```
// We will have four worker processes
#define NUM_WORKERS 4

// An array of packet channel send ports for sending work descriptors
// to each of the worker processes.
mcapi_port_t work_requests_out_port[NUM_WORKERS];

// An array of scalar channel receive ports for getting acks back from
// the workers.
mcapi_port_t acks_in_port[NUM_WORKERS];

mcapi_pktchan_send_hndl_t work_requests_out_hndl[NUM_WORKERS];
mcapi_pktchan_recv_hndl_t acks_in_hndl[NUM_WORKERS];


// function declarations
void create_and_init_workers(void);
void dispatch_packets(void);
void shutdown_lb(void);

// The entrypoint for this packet processing application.
int main(void)
{
      mcapi_param_t mcapi_parameters;
      mcapi_info_t mcapi_info;
      mcapi_status_t status;

      mcapi_initialize(DOMAIN_0, NODE_LOAD_BALANCER, &mcapi_parameters,
&mcapi_info, &status);

  create_and_init_workers();
  dispatch_packets();
  shutdown_lb();

  return 0;
}


void create_and_init_workers()
{
      int i;
      mcapi_request_t request;

  for (i = 0; i < NUM_WORKERS; i++)
  {
    mcapi_status_t status;

    // Spawn a new thread; pass parameters so the new thread will execute
    // worker_spawn_function(bootstrap_endpoint)
    spawn_new_thread(&worker_spawn_function, i);
    int worker_node = i + 1;

    // Create a send endpoint to send packets to the worker; get the
    // worker's receive endpoint via mcapi_endpoint_get()
    mcapi_endpoint_t work_request_out_endpoint =
mcapi_endpoint_create(MCAPI_PORT_ANY, &status);
    CHECK_STATUS(status);
    mcapi_endpoint_t work_request_remote_endpoint =
mcapi_endpoint_get(DOMAIN_0, worker_node, WORKER_REQUEST_PORT_ID, &status);
    CHECK_STATUS(status);
```