CTDB Remix

I: Dreaming the Fantasy

Amitay Isaacs
amitay@samba.org

Samba Team
IBM (Australia Development Labs, Linux Technology Center)

SambaXP 2017

CTDB Project

Motivation: Support for clustered Samba

- Multiple nodes active simultaneously
- Communication between nodes (heartbeat, failover)
- Distributed databases between nodes

Features:

- Volatile and Persistent databases
- Cluster-side messaging for Samba
- IP failover and load balancing
- Service monitoring

Community:

- `ctdb.samba.org`
- `git.samba.org/samba.git`
- `wiki.samba.org/index.php/CTDB_and_Clustered_Samba`

Overview

- Dreaming the Fantasy
  - How did we get here?
  - Evolving the design
  - Laying the foundations
  - New Architecture
- Designing the Reality

How did we get here?

SambaXP 2013

Wish List

- Split monolithic code into separate daemons
  - Logging, IP handling, Services monitoring
- Proper CTDB library - libctdb
  - Database operations are missing
  - Thread-safe (avoid talloc/tevent?)
- CTDB Protocol
  - Version tracking
  - Auto-generated marshalling/unmarshalling code
- Scalability – large number of nodes
  - Database recovery
  - Handling record contention
- Pluggable Monitoring and Failover
  - Integration with 3rd party HA

SambaXP 2013

Wish List

- Split monolithic code into separate daemons
  - Logging, IP handling, Services monitoring
- Proper CTDB library - libctdb
  - Database operations are missing
  - Thread-safe (avoid talloc/tevent?)
- CTDB Protocol
  - Version tracking
  - Auto-generated marshalling/unmarshalling code
- Scalability – large number of nodes
  - Database recovery
  - Handling record contention
- Pluggable Monitoring and Failover
  - Integration with 3rd party HA

- Introduced lock helper, event helper

2014 Recap

CTDB merges with Samba

- CTDB merged into Samba tree (Nov 2013)
- CTDB standalone waf build (Jun 2014)
- CTDB build integrated in toplevel build (Nov 2014)

2015 Recap

Parallel Database Recovery

- Protocol marshalling
- New abstractions
- New Communication framework (`tevent_req` based async)
- New Client Code
- Database recovery helper
- Re-implemented `ctdb` tool using new client API

2015 Recap

Parallel Database Recovery

- Protocol marshalling
- New abstractions
- New Communication framework (`tevent_req` based async)
- New Client Code
- Database recovery helper
- Re-implemented `ctdb` tool using new client API
- Introduced natgw helper

2016 Recap

- Introduced killtcp helper, lvs helper

2016 Recap

- Introduced killtcp helper, lvs helper

Event daemon

- New abstractions - `run_proc`, `sock_daemon`
- Event Protocol
- Event client code
- Event handling daemon

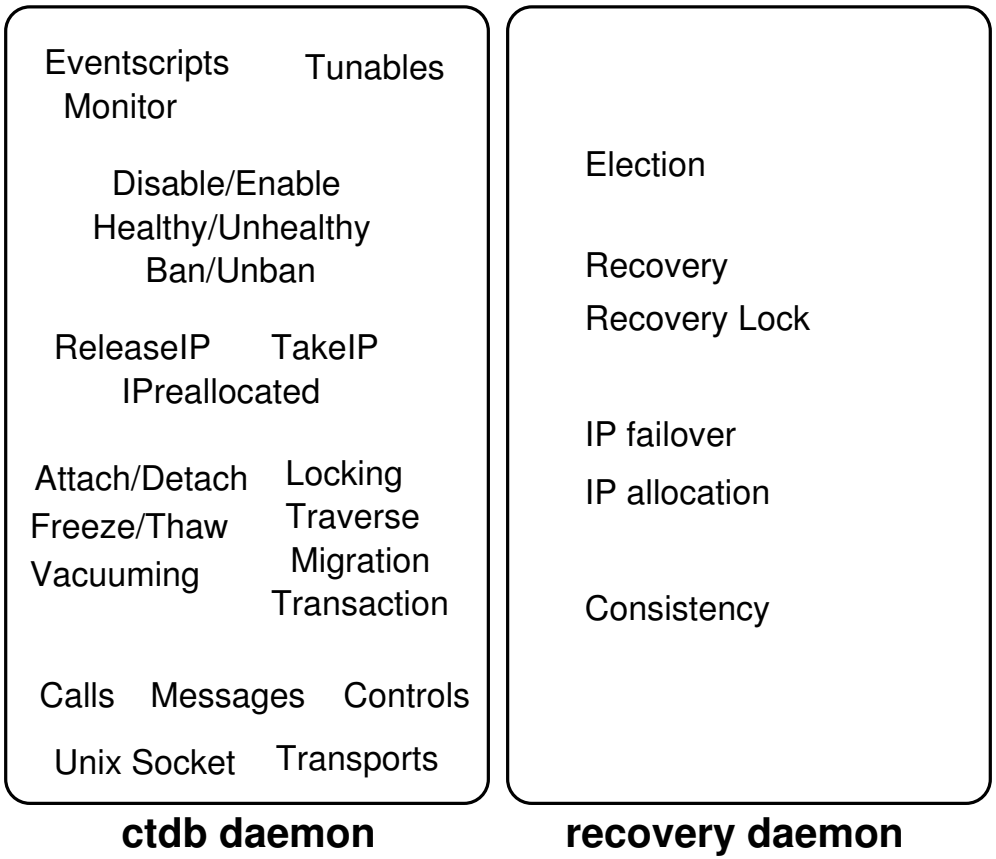
Evolving the design

Evolving the design

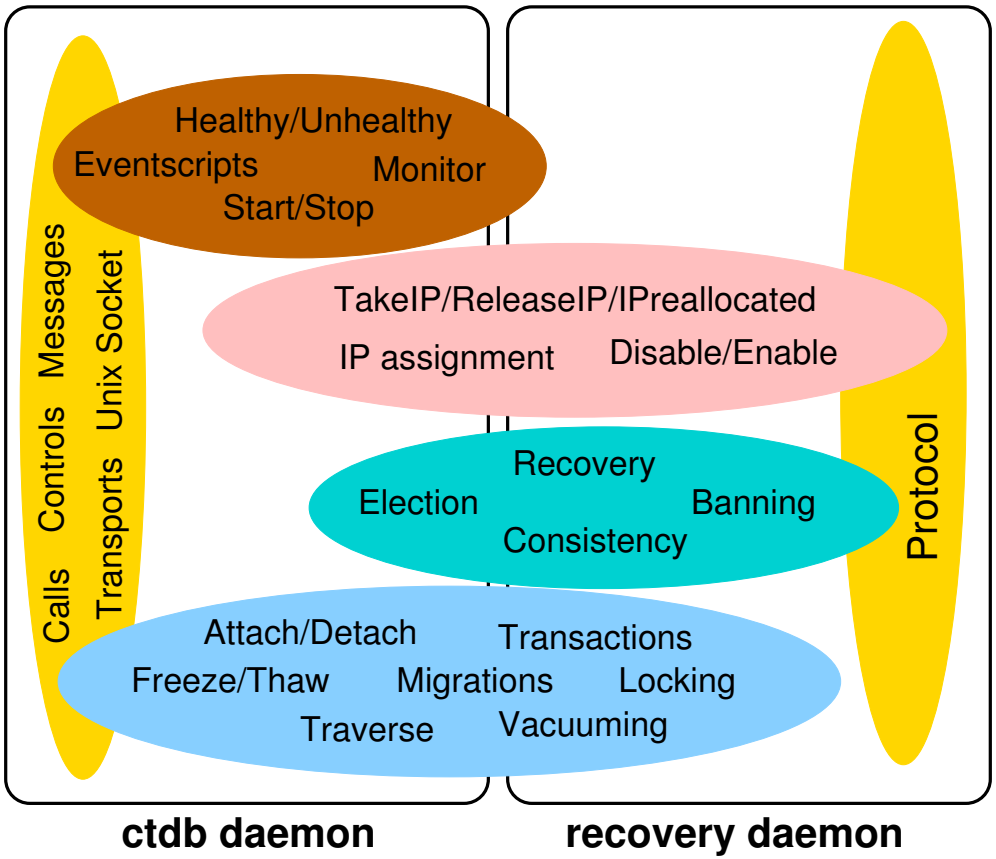
Identifying CTDB functions

Evolving the design

Identifying CTDB functions

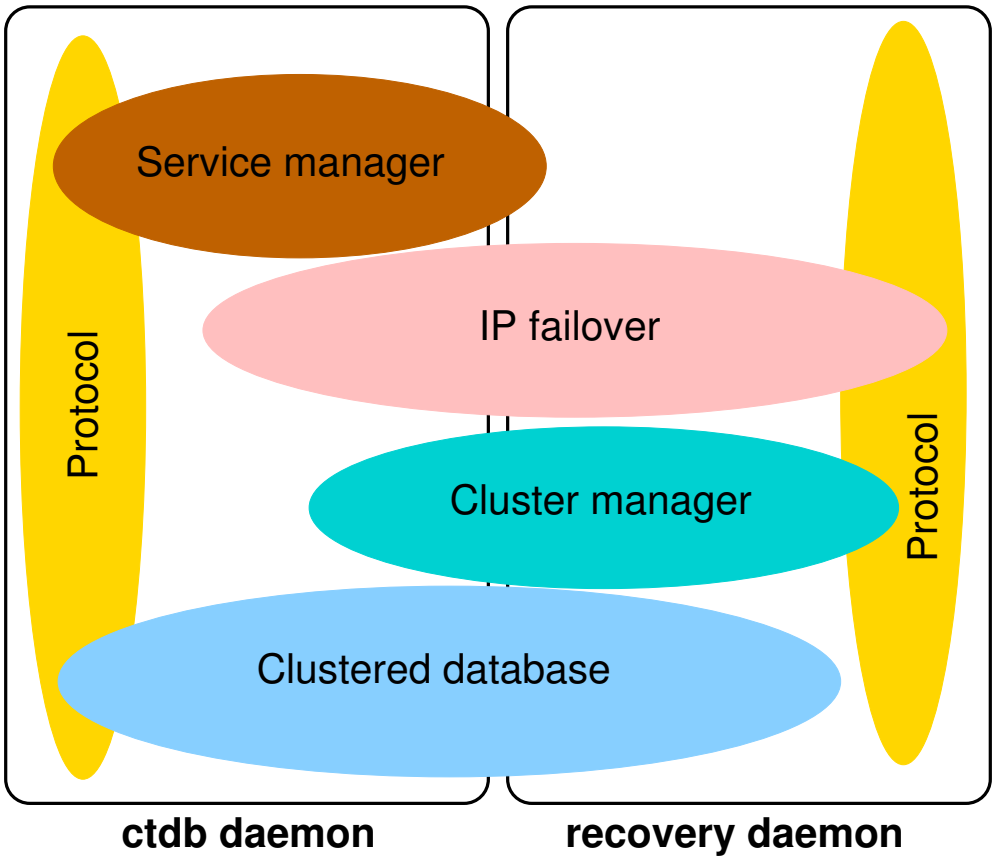


Grouping CTDB functions



Evolving the design

New subsystems



Evolving the design

Redesign of server code

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport

Motivation

- Avoid `m*n*n` connections

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport
  - Develop parallel transport and proxies (scalability)

Motivation

- Avoid `m*n*n` connections

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport
  - Develop parallel transport and proxies (scalability)
  - Convert CTDB transport to use proxy

Motivation

- Avoid `m*n*n` connections

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport
  - Develop parallel transport and proxies (scalability)
  - Convert CTDB transport to use proxy

Motivation

- Avoid `m*n*n` connections
- Unix datagrams (SambaXP 2015)
- tmsgd - fd passing (SambaXP 2016)

Evolving the design

Redesign of server code

- First approach
  - Main concern is the transport
  - Develop parallel transport and proxies (scalability)
  - Convert CTDB transport to use proxy

Motivation

- Avoid `m*n*n` connections
- Unix datagrams (SambaXP 2015)
- tmsgd - fd passing (SambaXP 2016)

- Proxy design never took off

Evolving the design

Redesign of server code

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

- Does not need any CTDB infrastructure

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

- Does not need any CTDB infrastructure
- `run_proc` abstraction

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

- Does not need any CTDB infrastructure
- `run_proc` abstraction
- New protocol - request, reply

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

- Does not need any CTDB infrastructure
- `run_proc` abstraction
- New protocol - request, reply
- Easy testing

Evolving the design

Redesign of server code

- Second approach
  - Split code out and create separate daemons
  - Avoid boilerplate, most daemons to use unix domain sockets
  - `sock_daemon` abstraction
  - First candidate — event daemon

`eventd`

- Does not need any CTDB infrastructure
- `run_proc` abstraction
- New protocol - request, reply
- Easy testing
- `run_event` abstraction

Laying the foundations

Laying the foundations

CTDB State management

- Solved differently for different things
  - tickles - Protocol to sync tickle lists
  - nfs locks - Using persistent database
- Persistent databases are slow
- We are in the business of clustered databases
- New database model?

Laying the foundations

CTDB State management

- Solved differently for different things
  - tickles - Protocol to sync tickle lists
  - nfs locks - Using persistent database
- Persistent databases are slow
- We are in the business of clustered databases
- New database model?

Replicated database

- State information needed during lifetime of CTDB
- Volatile (`CLEAR_IF_FIRST`)
- Replicated (Re-use existing API)
- Uses `g_lock` and transactions

Laying the foundations

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

Laying the foundations

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

persistent

replicated

Laying the foundations

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

		on disk	tmpfs
g_lock test	fcntl	5718	5750
persistent			
replicated			

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

replicated

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

	on disk	tmpfs
fcntl	11	11

replicated

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

	on disk	tmpfs
fcntl	11	11
mutexes	11	11

replicated

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

	on disk	tmpfs
fcntl	11	11
mutexes	11	11

replicated

	on disk	tmpfs
fcntl	583	619

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

	on disk	tmpfs
fcntl	11	11
mutexes	11	11

replicated

	on disk	tmpfs
fcntl	583	619
mutexes	80	89

Database Performance

- 2 nodes (Intel Xeon E5620, RHEL6), 30 second test

g_lock test

	on disk	tmpfs
fcntl	5718	5750
mutexes	7573	7893

persistent

	on disk	tmpfs
fcntl	11	11
mutexes	11	11

replicated

	on disk	tmpfs
fcntl	583	619
mutexes	80	89

- Why are transactions with mutexes so slow?

Laying the foundations

Node-to-node communication

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

- New packet type `CTDB_REQ_TUNNEL`

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

- New packet type `CTDB_REQ_TUNNEL`
- Uses existing CTDB transport

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

- New packet type `CTDB_REQ_TUNNEL`
- Uses existing CTDB transport
- Register tunnels with `tunnel_id` (new controls)

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

- New packet type `CTDB_REQ_TUNNEL`
- Uses existing CTDB transport
- Register tunnels with `tunnel_id` (new controls)
- Client API to encapsulate packets

Laying the foundations

Node-to-node communication

- `CTDB_REQ_MESSAGE`
  - Can carry abritrary data
  - Used by recovery daemon, samba
  - `SRVID` based
- Several issues
  - Multiple processes can get same message
  - Fire and forget

Tunnels

- New packet type `CTDB_REQ_TUNNEL`
- Uses existing CTDB transport
- Register tunnels with `tunnel_id` (new controls)
- Client API to encapsulate packets
- New daemons can use new protocol

Laying the foundations

Rethinking subsystem design

Laying the foundations

Rethinking subsystem design

- Loose coupling to CTDB

Laying the foundations

Rethinking subsystem design

- Loose coupling to CTDB
  - Notifications (subsystem $\rightarrow$)

Laying the foundations

Rethinking subsystem design

- Loose coupling to CTDB
  - Notifications (subsystem $\rightarrow$)
  - Actions (subsystem $\leftarrow$)

Laying the foundations

Rethinking subsystem design

- Loose coupling to CTDB
  - Notifications (subsystem $\rightarrow$)
  - Actions (subsystem $\leftarrow$)
- State transition graphs

Laying the foundations

Rethinking subsystem design

- Loose coupling to CTDB
  - Notifications (subsystem $\rightarrow$)
  - Actions (subsystem $\leftarrow$)
- State transition graphs
- Candidates
  - Cluster Manager
  - Service Monitoring
  - IP Failover

New Architecture

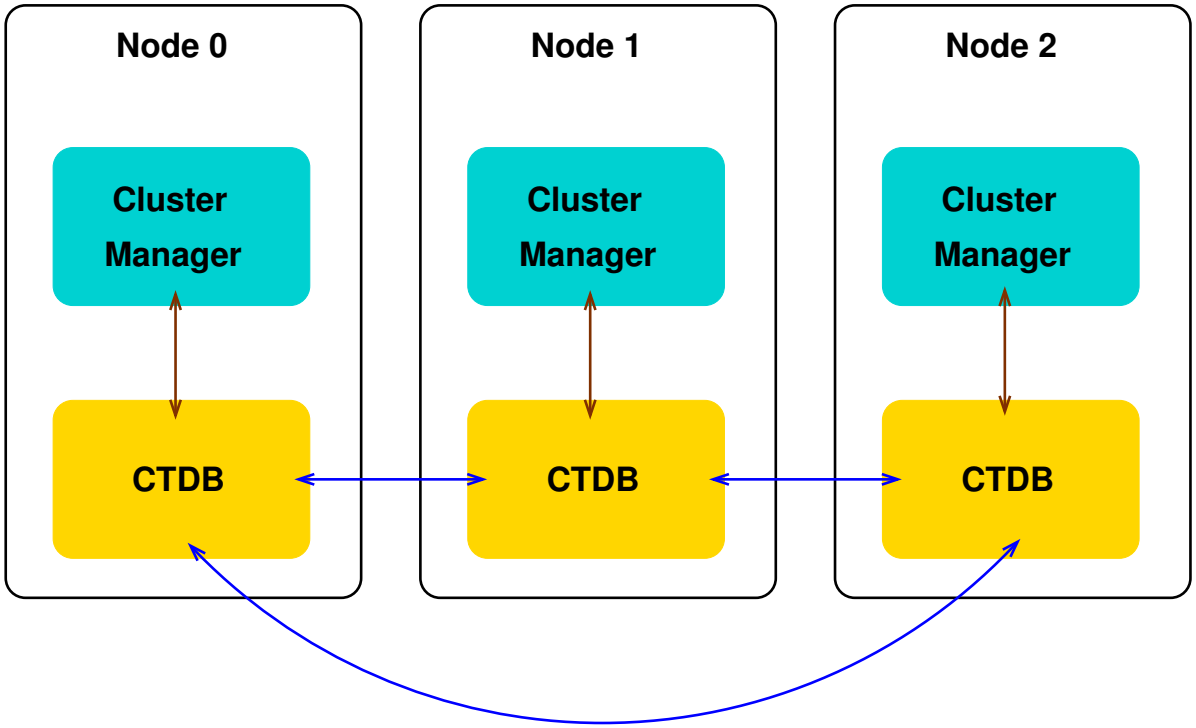
Cluster Manager

Cluster Manager

- Split cluster manager code

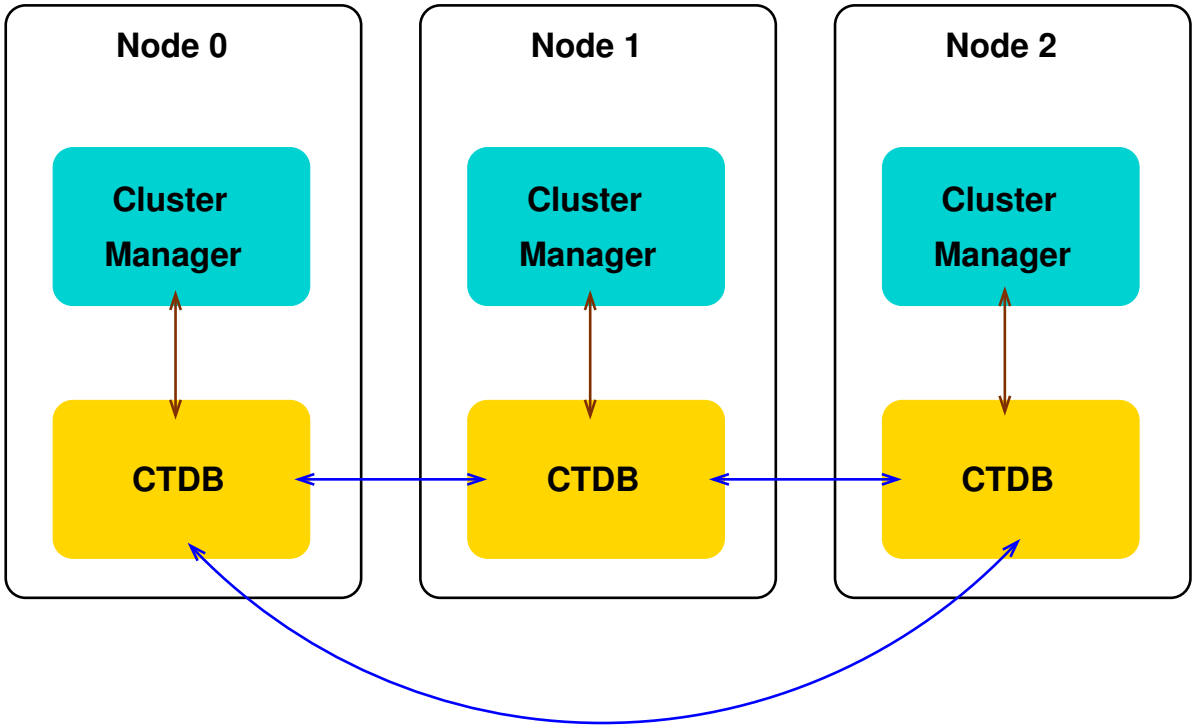
Cluster Manager

- Split cluster manager code



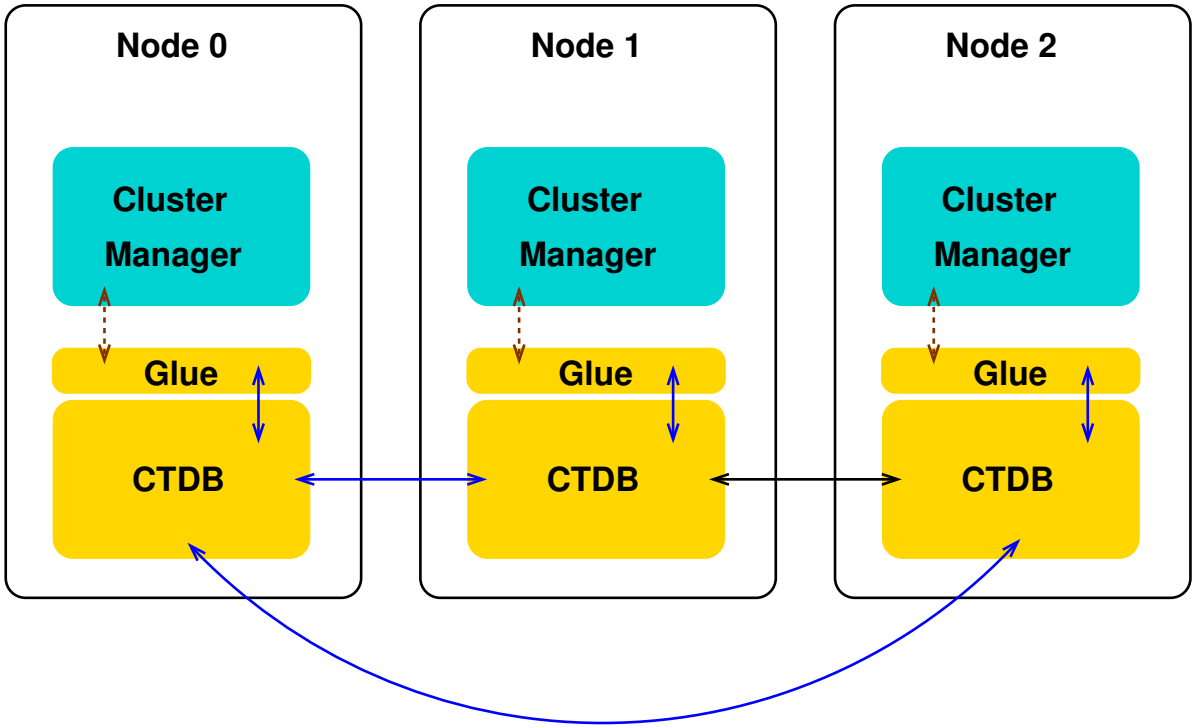
Cluster Manager

- Split cluster manager code
- Keep it loosely coupled



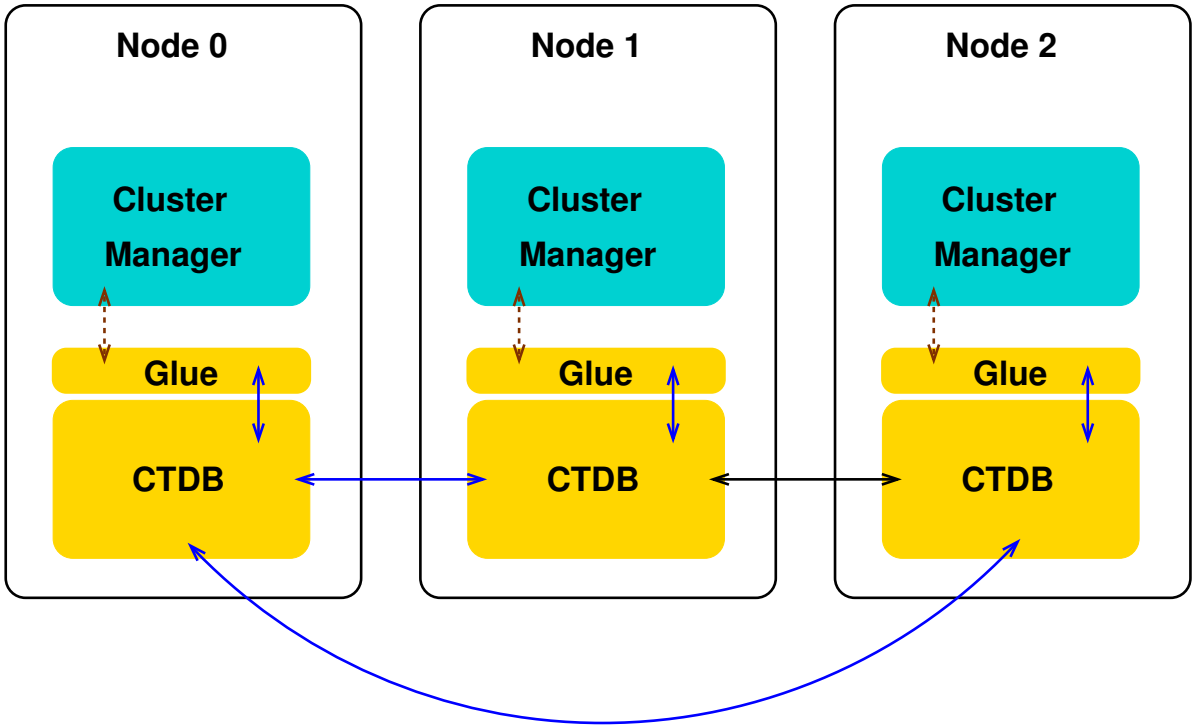
Cluster Manager

- Split cluster manager code
- Keep it loosely coupled



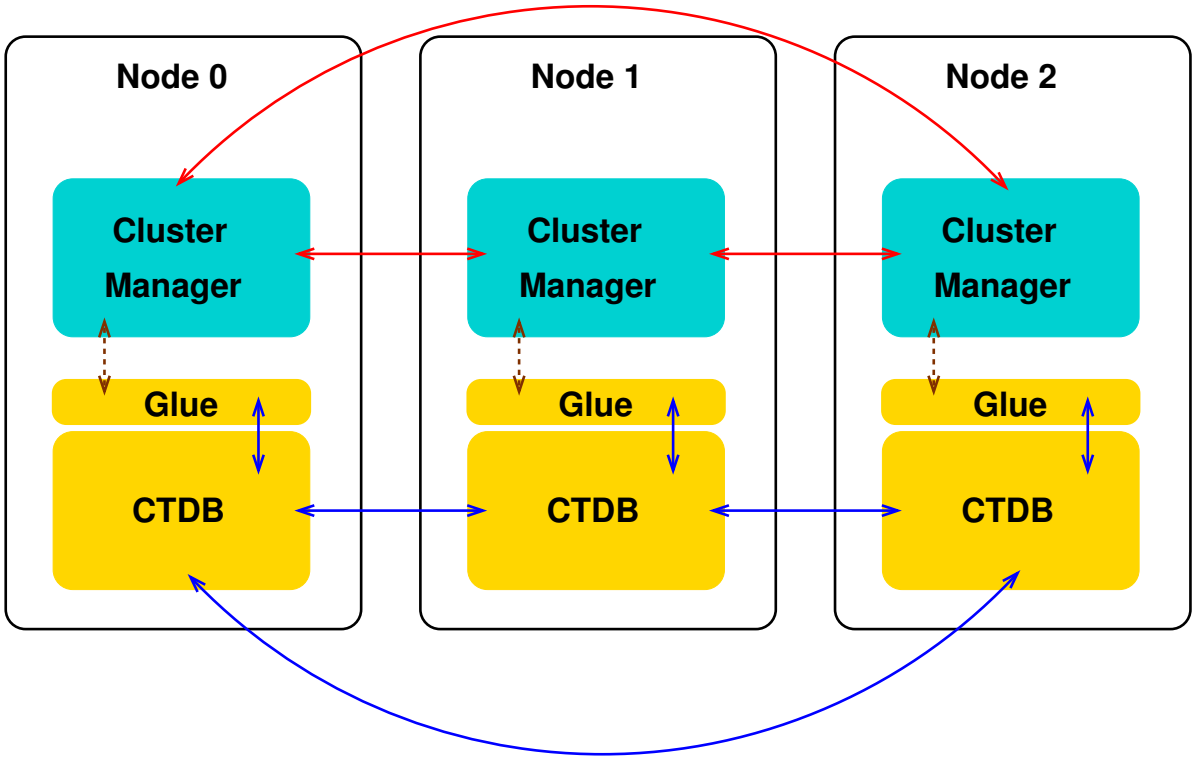
Cluster Manager

- Split cluster manager code
- Keep it loosely coupled
- Support 3rd party replacements (e.g. etcd)



Cluster Manager

- Split cluster manager code
- Keep it loosely coupled
- Support 3rd party replacements (e.g. etcd)



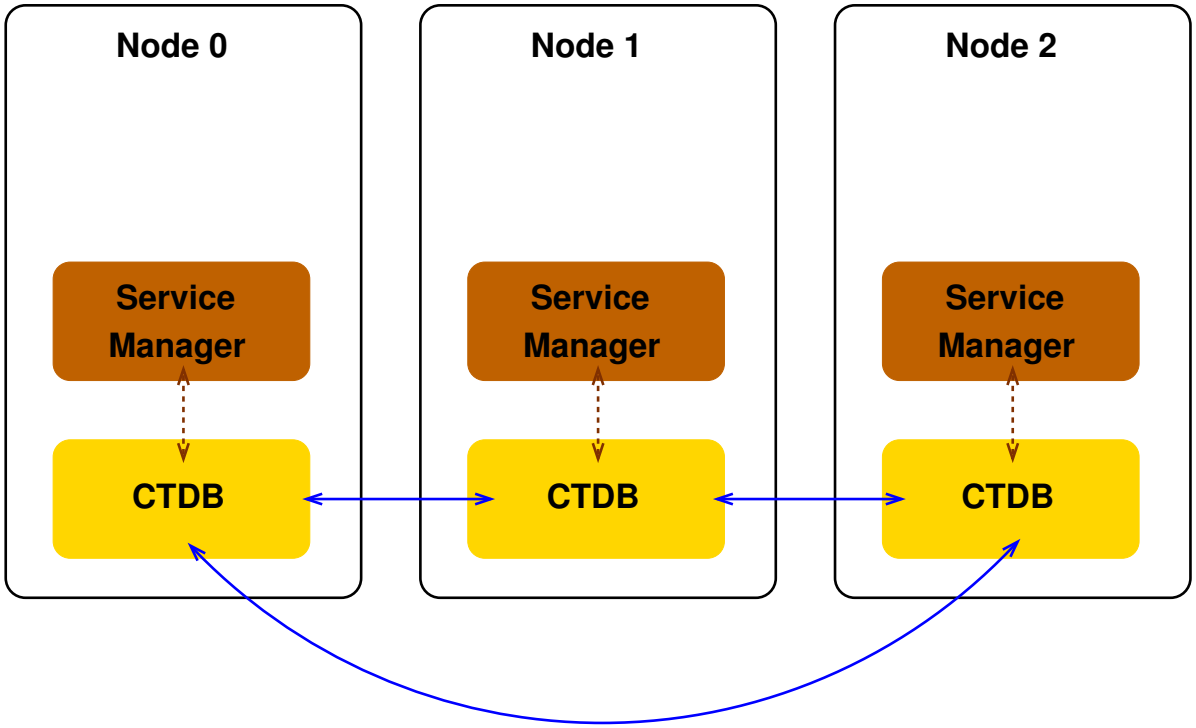
Service Manager and IP Failover

Service Manager and IP Failover

- Separate service manager code (`eventd` + . . .)

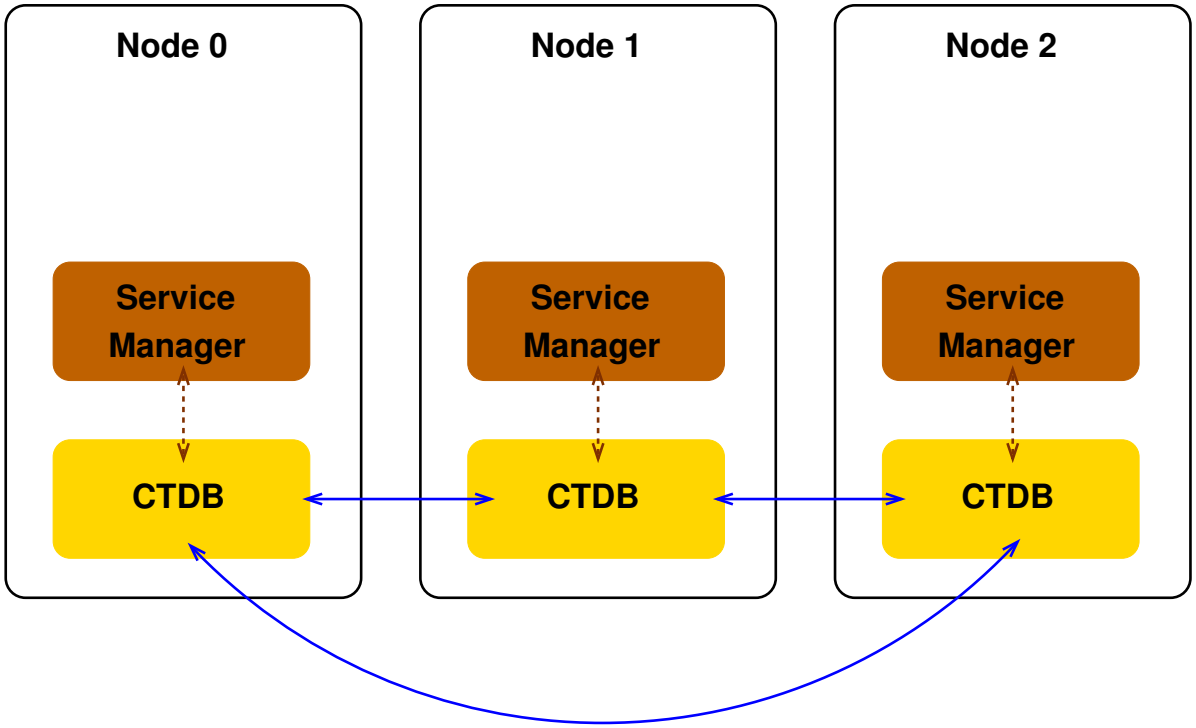
Service Manager and IP Failover

- Separate service manager code (`eventd` + . . .)



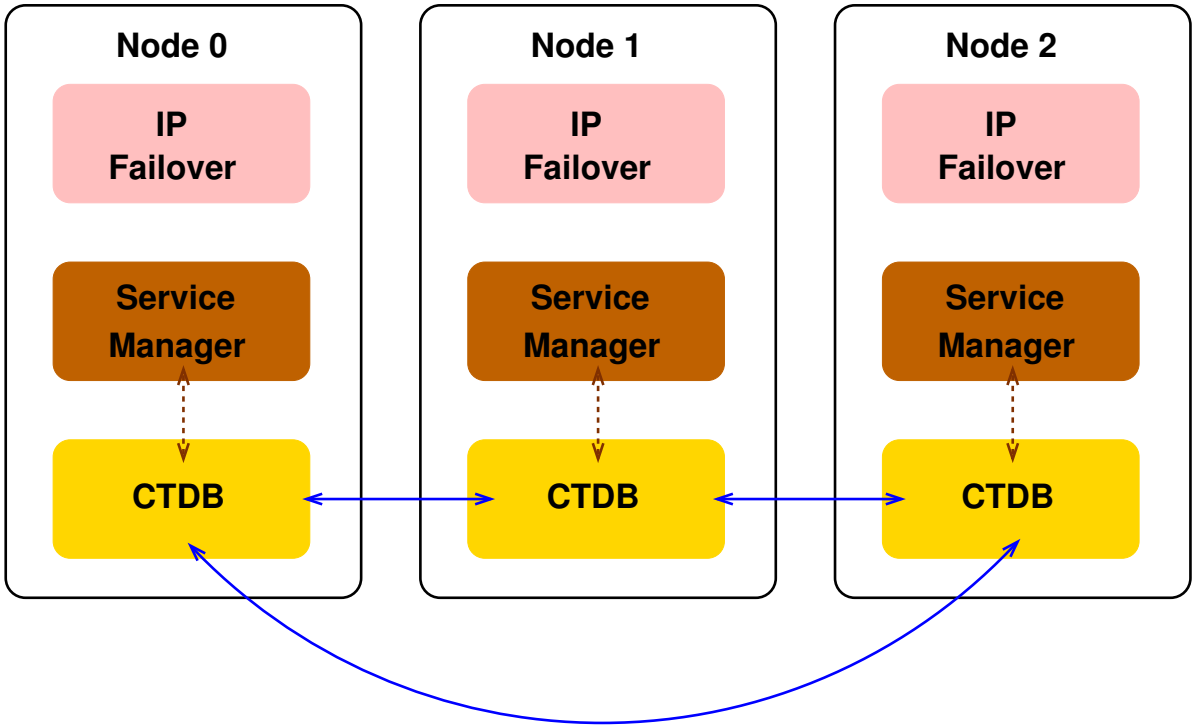
Service Manager and IP Failover

- Separate service manager code (`eventd` + ...)
- Separate IP failover code



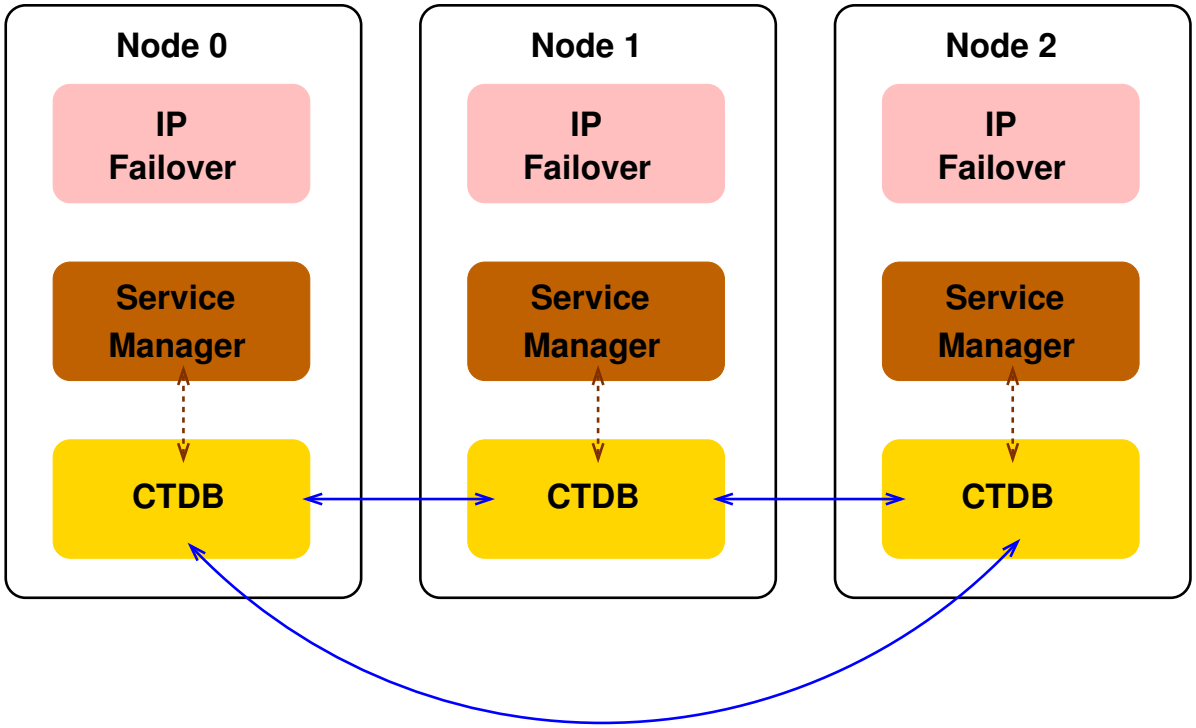
Service Manager and IP Failover

- Separate service manager code (`eventd` + ...)
- Separate IP failover code



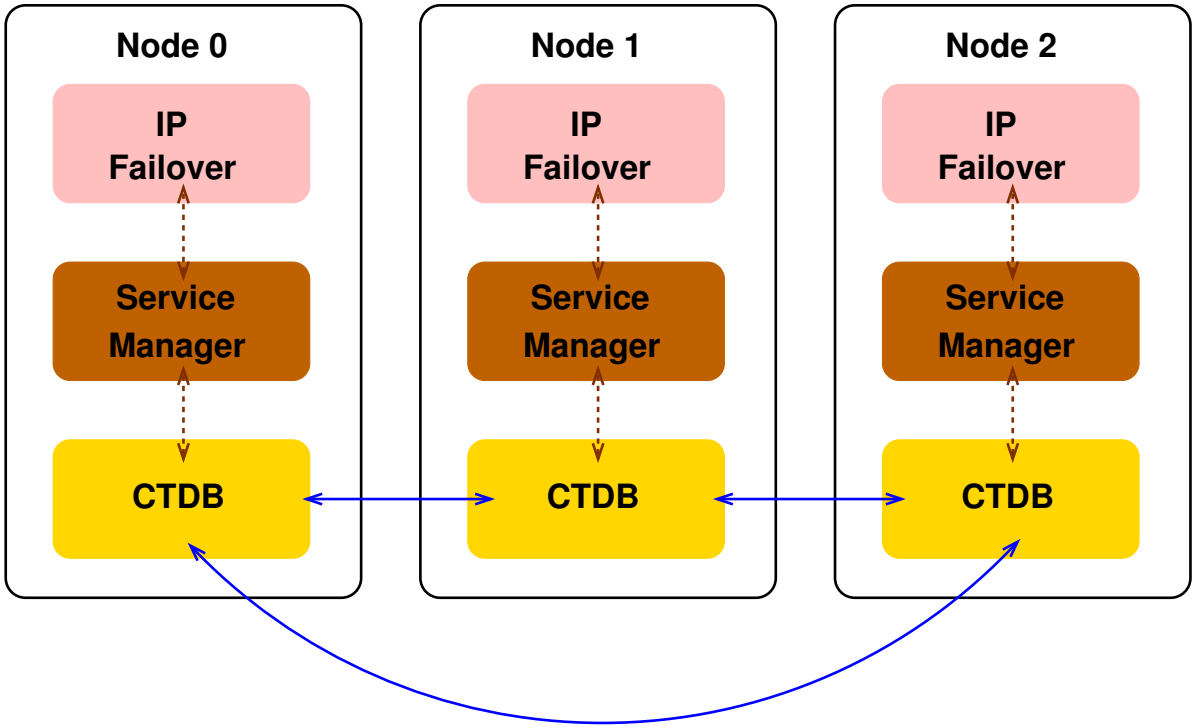
Service Manager and IP Failover

- Separate service manager code (`eventd` + ...)
- Separate IP failover code
- IP failover as a service



Service Manager and IP Failover

- Separate service manager code (`eventd` + ...)
- Separate IP failover code
- IP failover as a service



Questions / Comments