```
        mcapi_pktchan_send(data_send_chan, result_buffer, result_size,
                           &status);
        CHECK_STATUS(status);
      break;
      case DSP_TERMINATE:
        dsp_shutdown();
      break;
      default:
      break;
    }
    command = receive_dsp_cmd();
  }
}
```

7.5 Packet Processing

This example presents the typical startup and inner loop of a packet processing application. There are two source files: load_balancer.c and worker.c. The main entrypoint is in load_balancer.c.

The program begins in the load balancer, which spawns a set of worker processes and binds channels to them. Each worker has two channel connections to the load balancer: a packet channel for work requests and a scalar channel for acks. When work arrives on the packet channel from the load balancer, the worker processes it and then sends back an ack to the load balancer. The load balancer will not send new work to a worker unless an “ack” word is available.

This example uses both packet channels and scalar channels. Scalar channels are used for acks from the workers to the load balancer, since each ack is a single word and performance will be better without the packetization overhead. Packet channels are used to send work requests to the workers, since each work request is a structure containing multiple fields. The example also shows how to use both statically named and dynamic (anonymous) endpoints; it uses the endpoint creation functions mcapi_endpoint_create(), mcapi_create_anonymous_endpoint(), and mcapi_endpoint_get().

On architectures with hardware support for scalars it would make more sense to use scalar channels for the work requests as well. Work requests are usually composed of only a few words, so they can be reasonably sent as individual words. More importantly, the load balancer is the rate-limiting step in this program; we can always add more workers but they're useless if the load balancer can't feed them. Since the load balancer is limited by communication overhead, hardware accelerated scalar channels could potentially improve overall performance if the overhead of a packet channel send is greater than the comparable scalar channel sends.