MPI3 Hybrid Programming – Proposal for Helper Threads

June 14 Meeting notes: The following items need further discussion

1 Error handling/reporting: how do various errors get reported and handled – are errors thrown to MPI_COMM_WORLD? Or to the LEAVE call? Or ??

1.1 The proposal has each function returning error codes, but there is also a need to define how error handlers are used with this.

1.1.1 Where does one register a handler for helper threads calls?

2 Stacking Libraries – nesting. How would JOIN/LEAVE be used with libraries?

2.1 All (non-helper) MPI calls are used the same regardless of whether inside a JOIN/LEAVE or not, the implementation detects the JOIN/LEAVE case and does things differently internally. This means that a communications-only library need not be aware of whether the caller is within a JOIN/LEAVE or not, provided the caller observes the library's restrictions on threading (i.e. only one thread calls the library if it is single-thread).

2.1.1 What other types of complexity exist in current libraries?

2.1.2 What do we expect library writers to do when using helper threads?

3 Leave semantics clarification – is it a (more restrictive) “barrier-like” operation or (less restrictive) “collective”?

3.1 “barrier-like” means that all team members must call LEAVE before any will return from LEAVE.

3.2 “collective” (only) would mean that simply all members must call LEAVE, but there is not requirement that all will be in the LEAVE at the same instant.

3.3 Commentary: I think the distinction comes for implementations or run modes where JOIN/LEAVE are essentially not supported. I think a “productive” LEAVE will actually be barrier-like, although one could imagine a case where the “master” (thread doing MPI calls) reaches LEAVE before others and finds all communications completed, it could continue and when other threads reach LEAVE they simply pass-through. But we should probably ponder this more to be sure that “barrier-like” is not required under some circumstances, or that an implementation is free to use “barrier-like” if needed. Is it acceptable to state that the user cannot expect more than “collective”, but also must tolerate “barrier-like”?

4 What about a “leave but stay joined” sort of operation, like a fence.

4.1 The idea being there might be a performance benefit if a program wants to “synchronize” communications at various points between JOIN and LEAVE, compared to having to do a full LEAVE/JOIN (and possible barrier) sequence.

4.1.1 What would such a call be named?

4.1.2 Would this be “barrier-like” or simply collective? i.e. how would it differ from LEAVE?