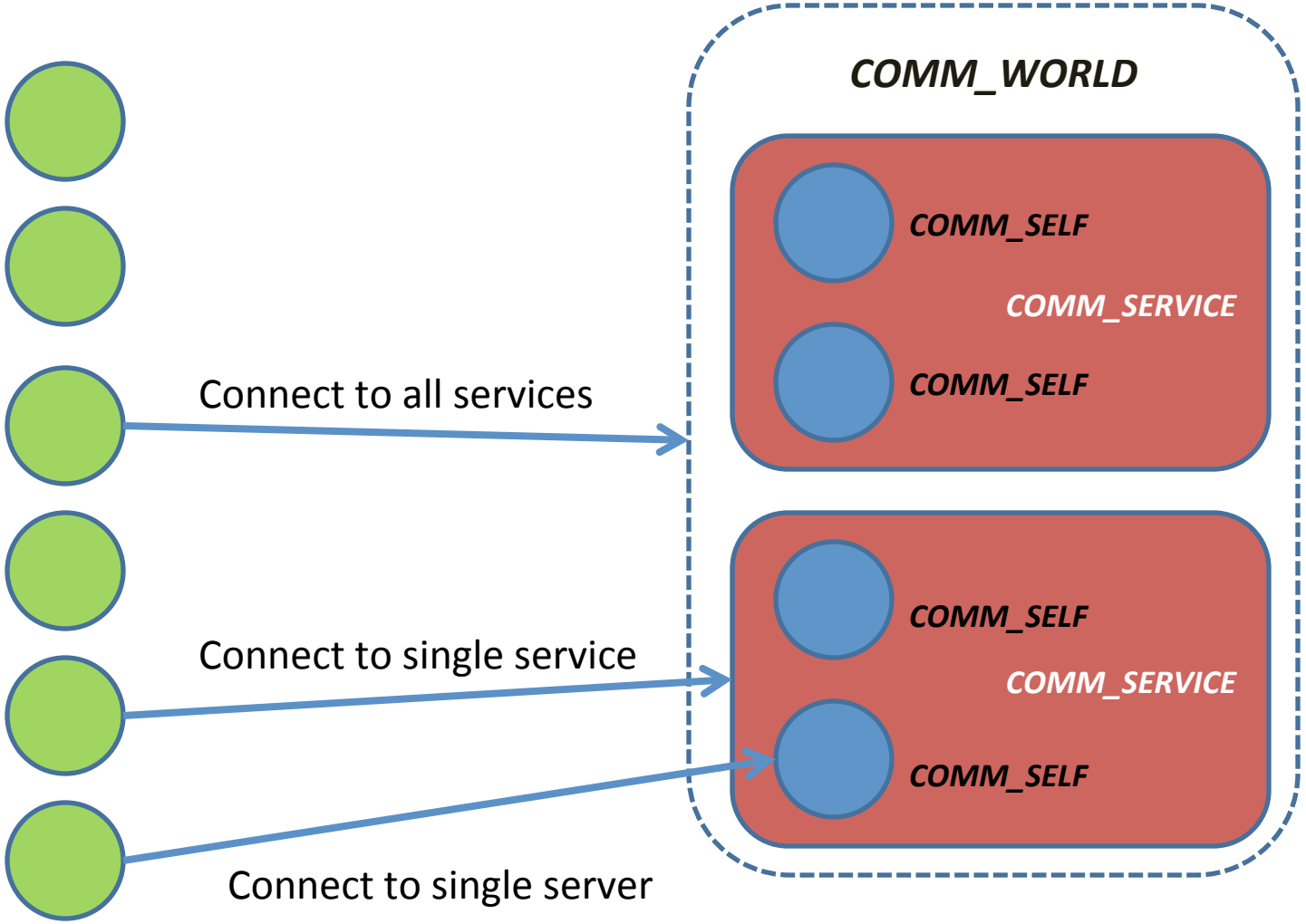
A Case for Nonblocking Operations as an Orthogonal Feature in MPI

Pavan Balaji

MPI-3 Status

- Nonblocking operations are not a truly first-class citizen in MPI (unlike communicators or datatypes (mostly))
- Some operations have nonblocking variants (SEND, RECV), while some don't (MPI_COMM_ACCEPT, MPI_WIN_FLUSH)

Server design with MPI (case study with VOCL virtualization framework)



VOCL infrastructure

- Connections can come in on COMM_SELF, COMM_SERVICE or COMM_WORLD, so the server has to post “Comm_accepts” on each of these communicators
  - Since there is no COMM_IACCEPT, we use a thread for each of these communicators
- Once a communicator is established, a window is created for each connecting client (currently limited to a few hundred)
  - The server occasionally posts RMA operations on client windows and “tests” whether they have remotely completed before informing other servers of the posted data
  - Since there is no WIN_IFLUSH or WIN_IUNLOCK or WIN_IFENCE operation, we use a thread for each of the windows (potentially a few hundred)
- This model is causing us to create a large number of threads for each MPI process (we use a SEND/RECV based workaround for the RMA part, but there is no workaround for the COMM_IACCEPT part, so we use at least 3 threads per MPI process)

Proposal

- Orthogonal Nonblocking operations for most (or even all?) operations
  - Request handler is critical, asynchronous progress might not be
- MPI_COMM_IACCEPT
- MPI_WIN_IFLUSH
- MPI_WIN_IUNLOCK
- …