```
	}

	for(int i=0; i<next_hndl; i++)
	{
		mrapi_rmem_detach(handles[i], &status);

		// ERROR CHECKING ON status NOT SHOWN

	}

	send_message_to_node_1(QUIT);

	return 0;

}
```

6.4 Synchronization Use Case

TI has several chips that have a General Purpose Processor (GPP) and a DSP. The GPP traditionally runs a higher level RTOS like Linux, QNX, WinCE, etc. The DSP traditionally runs a DSP/BIOS.

One typical use case is to use the DSP as a video/audio accelerator. The GPP sends rmem processor call (RCP) messages to the DSP. An RCP message contains a pointer to the data to process, type and size of the data, how to process, etc. Once it is finished, the DSP sends a message back to complete the GPP RCP call.

The typical size of an RCP message is 4K bytes. The throughput is ~100 messages per second both ways (30 frames/second for video and 30-50 frames/second for audio). A typical system has ~64 messages in a system.

Generally the communication between the processors is either shared memory and interrupts or specialized hardware mechanisms. In the case of shared memory, the chips must support the same type of synchronization mechanism. The typical mechanisms include: spinlocks, hardware semaphores, support for Peterson's exclusion algorithm, and a few more.

Typically the GPP is the master and controls the starting/stopping of the DSP. All communication mechanisms must support the stopping of one side. The communication and synchronization mechanisms must also be portable to allow the easy migration of code to a different processor and OS.

6.5 Networking use case

Packet Processing Usage Case Patrick Griffin, Tilera

This use case extends the packet processing use case from the MCAPI specification to take advantage of MRAPI functions.

MRAPI's support for shared memory and mutexes is used to create a shared memory 'flow state' table, which tracks information about groups of packets flowing between the same source and destination hosts.

Use Case Description:

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.