Contents

1. IPC for Multicore Tock

1.1 Multicore Tock

- Multikernel design
  - Each core runs a single Tock kernel instance
    - Each kernel instance lives in a ***separate address space***
  - Communication is through message passing between kernel instances
    - ***Only*** shared memory is the message channel

1.2 IPC Design Constraints

1. Asynchrony
    - Inter-kernel communication is asynchronous
    - `ipc_discover(...)` is synchronous
2. Service/client identifier
    - Kernel-instance-aware
3. No memory sharing between kernel instances
    - Except for inter-kernel message channel
4. The other kernel instance may fail
    - Avoid poisoning and fate-sharing

1.3 IPC Interface (Message Passing)

- `int ipc_ng_discover_sync(char*, id_t*) // Discover service`
- `int ipc_ng_self(id_t*); // Global Process ID`
- `int ipc_ng_register_rx_callback_from(id_t, cb, buf_t*) // Register callback for clients/as service`
- `int ipc_ng_tx_to_service(id_t, buf_t*, len_t)`
- `int ipc_ng_tx_to_client(id_t, buf_t*, len_t)`
- `int ipc_ng_swap_receive_buffer(...)`
- (Not implemented) `int ipc_ng_tx_with_timeout(...)`

1.4 IPC Implementation

- Message passing
  - Require dedicated tx and rx buffers
  - 1 copy for intra-kernel IPC
  - 2 copies for inter-kernel IPC
- Message passing by reference
  - Limited by MPUs
  - No RX buffers, 0 copy, no poisoning and fate-sharing

1.5 Summary

- Asynchrony
- Message passing
- Timeout