The CTDB Report

Martin Schwenke <martin@meltin.net>
Amitay Isaacs <amitay@samba.org>

Samba Team / IBM (ADL, LTC)

SambaXP 2020

Overview

Audience

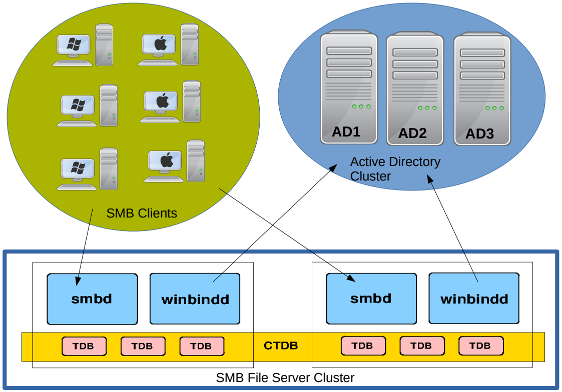
Developers! :-)

Audience

Developers! :-)

. . . but don't leave if you're not a developer. . . this might still be interesting!

Clustered Samba



What is CTDB?

- Clustered database for Samba metadata

What is CTDB?

- Clustered database for Samba metadata
- Cluster-wide messaging transport

What is CTDB?

- Clustered database for Samba metadata
- Cluster-wide messaging transport
- Cluster management — leadership, membership

What is CTDB?

- Clustered database for Samba metadata
- Cluster-wide messaging transport
- Cluster management — leadership, membership
- Dynamic IP address failover

Progress in the past year

Committers

Martin Schwenke	269
Amitay Isaacs	32
Björn Jacke	14
Volker Lendecke	11
Ralph Boehme	8
Mathieu Parent	3
Anoop C S	2
David Disseldorp	2
Swen Schillig	2
Andrew Bartlett	1
Björn Baumbach	1
Günther Deschner	1
Noel Power	1
Rafael David Tinoco	1
Renaud Fortier	1
	349

Commits by area

Test (flakiness, portability, restructuring, ...)	140
Code cleanups (csbuild, memory leaks, ...)	44
Vacuuming (improvements, simplification, testing)	29
TCP connectivity (bug fixes)	24
Typos (docs, debug, comments, ...)	19
Cluster mutex (lock rechecking, ...)	18
Recovery (bug fixes, improvements)	17
Common code features (cmdline, conf, ...)	9
Hot records (bug fixes)	8
Build	8
Tools (ctdb, onnode, ...)	6
Scripts (event scripts, ...)	6
Docs	2
Other	19
Total	349

Tests

- Clustered Samba now tested in autobuild
  - Effort originally started by Michael Adam
  - Continued and completed by Volker Lendecke
  - Assisted by Martin (integrated CTDB's `local_daemons.sh`)
  - Initially just a single test (`base.ntdeny2`)

Tests

- Clustered Samba now tested in autobuild
  - Effort originally started by Michael Adam
  - Continued and completed by Volker Lendecke
  - Assisted by Martin (integrated CTDB's `local_daemons.sh`)
  - Initially just a single test (`base.ntdeny2`)
- Added collections of test suites:
  UNIT, INTEGRATION, CLUSTER
- Formalised test results: skip, fail
- Fixed a lot of flaky tests
- More test infrastructure passes ShellCheck
- Lots of cleanups

Code cleanups

- CTDB standalone compile nearly passes csbuild
- `ctdb/` subdirectory is now `unsigned`/`signed`-clean
- More shell scripts (mostly) pass ShellCheck

Vacuuming

- Vacuuming simplified
- All in the vacuuming child (nothing left in recovery daemon)
- Vacuuming child fetches records to LMASTER for deletion
- Added control to trigger fast vacuuming run for testing
- Now have quite extensive vacuuming tests

TCP connectivity

Connectivity problems when starting lots of daemons

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1 Reverted a commit from 2008, connectivity now at TCP level

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!
5. Fixed!

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!
5. Fixed!
6. Oops! Rejecting connections after hard reboot (or firewall)

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!
5. Fixed!
6. Oops! Rejecting connections after hard reboot (or firewall)
7. Fixed!

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!
5. Fixed!
6. Oops! Rejecting connections after hard reboot (or firewall)
7. Fixed!
8. Code now much cleaner and comprehensible

TCP connectivity

Connectivity problems when starting lots of daemons

- Found that connectivity depended on receiving packets
- Packets not always sent

1. Reverted a commit from 2008, connectivity now at TCP level
2. Noticed timeouts at startup
3. Replies dropped when only connected in one direction
4. Oops! Orphaning incoming queue!
5. Fixed!
6. Oops! Rejecting connections after hard reboot (or firewall)
7. Fixed!
8. Code now much cleaner and comprehensible
9. Contributors: Amitay, Martin, Noel, Ralph, Volker

Cluster mutex (aka recovery lock)

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have "elected before connected" problems...

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have “elected before connected” problems. . .
- Add leader broadcast

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have “elected before connected” problems. . .
- Add leader broadcast
- Wait a few seconds for leader broadcast

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have "elected before connected" problems...
- Add leader broadcast
- Wait a few seconds for leader broadcast
- Take **cluster lock** on election win, instead of **recovery lock**

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have "elected before connected" problems...
- Add leader broadcast
- Wait a few seconds for leader broadcast
- Take **cluster lock** on election win, instead of **recovery lock**
- Race for cluster lock in place of election

Cluster mutex (aka recovery lock)

- Event script tested for existence of recovery lock
- What if inode number changes?
- Add a periodic recheck (default 5s) to fcntl helper
- Tests...

To do:

- Have "elected before connected" problems...
- Add leader broadcast
- Wait a few seconds for leader broadcast
- Take **cluster lock** on election win, instead of **recovery lock**
- Race for cluster lock in place of election
- Experimental branch, passes tests...

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

"Invalidate database records when a node becomes inactive"

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated
5. Recovery completes and clears the “invalid records” flags

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

"Invalidate database records when a node becomes inactive"

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated
5. Recovery completes and clears the "invalid records" flags
6. Node becomes active

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

"Invalidate database records when a node becomes inactive"

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated
5. Recovery completes and clears the "invalid records" flags
6. Node becomes active
7. Recovery includes database contents from inactive node. . .

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated
5. Recovery completes and clears the “invalid records” flags
6. Node becomes active
7. Recovery includes database contents from inactive node. . .
8. . . . resurrecting any records that have since been deleted!

Recovery fixes — bug #14294

Issue #1: Resurrection of deleted records during recovery

“Invalidate database records when a node becomes inactive”

1. Node becomes inactive (e.g. `ctdb stop`)
2. Recovery starts before database records are invalidated
3. Recovery begins, including inactive node
4. Meanwhile database records are invalidated
5. Recovery completes and clears the “invalid records” flags
6. Node becomes active
7. Recovery includes database contents from inactive node. . .
8. . . . resurrecting any records that have since been deleted!
9. Fixed by confirming flags of remote nodes and dropping inactive nodes from recovery

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon
2. Databases frozen in recovery helper

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon
2. Databases frozen in recovery helper
   - Potentially with different sets of nodes

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon
2. Databases frozen in recovery helper
   - Potentially with different sets of nodes
   - So late-joining nodes might not have all databases!

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon
2. Databases frozen in recovery helper
   - Potentially with different sets of nodes
   - So late-joining nodes might not have all databases!
3. The result is that late-joining nodes can be banned

Recovery fixes — bug #14294 (continued)

Issue #2: Unwanted node banning

A hangover from supporting both serial & parallel recovery

1. Missing databases attached by recovery daemon
2. Databases frozen in recovery helper
   - Potentially with different sets of nodes
   - So late-joining nodes might not have all databases!
3. The result is that late-joining nodes can be banned
4. Fixed by moving attach of missing databases into recovery helper

Hot records

- *Hot records* are those that have been migrated the most times in a 1 second period

Hot records

- *Hot records* are those that have been migrated the most times in a 1 second period
- 10 hottest records recorded as a per-database statistic

Hot records

- *Hot records* are those that have been migrated the most times in a 1 second period
- 10 hottest records recorded as a per-database statistic
- Recording of hot records database statistics was broken
- Logging of hot records was not useful

Hot records

- *Hot records* are those that have been migrated the most times in a 1 second period
- 10 hottest records recorded as a per-database statistic
- Recording of hot records database statistics was broken
- Logging of hot records was not useful
- Fixed. . .

Plans

Separate daemons

- event daemon
- service daemon
- failover daemon + connection tracking daemon
- cluster daemon
- database daemon
- transport
- smbd proxy
- . . .

Design ideas

Design ideas

New abstractions

- `tdaemon`
- `tclient?`

Design ideas

New abstractions

- `tdaemon`
- `tclient?`

New daemons

- `masterd`
- `transportd`

Design ideas

New abstractions

- `tdaemon`
- `tclient?`

New daemons

- `masterd`
- `transportd`

Flaws in previous design

- Too many sockets
- Protocol inconsistencies
- Too much copy/paste of code
- Complicated test set up

Too many sockets

Too many sockets

- Each daemon had a unix domain socket

Too many sockets

- Each daemon had a unix domain socket
- Management of client connections (`tdaemon`)

Too many sockets

- Each daemon had a unix domain socket
- Management of client connections (`tdaemon`)

- Unix datagram messaging

Too many sockets

- Each daemon had a unix domain socket
- Management of client connections (`tdaemon`)

- Unix datagram messaging
- Transport API

Too many sockets

- Each daemon had a unix domain socket
- Management of client connections (`tdaemon`)

- Unix datagram messaging
- Transport API
- Problem solved!

Protocol inconsistencies

Protocol inconsistencies

- Separate protocol header for each daemon

Protocol inconsistencies

- Separate protocol header for each daemon
- New protocol

Protocol inconsistencies

- Separate protocol header for each daemon
- New protocol

```
    struct transport_header {
        uint32_t length;
        uint32_t magic;
        uint16_t protocol_version;
        uint16_t payload_version;
        struct transport_endpoint dst;
        struct transport_endpoint src;
        uint32_t flags;
        uint32_t reqid;
    }
```

Protocol inconsistencies

- Separate protocol header for each daemon

- New protocol
- Design it right from beginning – endian neutral

```
    struct transport_header {
        uint32_t length;
        uint32_t magic;
        uint16_t protocol_version;
        uint16_t payload_version;
        struct transport_endpoint dst;
        struct transport_endpoint src;
        uint32_t flags;
        uint32_t reqid;
    }
```

Protocol inconsistencies

- Separate protocol header for each daemon

- New protocol
- Design it right from beginning – endian neutral
- Proper marshalling API (`struct transport_packet`)

```
    struct transport_header {
        uint32_t length;
        uint32_t magic;
        uint16_t protocol_version;
        uint16_t payload_version;
        struct transport_endpoint dst;
        struct transport_endpoint src;
        uint32_t flags;
        uint32_t reqid;
    }
```

Too much copy/paste of code

Too much copy/paste of code

- `tdaemon` – designed around Unix domain sockets

Too much copy/paste of code

- `tdaemon` – designed around Unix domain sockets
- Handled connections, clients

Too much copy/paste of code

- `tdaemon` – designed around Unix domain sockets
- Handled connections, clients
- Still required lot of code for protocol handling

Too much copy/paste of code

- `tdaemon` – designed around Unix domain sockets
- Handled connections, clients
- Still required lot of code for protocol handling
- Needs more thought . . .

Complicated test set up

Complicated test set up

- Multiple daemons needed to be running for testing

Complicated test set up

- Multiple daemons needed to be running for testing
- Treat each daemon as a computation

Complicated test set up

- Multiple daemons needed to be running for testing
- Treat each daemon as a computation
- Each daemon code as a shared library

Complicated test set up

- Multiple daemons needed to be running for testing
- Treat each daemon as a computation
- Each daemon code as a shared library
- `masterd`

Complicated test set up

- Multiple daemons needed to be running for testing
- Treat each daemon as a computation
- Each daemon code as a shared library
- `masterd`
- Single process vs forked processes model

Complicated test set up

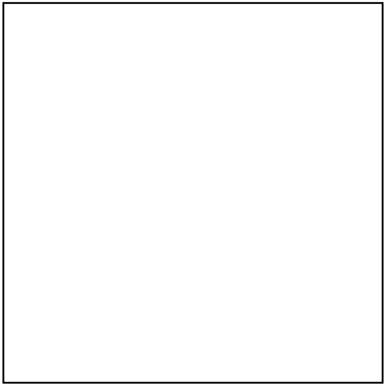
- Multiple daemons needed to be running for testing
- Treat each daemon as a computation
- Each daemon code as a shared library
- `masterd`
- Single process vs forked processes model
- Build / dependency issues

Complicated test set up

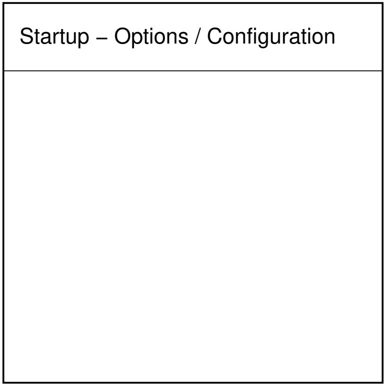
- Multiple daemons needed to be running for testing
- Treat each daemon as a computation
- Each daemon code as a shared library
- `masterd`
- Single process vs forked processes model
- Build / dependency issues
- Need better solution ...

What's in a daemon

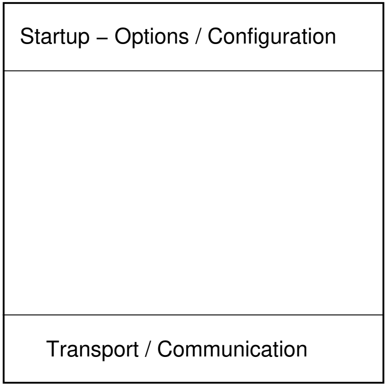
What's in a daemon



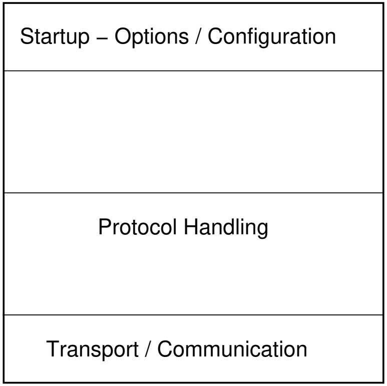
What’s in a daemon



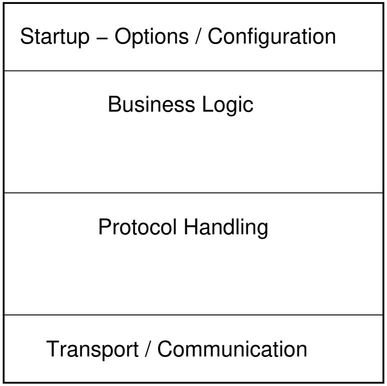
What's in a daemon



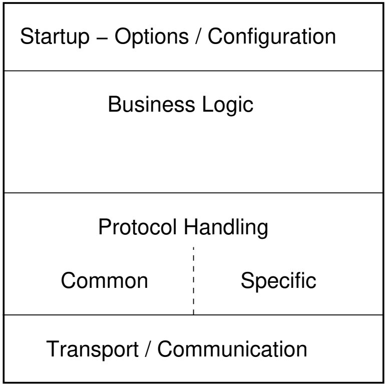
What's in a daemon



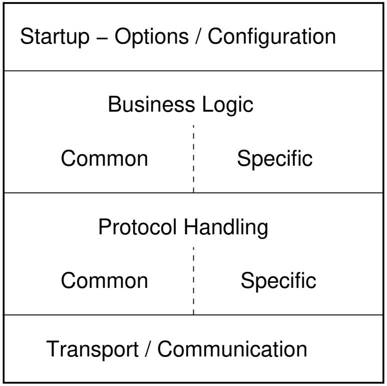
What's in a daemon



What's in a daemon

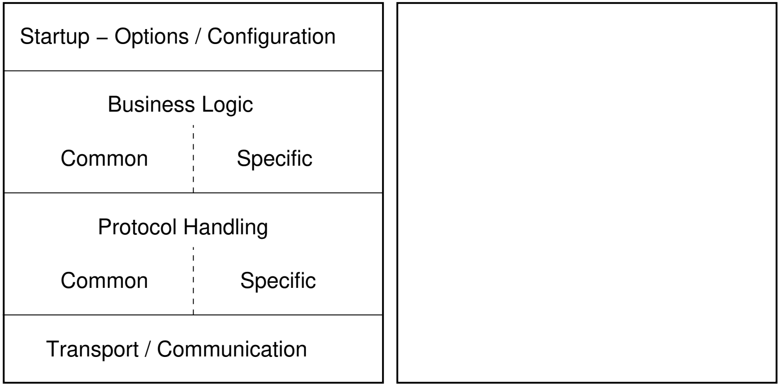


What's in a daemon

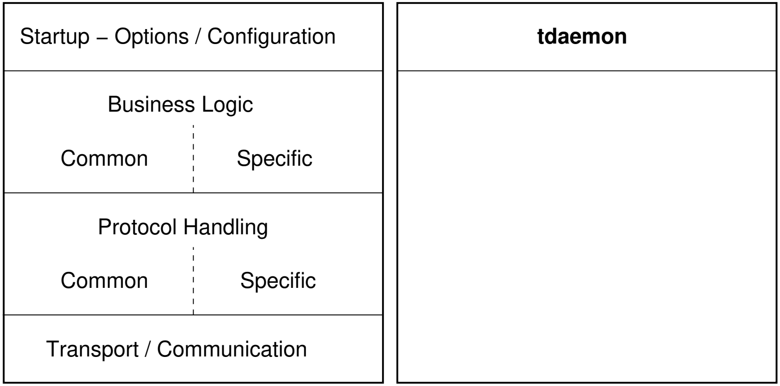


Structuring a daemon

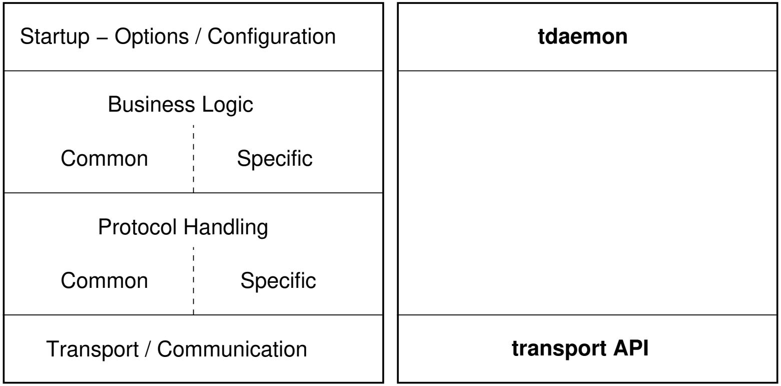
Structuring a daemon



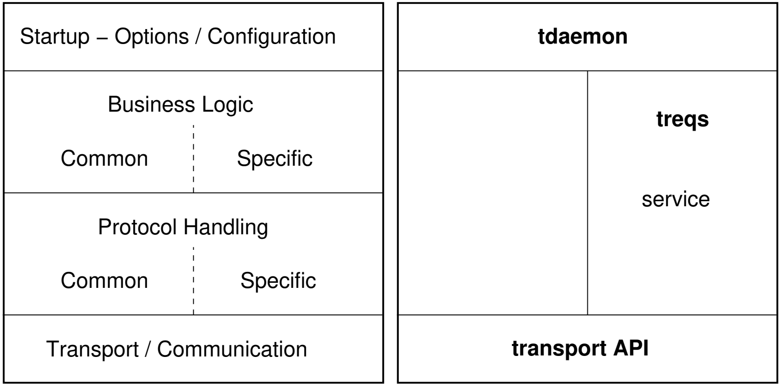
Structuring a daemon



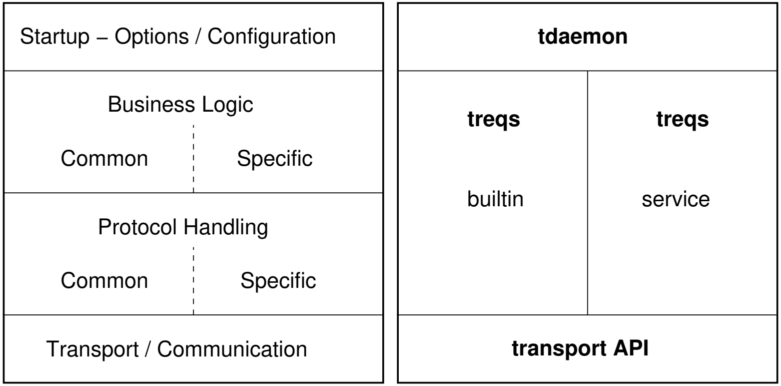
Structuring a daemon



Structuring a daemon



Structuring a daemon



Code: transport daemon

Code: transport daemon

```
int main(int argc, const char **argv)
{
    struct tdaemon *daemons[2];

    daemons[0] = transport_tdaemon();
    daemons[1] = NULL;

    return tdaemon_main(argc, argv, daemons);
}
```

tdaemon abstraction

tdaemon abstraction

- One daemon (abstraction) to rule them all

tdaemon abstraction

- One daemon (abstraction) to rule them all
- Single process and forked processes model

tdaemon abstraction

- One daemon (abstraction) to rule them all
- Single process and forked processes model

```
static struct tdaemon _transport_tdaemon;

struct tdaemon *transport_tdaemon(void)
{
    _transport_tdaemon = (struct tdaemon) {
        .name = "transportd",
        .endpoint_id = CTDB_ENDPOINT_TRANSPORT,
        .builtin = builtin_treqs(),
        .service = transport_treqs(),
    };
    return &_transport_tdaemon;
}
```

tdaemon abstraction

- One daemon (abstraction) to rule them all
- Single process and forked processes model
- Actual business logic is separated into backends (`treqs`)

```
static struct tdaemon _transport_tdaemon;

struct tdaemon *transport_tdaemon(void)
{
    _transport_tdaemon = (struct tdaemon) {
        .name = "transportd",
        .endpoint_id = CTDB_ENDPOINT_TRANSPORT,
        .builtin = builtin_treqs(),
        .service = transport_treqs(),
    };
    return &_transport_tdaemon;
}
```

treqs abstraction

treqs abstraction

- Encapsulate business logic for a service

treqs abstraction

- Encapsulate business logic for a service
- Independent of transport

treqs abstraction

- Encapsulate business logic for a service
- Independent of transport

```
struct treqs {
    struct tevent_req * (*init_send)();
    bool (*init_recv)();
    struct tevent_req * (*reconfigure_send)();
    bool (*reconfigure_recv)();
    struct tevent_req * (*activate_send)();
    bool (*activate_recv)();
    struct tevent_req * (*deactivate_send)();
    bool (*deactivate_recv)();
    bool (*command_match)();
    struct tevent_req * (*dispatch_send)();
    bool (*dispatch_recv)();
    struct tevent_req * (*run_send)();
    bool (*run_recv)();
};
```

service (treqs) backend

service (treqs) backend

- Implement service specific logic

service (treqs) backend

- Implement service specific logic
- Protocol handling, main loop

service (treqs) backend

- Implement service specific logic
- Protocol handling, main loop

```
    static struct treqs _transport_treqs = {
        .init_send = transport_backend_init_send,
        .init_recv = transport_backend_init_recv,
        .reconfigure_send = transport_backend_reconfigure_send,
        .reconfigure_recv = transport_backend_reconfigure_recv,
        .command_match = transport_backend_command_match,
        .dispatch_send = transport_backend_dispatch_send,
        .dispatch_recv = transport_backend_dispatch_recv,
        .run_send = transport_backend_run_send,
        .run_recv = transport_backend_run_recv,
    };
```

builtin (treqs) backend

builtin (treqs) backend

- Built-in (common) request handling for all daemons

builtin (treqs) backend

- Built-in (common) request handling for all daemons
- ping, (de)activate, debug, memory use, . . .

Way ahead

Way ahead

What's missing

Way ahead

What's missing

- Integrating config handling
- Sorting out single process transport
- Test infrastructure changes?

Way ahead

What's missing

- Integrating config handling
- Sorting out single process transport
- Test infrastructure changes?

What's next

Way ahead

What's missing

- Integrating config handling
- Sorting out single process transport
- Test infrastructure changes?

What's next

- `transport` done; testing ...
- Implement other daemons - `cluster`, `event`, ...

Forward?

Incremental progress?

Incremental progress?

- Retro-fitting current `ctdbd` to use new transport API is not a sane option

Incremental progress?

- Retro-fitting current `ctdbd` to use new transport API is not a sane option
- Should we implement the transport API against existing `ctdbd` (i.e. use existing `ctdbd` as transport)?

Incremental progress?

- Retro-fitting current `ctdbd` to use new transport API is not a sane option
- Should we implement the transport API against existing `ctdbd` (i.e. use existing `ctdbd` as transport)?
- This is churn but potentially lets us get some of our work into a release before everything is finished

Incremental progress?

- Retro-fitting current `ctdbd` to use new transport API is not a sane option
- Should we implement the transport API against existing `ctdbd` (i.e. use existing `ctdbd` as transport)?
- This is churn but potentially lets us get some of our work into a release before everything is finished
- Maybe this is worth doing. . .

CTDB developers needed

CTDB developers needed

- Samba Team has zero full time CTDB developers

CTDB developers needed

- Samba Team has zero full time CTDB developers
- Some amount of burnout. . .

CTDB developers needed

- Samba Team has zero full time CTDB developers
- Some amount of burnout. . .
- Any volunteers?

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.

Questions?