Dora Stop Hang - Root Cause Analysis

Summary

When running `dora stop` on a dataflow with dynamic nodes, the command hangs indefinitely. This document analyzes the root cause and provides recommendations.

Environment

- dora-rs version: 0.4.0
- Platform: macOS (Darwin 24.6.0)
- Mode: Local (`dora up` + `dora start`)

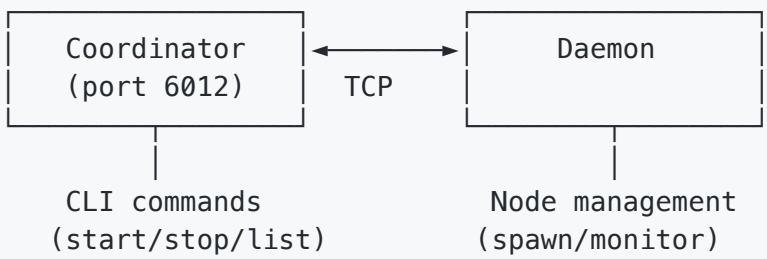
Symptoms

```
$ dora start voice-chat.yml --detach
$ dora stop <uuid>
# Hangs forever...
```

Observations:

- Daemon shows "Running" for ~30 seconds, then "Not running"
- Coordinator remains running
- `dora stop` never returns

Architecture Overview



Two TCP connections between coordinator and daemon:

1. **Connection #1**: Coordinator → Daemon events + Daemon → Coordinator replies

2. **Connection #2**: Daemon → Coordinator events (AllNodesFinished, etc.)

Root Cause

Primary Issue: Heartbeat Timeout False Positive

Location: `dora-daemon-0.4.0/src/lib.rs:444-445`

```
if self.last_coordinator_heartbeat.elapsed() > Duration::from_secs(20) {
    bail!("lost connection to coordinator")
}
```

The daemon has a 20-second heartbeat timeout. However, heartbeats can be delayed due to **single-threaded event processing** in the coordinator.rs task:

```
// dora-daemon-0.4.0/src/coordinator.rs:73-137
loop {
    let event = socket_stream_receive(&mut stream).await;  // 1. Receive
    tx.send(event).await;                                   // 2. Send to
daemon
    reply_rx.await;                                         // 3. BLOCKED
waiting
    socket_stream_send(reply).await;                        // 4. Send reply
}
```

While blocked at step 3 (waiting for daemon to process StopDataflow), **heartbeats cannot be received**.

Timeline of Failure

```
T+0s     Coordinator sends StopDataflow
         coordinator.rs task blocked waiting for reply

T+0-15s  Daemon processing node stop events (SpawnedNodeResult)
         Heartbeats buffered in TCP, not processed

T+15s    Grace period ends, nodes killed
         StopDataflow finally processed, reply sent

T+20s    Daemon checks heartbeat
         last_coordinator_heartbeat.elapsed() > 20s
         bail!("lost connection to coordinator")
         DAEMON EXITS

T+20s+   Coordinator waiting for DataflowFinishedOnDaemon
         Daemon is dead, event never arrives
         dora stop HANGS FOREVER
```

Secondary Issue: No Timeout on tcp_receive

Location: `dora-coordinator-0.4.0/src/lib.rs:1235`

```rust
let reply_raw = tcp_receive(&mut daemon_connection.stream)
    .await  // NO TIMEOUT - blocks forever if daemon dies
    .wrap_err("failed to receive stop reply from daemon")?;
```

Location: `dora-coordinator-0.4.0/src/tcp_utils.rs:14-23`

```rust
pub async fn tcp_receive(connection: &mut TcpStream) ->
std::io::Result<Vec<u8>> {
    // No timeout wrapper!
    connection.read_exact(&mut raw).await?;
    // ...
}
```

Tertiary Issue: Orphaned Reply Senders

Location: `dora-coordinator-0.4.0/src/lib.rs:603`

```rust
dataflow.stop_reply_senders.push(reply_sender);
```

When daemon dies before sending `AllNodesFinished`, the `stop_reply_senders` are never answered, causing `dora stop` to hang.

Dynamic Nodes Behavior

The error messages for dynamic nodes are **expected**, not bugs:

```
ERROR   mofa-audio-player: daemon  failed to receive finished signal
ERROR   mofa-mic-input: daemon  failed to receive finished signal
```

Location: `dora-daemon-0.4.0/src/spawn/prepared.rs:110-118`

Dynamic nodes (`path: dynamic`) are placeholders waiting for external connection (e.g., MoFA Studio UI). The "failed to receive finished signal" occurs because:

1. No process is spawned for dynamic nodes
2. The `finished_tx` channel is dropped immediately
3. `finished_rx.await` returns `Err`
4. Error is logged, restart_loop exits

This is correct behavior - dynamic nodes shouldn't have a process until connected.

Recommendations

Short-term Fixes (dora-rs patches needed)

1. Add timeout to tcp_receive

```rust
// dora-coordinator-0.4.0/src/lib.rs
use tokio::time::{timeout, Duration};

let reply_raw = timeout(
    Duration::from_secs(60),
    tcp_receive(&mut daemon_connection.stream)
).await
    .wrap_err("timeout waiting for daemon reply")?
    .wrap_err("failed to receive stop reply from daemon")?;
```

2. Handle daemon disconnect in stop flow

```
// When daemon connection closes, notify stop_reply_senders
for sender in dataflow.stop_reply_senders.drain(..) {
    let _ = sender.send(Err(eyre!("daemon disconnected")));
}
```

3. Fix heartbeat during blocked reply wait

Option A: Process heartbeats in separate task

```
// Split the coordinator.rs task into two:
// 1. Receive task: receives events, queues them
// 2. Process task: sends to daemon, waits for reply
```

Option B: Adaptive heartbeat timeout

```
// Track when we're waiting for a reply
if waiting_for_reply {
    // Extend timeout or skip check
}
```

Long-term Recommendations

1. Use Gossip Protocol for Failure Detection

Replace simple heartbeat with SWIM-style protocol:

- Indirect probing through peers
- Suspicion mechanism before declaring dead
- Reduces false positives significantly

2. Simplify Local Mode

For single-machine deployments, the coordinator/daemon separation adds unnecessary complexity:

```
// Add a "local mode" that runs everything in-process
dora run --local voice-chat.yml
```

3. Implement Circuit Breaker for Stop

```
// If stop takes too long, fail fast instead of hanging
let result = timeout(Duration::from_secs(120), stop_dataflow(...)).await;
match result {
    Ok(Ok(_)) => println!("Stopped successfully"),
    Ok(Err(e)) => println!("Stop failed: {}", e),
    Err(_) => {
        println!("Stop timed out, force killing...");
        force_kill_dataflow(...);
    }
}
```

Workarounds (Current Version)

1. Use `dora destroy` instead of `dora stop`

```
# Instead of:
dora stop <uuid>

# Use:
dora destroy  # Kills everything
```

2. Manual cleanup when hung

```
# Kill hung stop command
pkill -f "dora stop"

# Kill daemon and coordinator
pkill -f "dora daemon"
pkill -f "dora coordinator"

# Kill any orphaned node processes
pkill -f "dora-maas\|primespeech\|dora-asr"
```

3. Use `dora run` for local development (if no dynamic nodes)

```
# Simpler, no coordinator/daemon
dora run voice-chat.yml
```

Note: `dora run` doesn't support dynamic nodes.

4. Longer grace period

```
# Give more time for nodes to stop gracefully
dora stop --grace-duration 30s <uuid>
```

Files Analyzed

File	Description
`dora-daemon-0.4.0/src/lib.rs`	Daemon main loop, heartbeat check
`dora-daemon-0.4.0/src/coordinator.rs`	Daemon's coordinator connection task
`dora-daemon-0.4.0/src/spawn/prepared.rs`	Node spawning, dynamic node handling
`dora-daemon-0.4.0/src/pending.rs`	Pending node management
`dora-coordinator-0.4.0/src/lib.rs`	Coordinator main loop, stop handling
`dora-coordinator-0.4.0/src/tcp_utils.rs`	TCP send/receive (no timeout)

References

- dora-rs source: `~/.cargo/registry/src/index.crates.io-*/dora-*-0.4.0/`
- dora-rs GitHub: https://github.com/dora-rs/dora
- SWIM Protocol: https://www.cs.cornell.edu/projects/Quicksilver/public_pdfs/SWIM.pdf
- Phi Accrual Failure Detector: https://www.computer.org/csdl/proceedings-article/srds/2004/22390066/12OmNviEkRx