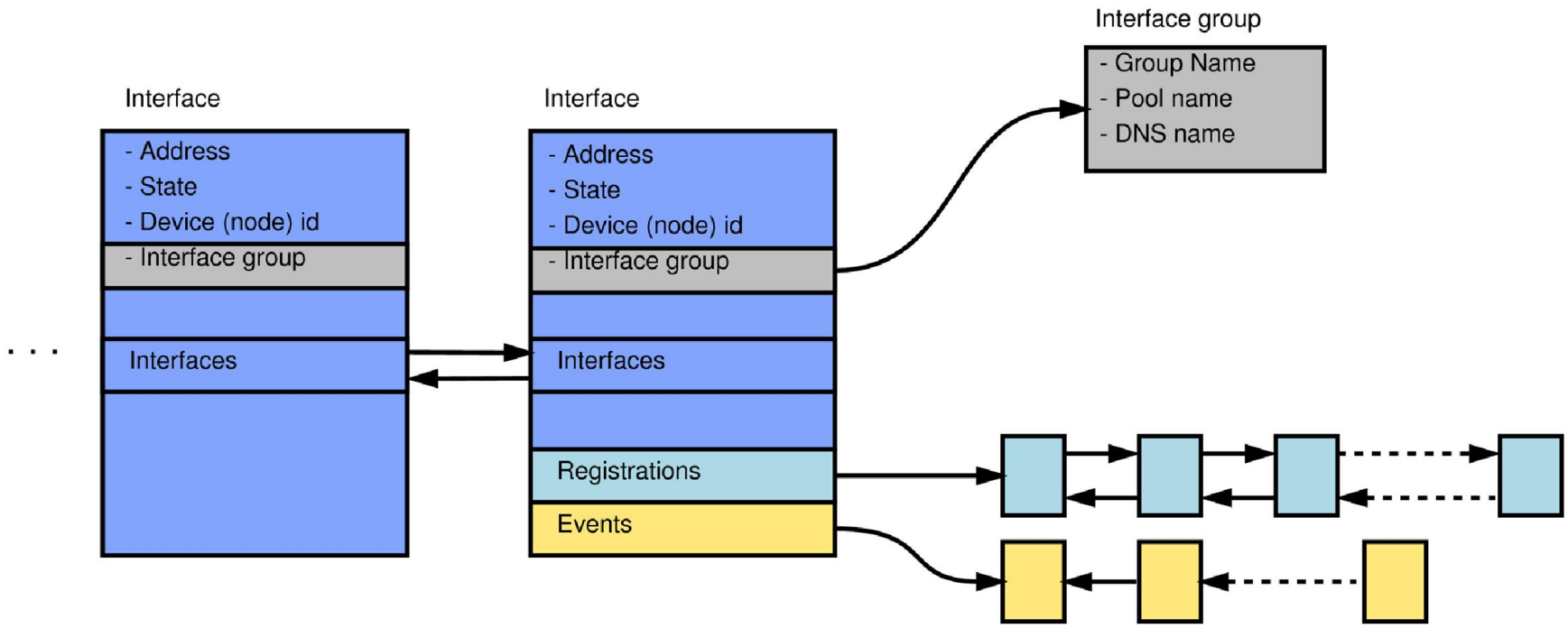
Resource events (contd.)

- Modules have a large degree of freedom in what can cause an event submission
- The server has the authority to say which events will turn into the actual notifications

Resource event

- What does it include?
  - Module Id
  - Type of event (*changed/added/removed*)
  - Resource
  - Destination (optional, if the module has any suggestions)

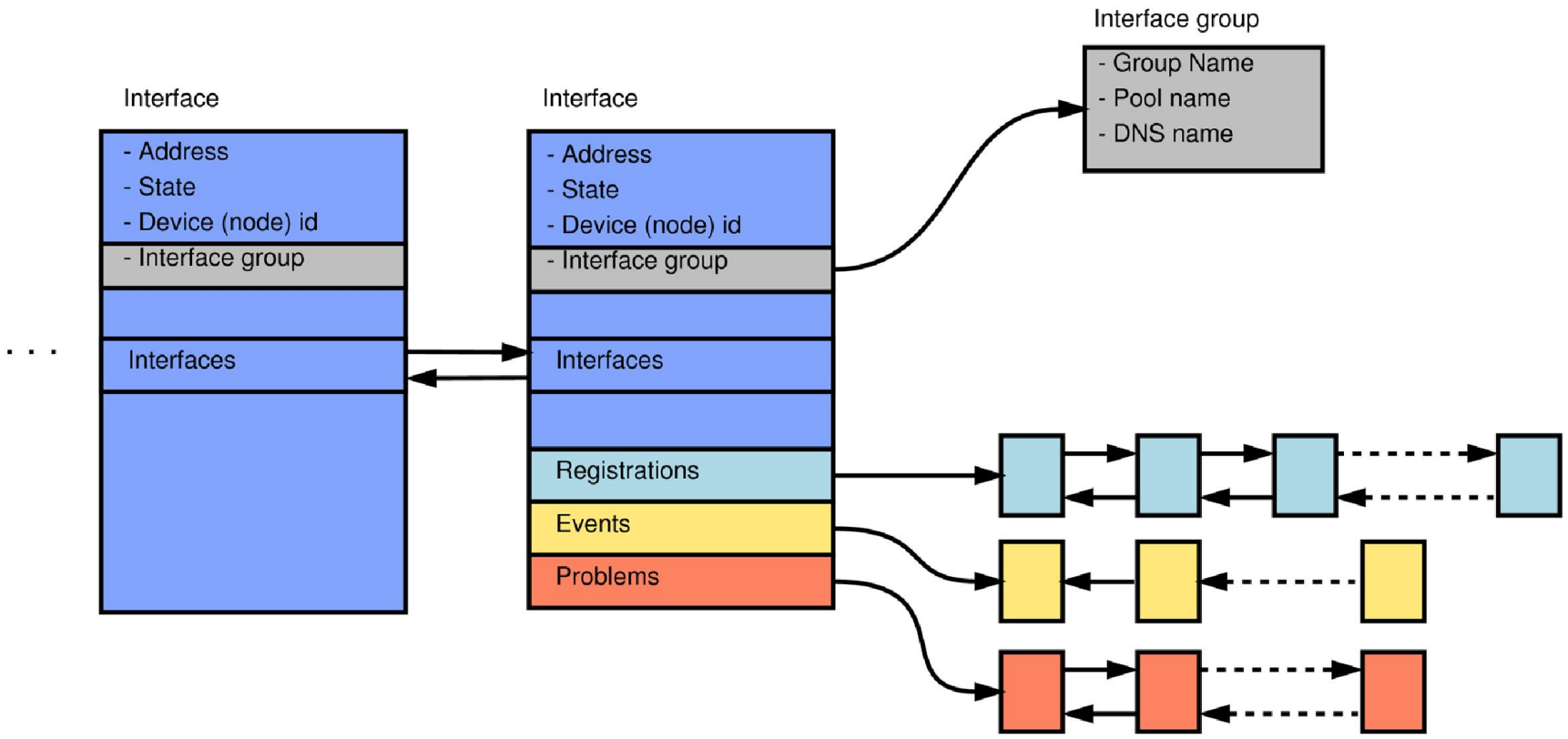
Interface events queue



Keeping track of the availability

- Multiple different modules look at different aspects of availability
- We need all of them to give us a “go” in order to consider an *Interface* available
- Witness server updates a list of *Problems* for each *Interface* as “go-s” and “no-go-s” come in their respective events
- The list is empty = There are no problems = The interface is available

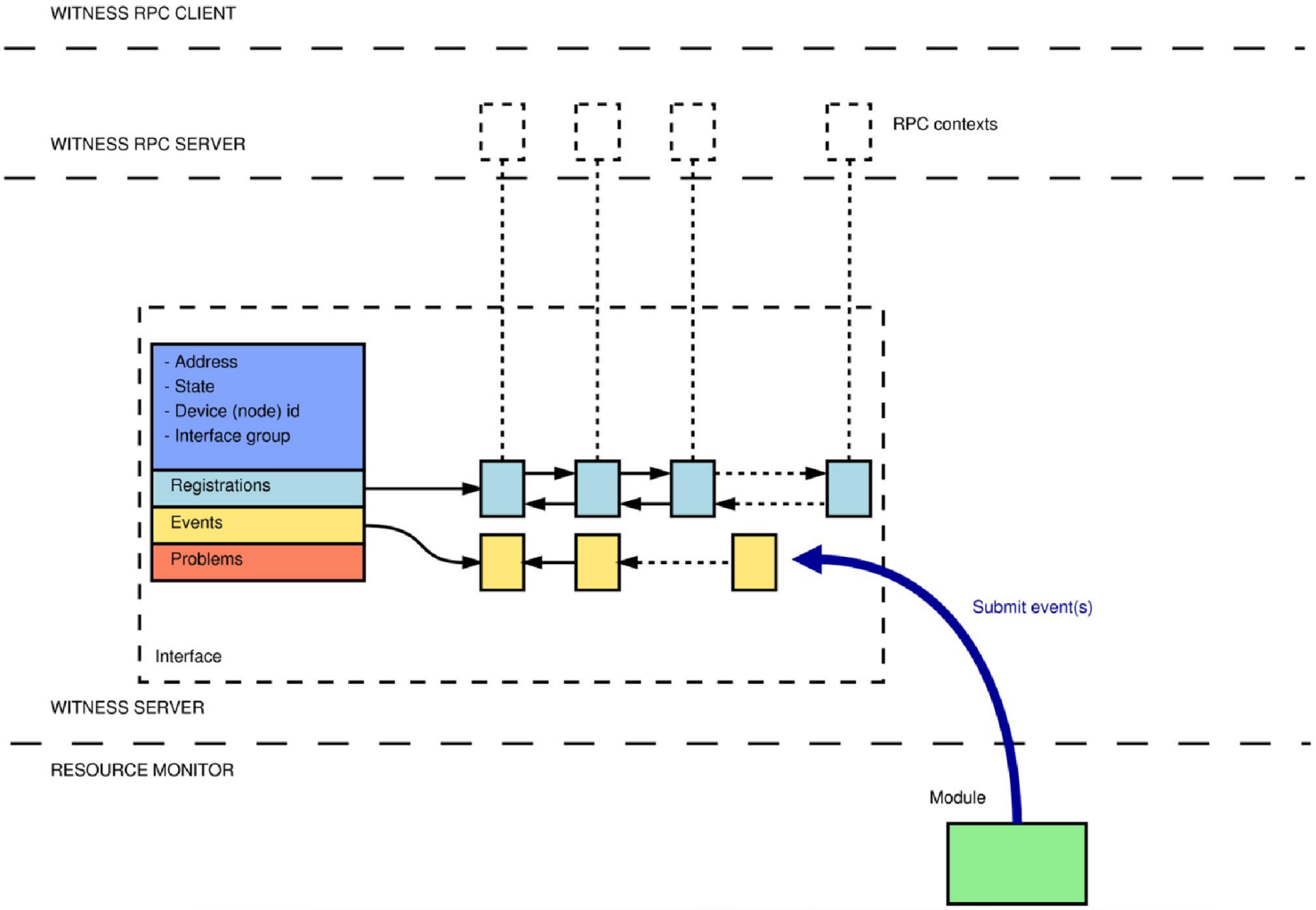
Keeping track of availability



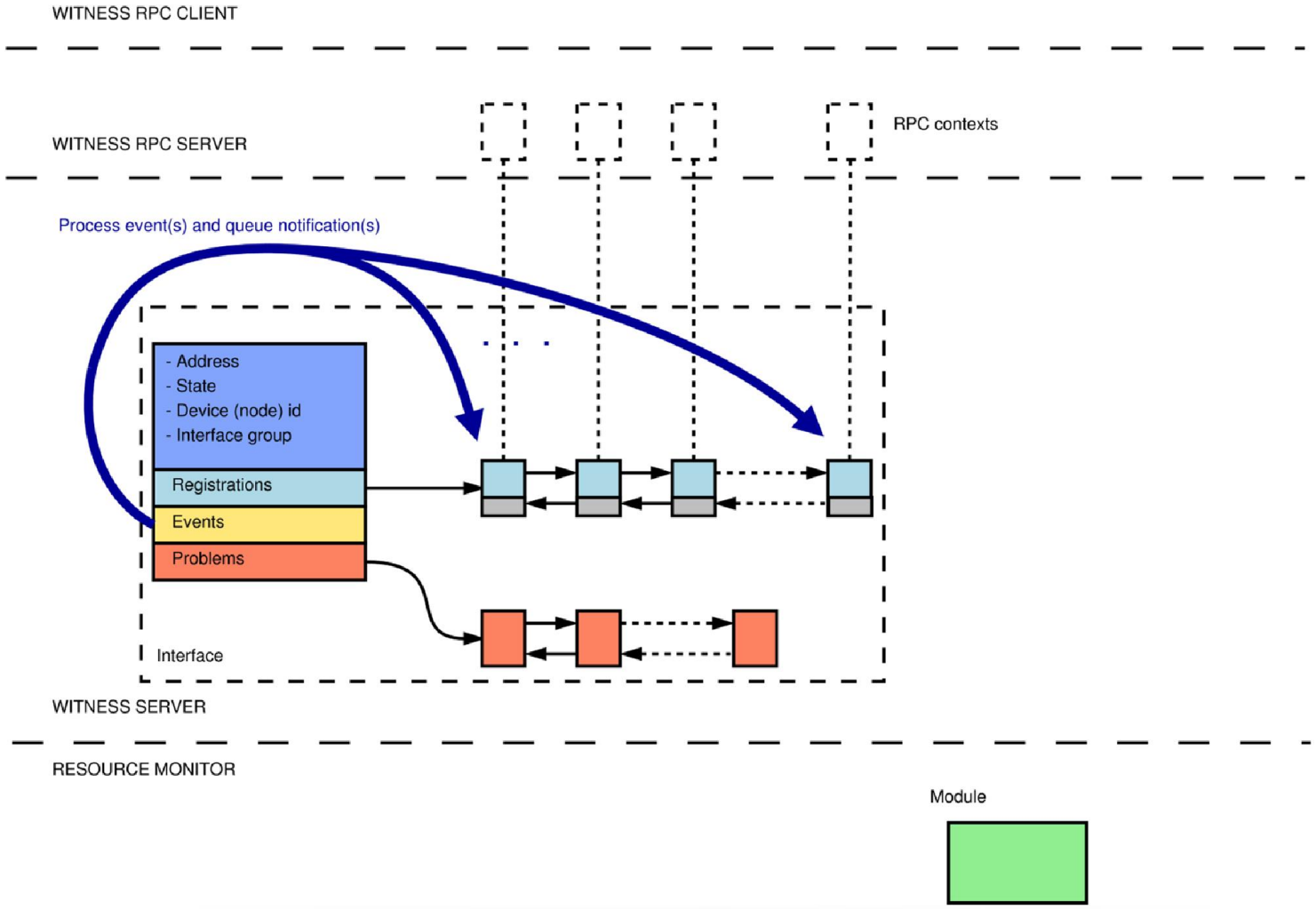
Updating interface state

- Any module can submit events to an interface at any time (given subscriptions)
- Witness server starts a *work item* (a function started in a separate thread) to process the events
- After processing, subsequent *work items* are started to queue notifications in each individual client registration
- Work items queuing the notifications resume execution of asynchronous request and send the responses to the witness clients

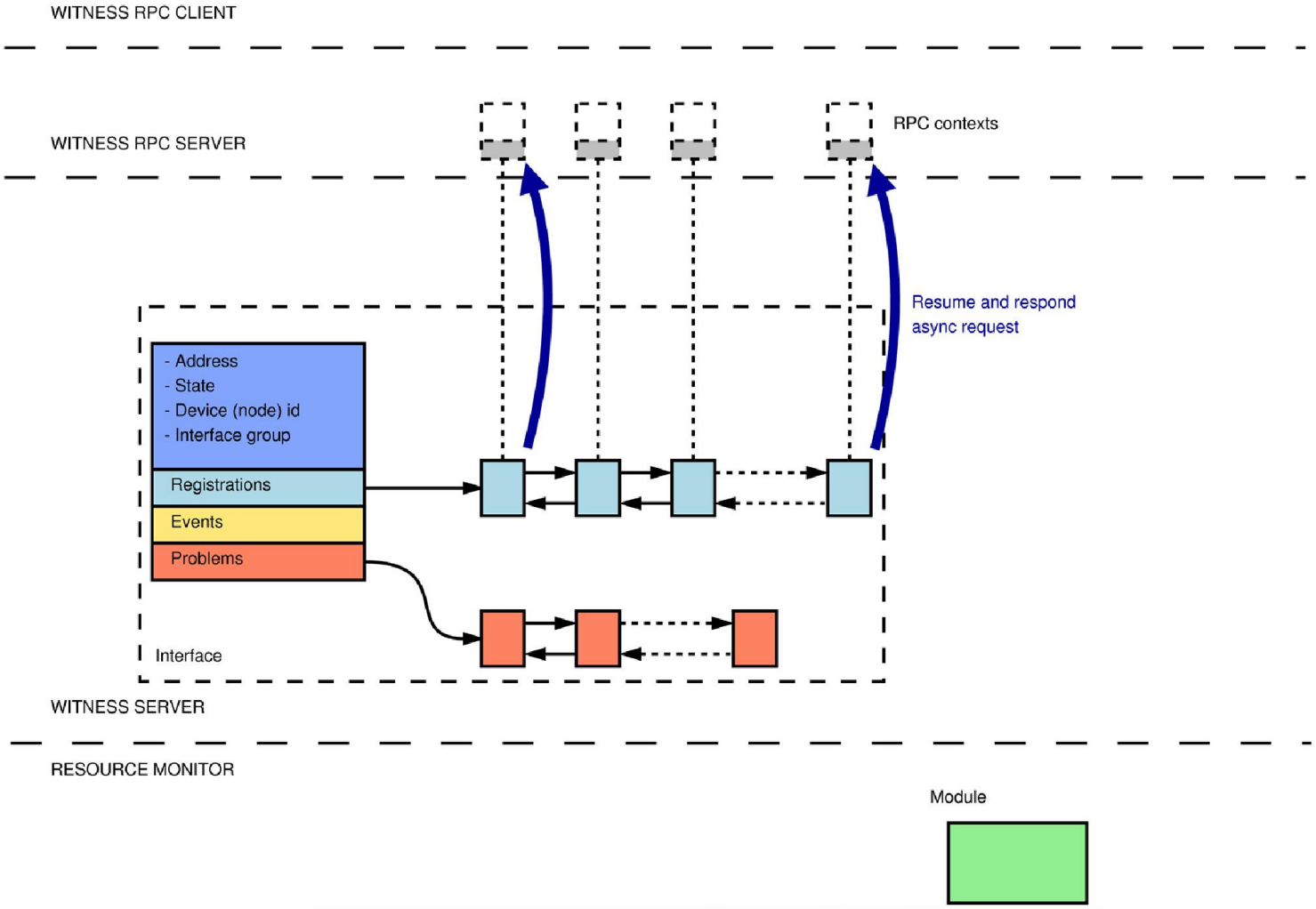
Updating interface (submit)



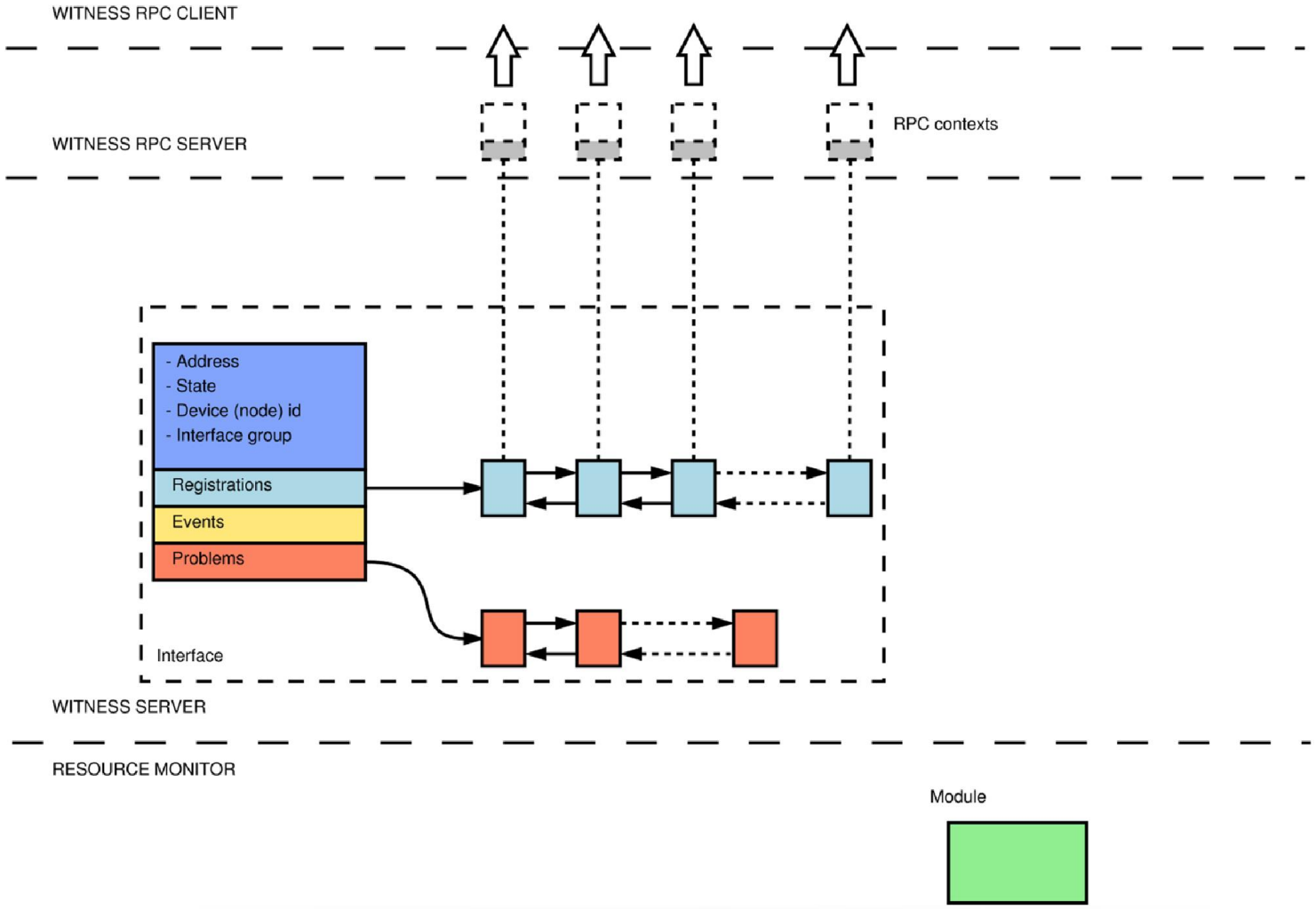
Updating interface state (process)



Updating interface state (wake up)



Updating interface state (notify)



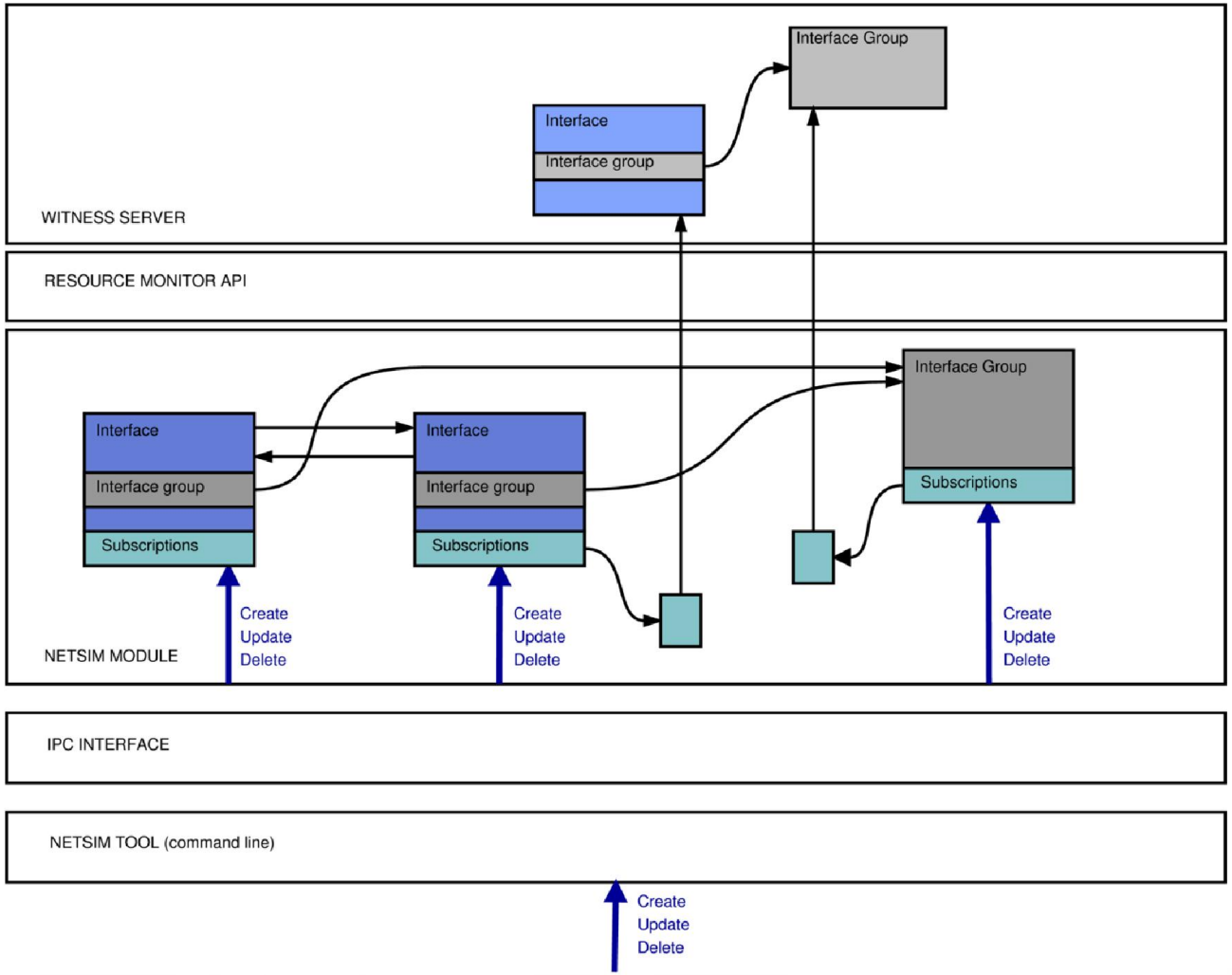
Resource monitor modules

- Different modules can keep track of different things independently
- Each module handles its specific failover scenario

Scenario: Testing

- A module with an IPC interface and a command line client simulates the network interfaces and groups and their changes
- Can create and keep an arbitrary number of groups and interfaces
- Useful for simulating unusual events

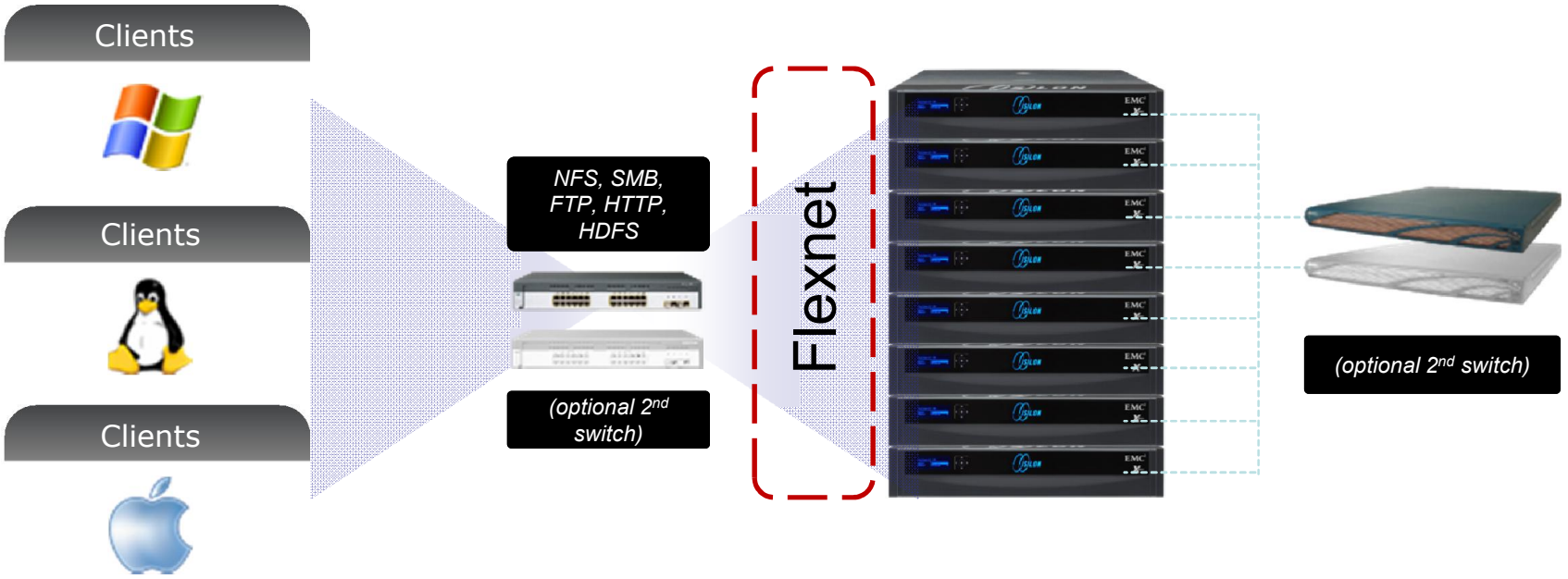
Testing module (netsim)



Scenario: Network interface failure

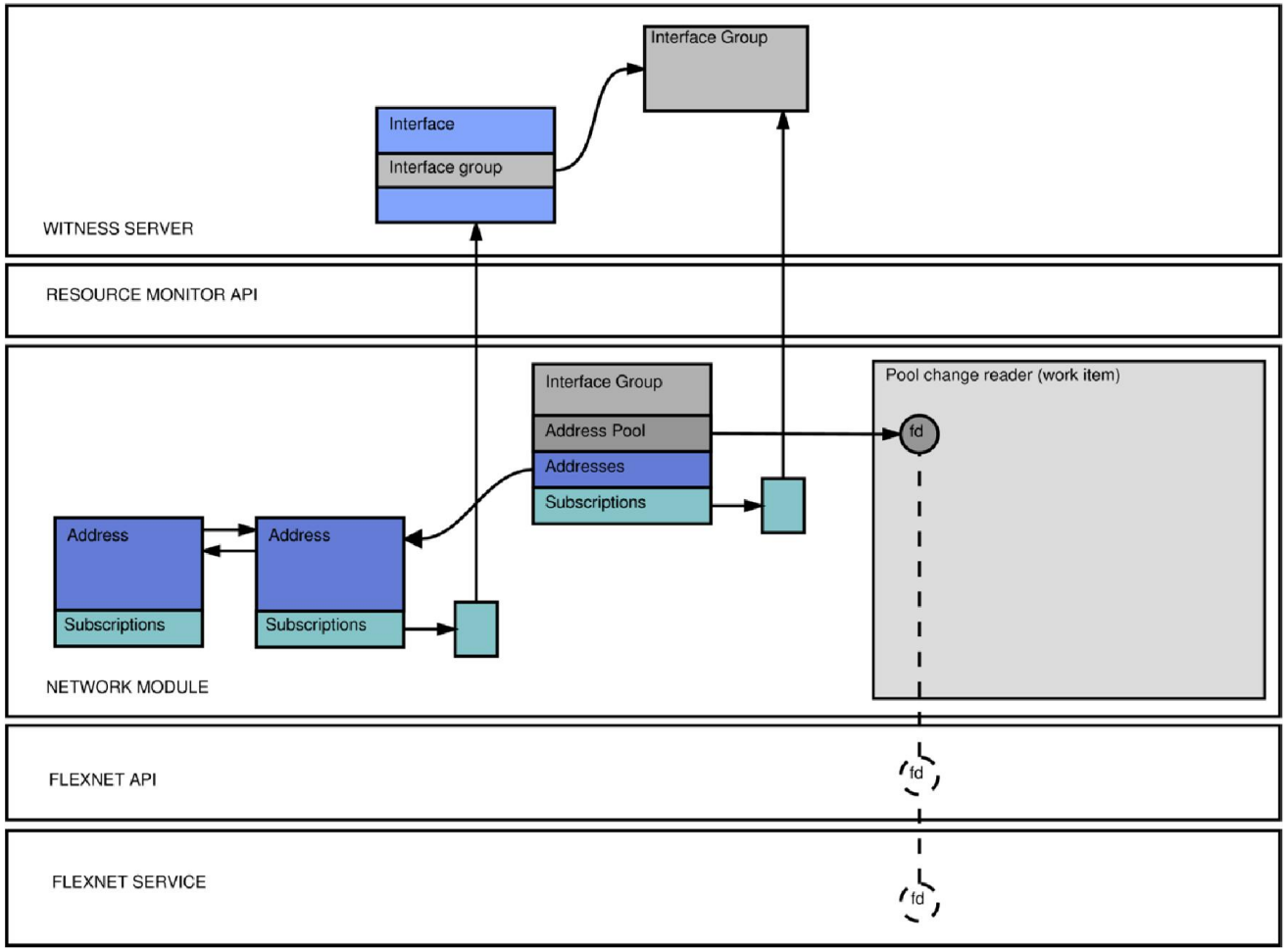
- Wired to OneFS cluster networking configuration (*Flexnet*)
- Interface and address pool information received from the system service
- Waiting for changes in a separate thread watching individual address pools
- Notified through file descriptors

Flexnet Service in OneFS cluster



Client/Application Layer	Ethernet Layer	Isilon IQ Storage Layer	Intracluster Communication Infiniband

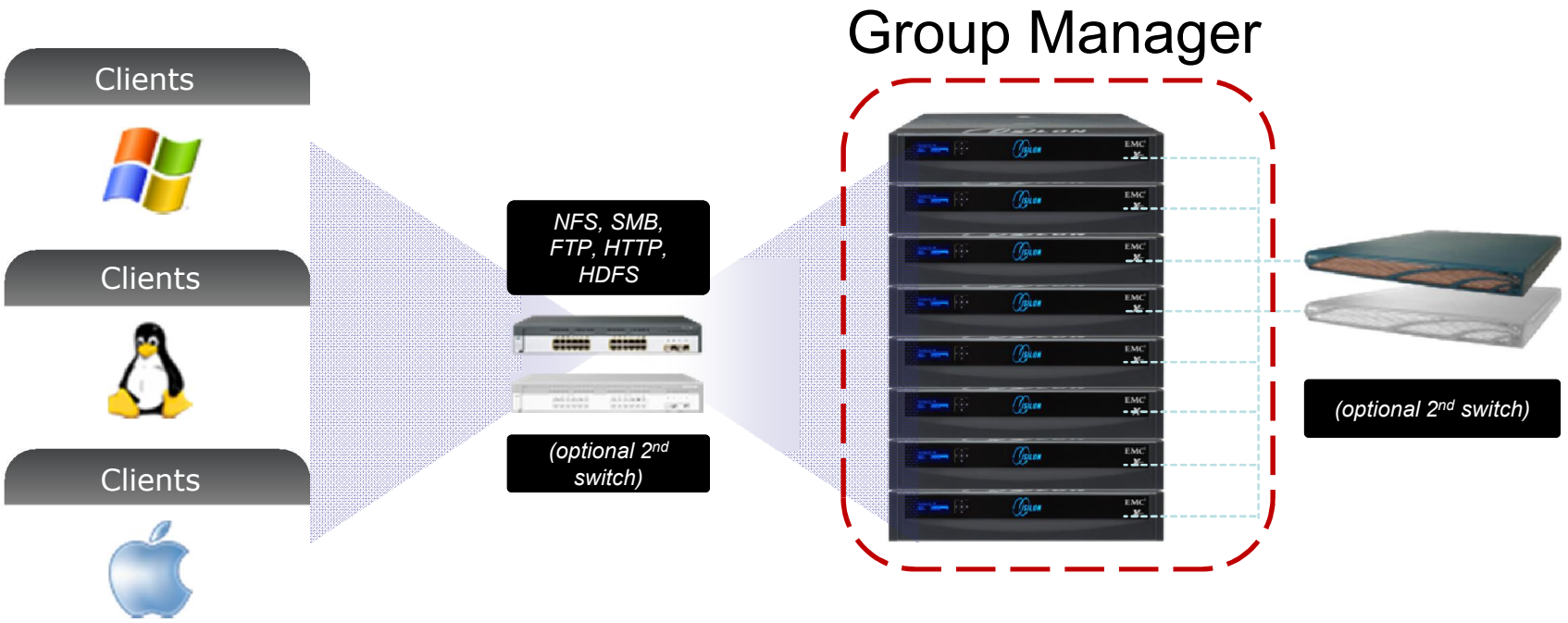
Network module



Scenario: Server process failure

- OneFS *Group Manager* watching other nodes in the cluster provides the feed
- It can keep track of the state of certain processes on other nodes
- The module gets notified about the changes in the same way as Network module

Group Manager in OneFS cluster



Client/Application Layer	Ethernet Layer	Isilon IQ Storage Layer	Intracluster Communication Infiniband

Scenario: Maintenance

- Sometimes we need to gracefully take a node off the cluster
- Existing client connections should “go away”
- The module can make the node interfaces look unavailable
- It can also move all connections to a different node or even a completely different group

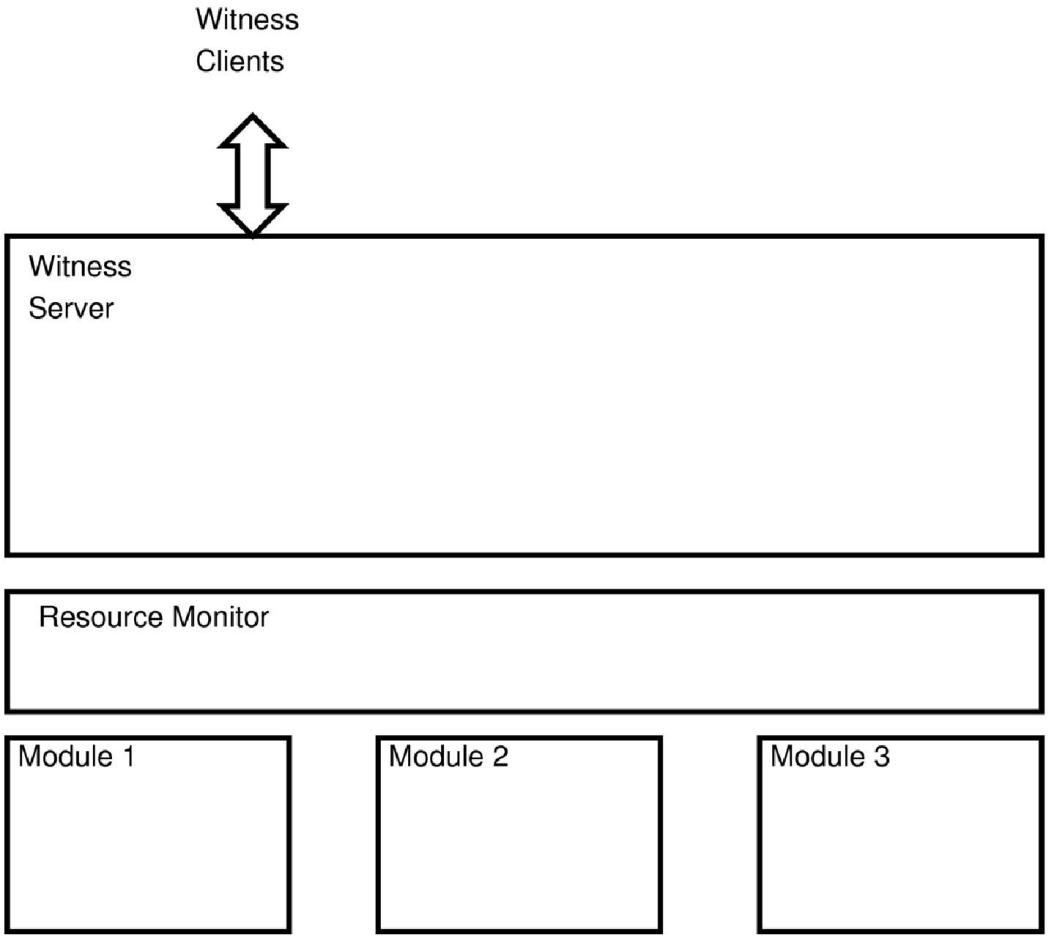
Beyond failover

- Witness “move” notification can be used for load balancing
- What would it take?
  - *Connection* resource type (to have a control over individual connections)
  - A module checking the load on other nodes and requesting the move if one of them is overloaded (perhaps another use for witness)

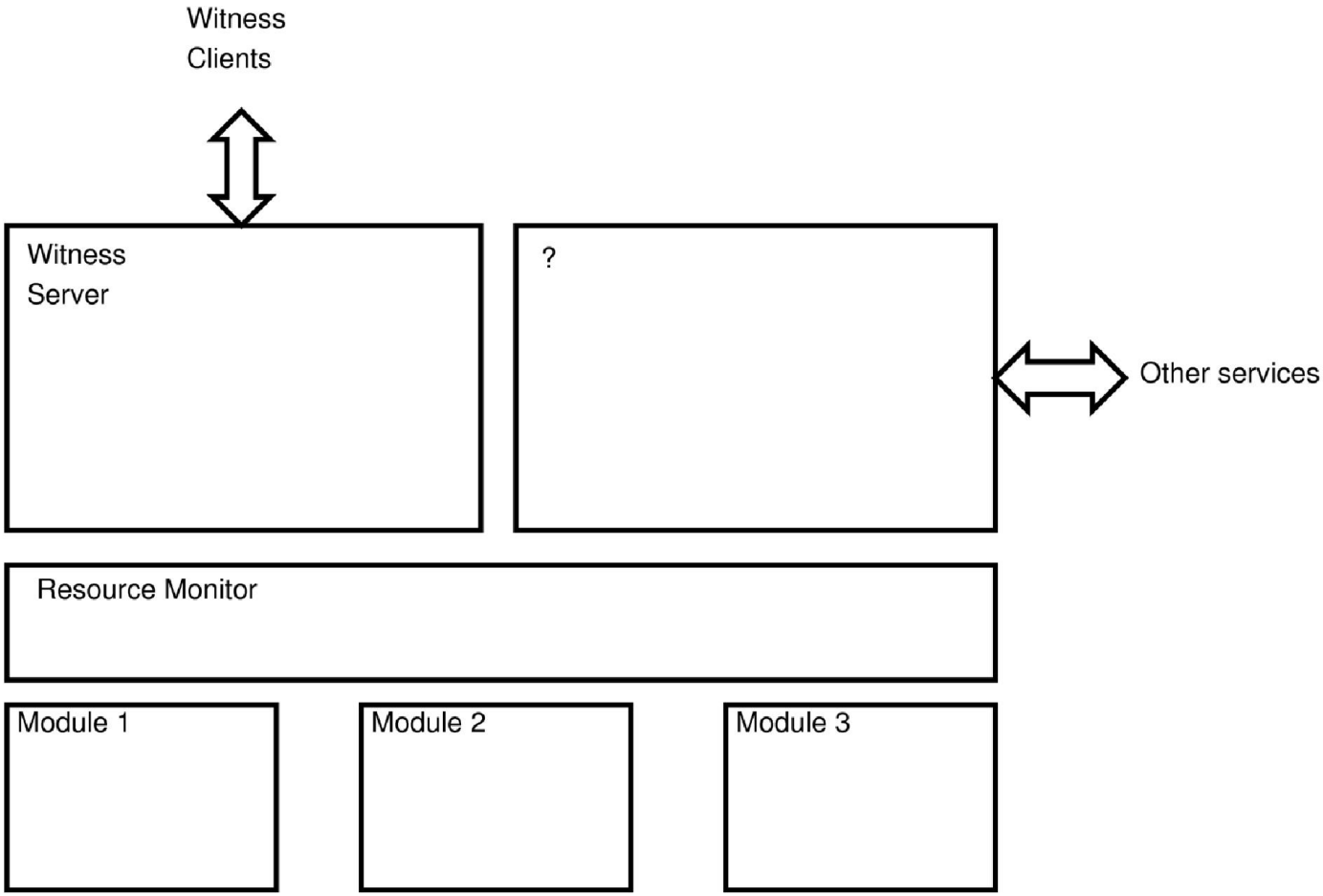
Beyond Witness itself

- Witness RPC is not in fact tied to SMB protocol very much
- Information provided by the Resource Monitor (network interfaces status) may be useful for other services, too

Beyond Witness itself



Beyond Witness itself



Thank you!

- Questions?

Rafal Szczesniak
rafal.szczesniak@isilon.com

43