```
	mrapi_status_t status; /* For error checking */

	/* An array of remote memory handles */
	mrapi_hndl_t handles[MAX];

	int next_hndl = 0;

	while(...)
	{
		/* Node 2 does some processing which we do not specify here.
			Once in a while, Node 2 needs to use Node 1's memory as a
			"spill" buffer, thus requiring access to a region of this
			memory */

		...

		if( need to spill )
		{
			int number_of_bytes_required = ...;
			send_message_to_node_1(MORE_MEMORY_PLEASE);
			send_message_to_node_1(number_of_bytes_required);

			/* Node 1 will receive these messages and create some
				remotely accessible memory, for which it will send
an id */
			mrapi_rmem_id_t id = receive_id_from_node1();

			/* Use the id to get a handle for the remote memory */
			handles[next_hndl] = mrapi_rmem_get(id,
AGREED_ACCESS_TYPE,&status);

			// ERROR CHECKING ON status NOT SHOWN

			mrapi_rmem_attach(handles[next_hndl,
				AGREED_ACCESS_TYPE,
				&status);

			// ERROR CHECKING ON status NOT SHOWN

			next_hndl++;

			/* Now Node 2 can do some work using this remote memory,
via
				MRPI calls such as mrapi_rmem_read and
mrapi_rmem_write.
				We do not show details as this would be application-
specific
			*/
			...

		}
```