```
void worker_spawn_function(int worker_num)
{
      mcapi_param_t mcapi_parameters;
      mcapi_info_t mcapi_info;
      mcapi_status_t status;

      mcapi_initialize(DOMAIN_0, (1 + worker_num), &mcapi_parameters,
&mcapi_info, &status);

  bind_channels();
  do_work();
  shutdown();
}

void bind_channels()
{
  mcapi_status_t status;
      mcapi_request_t request;
      mcapi_pktchan_recv_hndl_t work_request_in;
      mcapi_sclchan_send_hndl_t ack_out;

  // Create a packet receive endpoint for incoming work requests; use
  // a static port number so that load balancer can look it up via
  // mcapi_endpoint_get().
  mcapi_endpoint_t work_request_in_endpoint =
mcapi_endpoint_create(WORKER_REQUEST_PORT_ID, &status);
  CHECK_STATUS(status);

  // The load balancer will bind a channel to that endpoint; we use
  // the open function to synchronize and initialize our local packet
  // receive port.
  mcapi_pktchan_recv_open_i(&work_request_in, work_request_in_endpoint,
&request, &status);
  CHECK_STATUS(status);

  // Repeat the process to create a scalar channel from us back to the
  // load balancer.
  mcapi_endpoint_t ack_out_endpoint =
mcapi_endpoint_create(WORKER_ACK_PORT_ID, &status);
  CHECK_STATUS(status);

  mcapi_sclchan_send_open_i(&ack_out, ack_out_endpoint, &request, &status);
  CHECK_STATUS(status);
}


void do_work()
{
  mcapi_status_t status;
  size_t size;

  // The first thing we do is send an ack so that the load balancer
  // knows we're ready for work.
  mcapi_sclchan_send_uint8(ack_out, 1 /* Any value would do. */, &status);

  while (1)
  {
    // receive a work request from the load balancer
    PacketInfo* work_info;
    mcapi_pktchan_recv(work_request_in, (void*)&work_info, &size, &status);
    CHECK_STATUS(status);
```