MPI SESSIONS

Challenges of Local Only Sessions

- or –

A Motivation for Collective Sessions

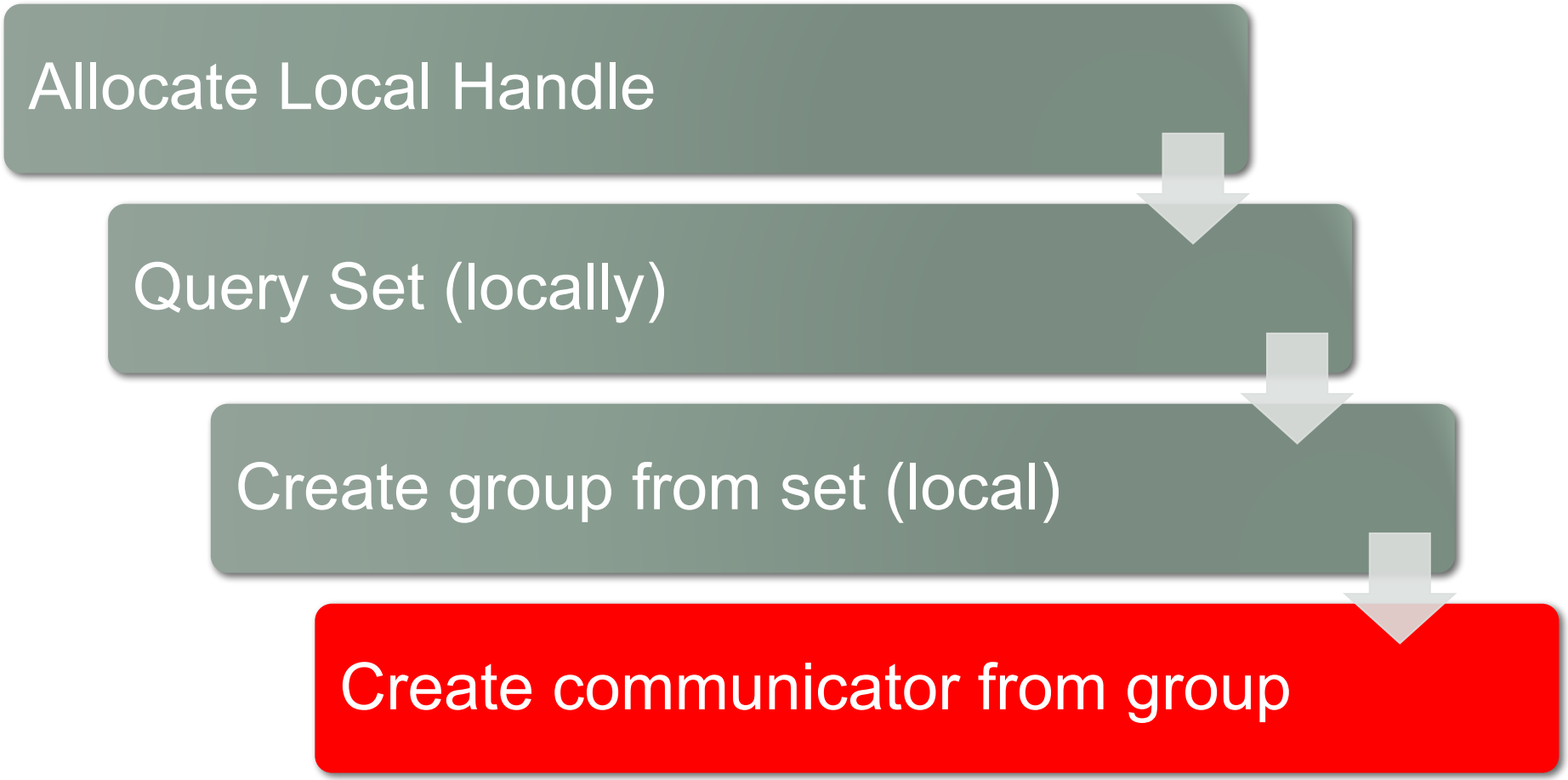
LLNL-PRES-738998

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

MPI Sessions – The Initial Idea

- **Goal 1: Solve the multi-init problem**
  - Enable libraries to do their own init
  - Enable dynamic init/un-init/re-init
  - Enable better composability
- **Goal 2: Solve the scalability problem**
  - Only initialize what you want/need
  - Don’t wire up all available resources
  - Adjust resources to actually needed communication
- **Goal 3: Step-wise elimination of MPI_Init**
  - As well as MPI_COMM_WORLD
  - Try to do what MPI should have done in the first place

Sessions Workflow



And then came the ideas …

- **Use of Sessions for multi-threading**
  - Different thread models for each sessions
  - Adjust thread model based on application/library needs
- **Use of Sessions for fault tolerance**
  - Fault isolation between sessions
  - Support for multiple FT approaches
- **Use of Sessions for dynamic applications**
  - Sessions could support ensemble computation
  - Sessions could support connecting multiple application components
- **Use of Sessions for Multi-MPI support**
  - Each session could run its own MPI library
  - Options for ABI problem or to connect multiple systems
- **Use of Sessions for in-situ analysis**
  - Requires wiring up second communication fabric
  - Visualization, I/O, Debuggers, Performance Analysis, …

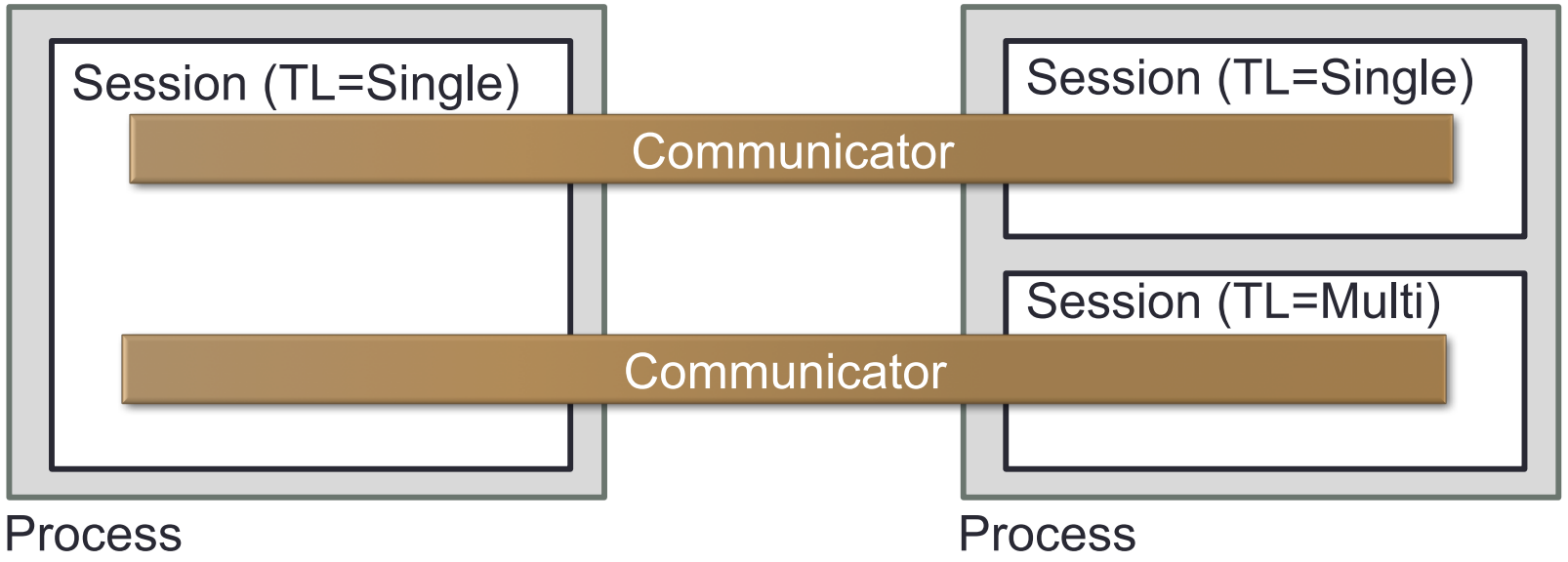
General Concept of Resource Isolation

- **A session encapsulates all resources of its MPI usage**
  - Sessions are independent
  - Sessions can be created and destroyed
  - No shared resources in the MPI library between sessions
- **Communication between sessions only on-node**
  - Within a single MPI process
- **Only exception are resources not derived from MPI_COMM_WORLD**
  - No ability to intercept them
  - Still need to figure out exactly what happens to them

Local Sessions No Longer Support This (or at least make it much harder)

- **Does not prevent inconsistent instantiations**
- **Does not guarantee fault isolation**
- **Unclear how tool stacks can be deployed**
- **Makes it hard to impossible to use with multiple MPIs**
- **Can lead to mixing of thread levels**

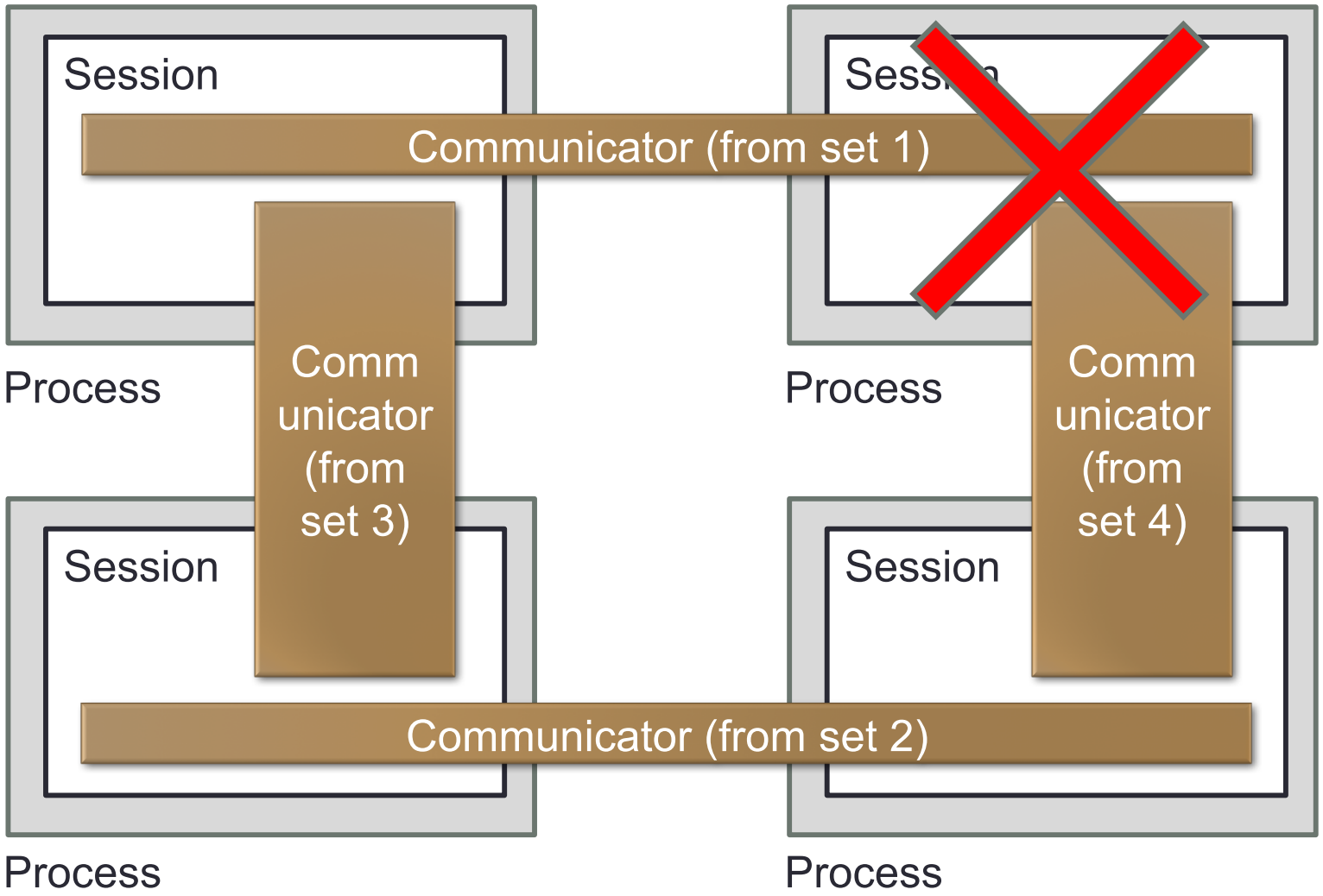
Issue: Inconsistent Instantiations



- **Breaks resource isolation**
  - Backdoor for two sessions to communicate directly
- **Prevents clean fault models**
  - Failure in one session kill second session

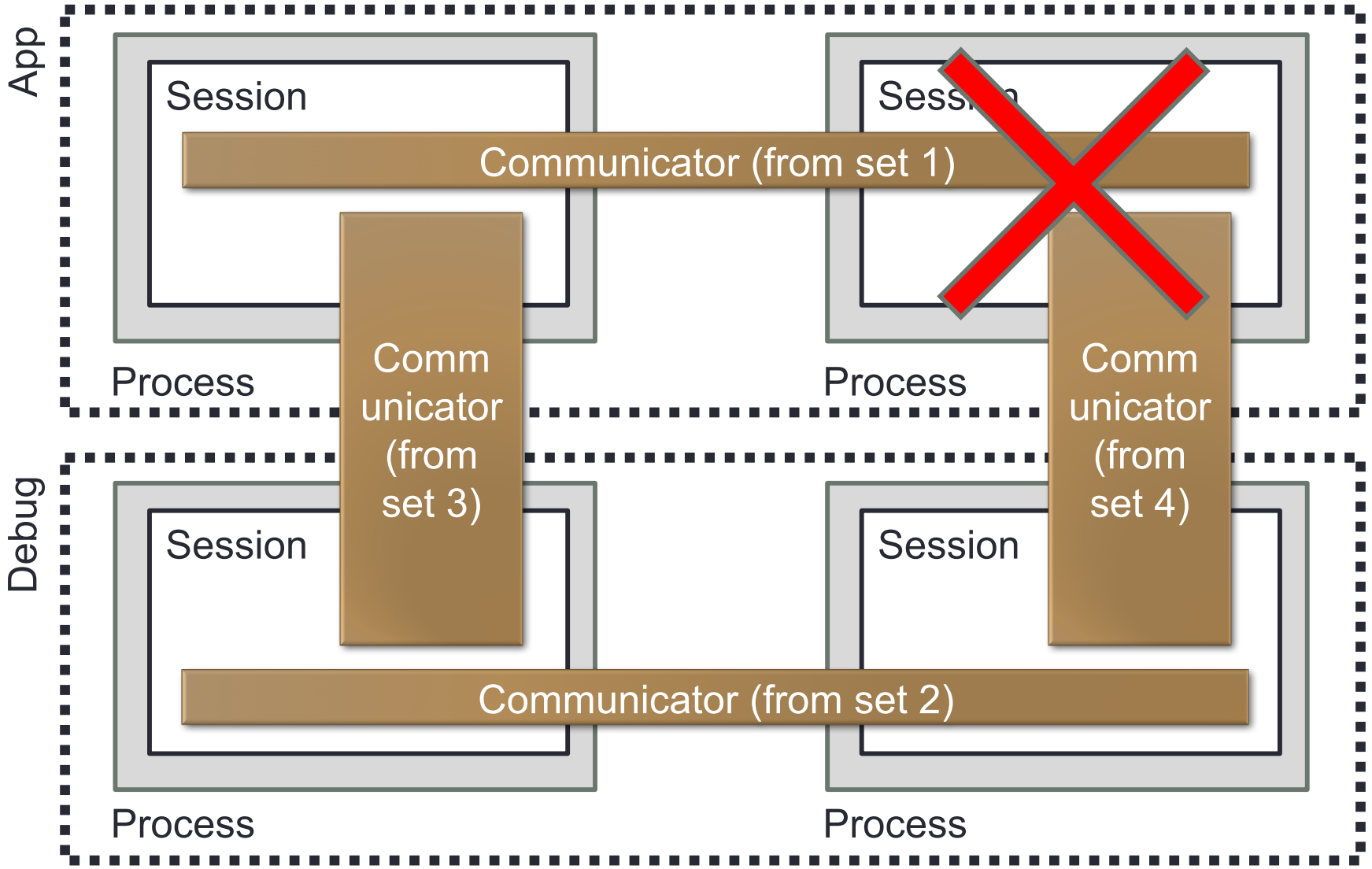
Issue: Fault Isolation Limited

(By allowing multiple sets/comms per session)



Issue: Fault Isolation Limited

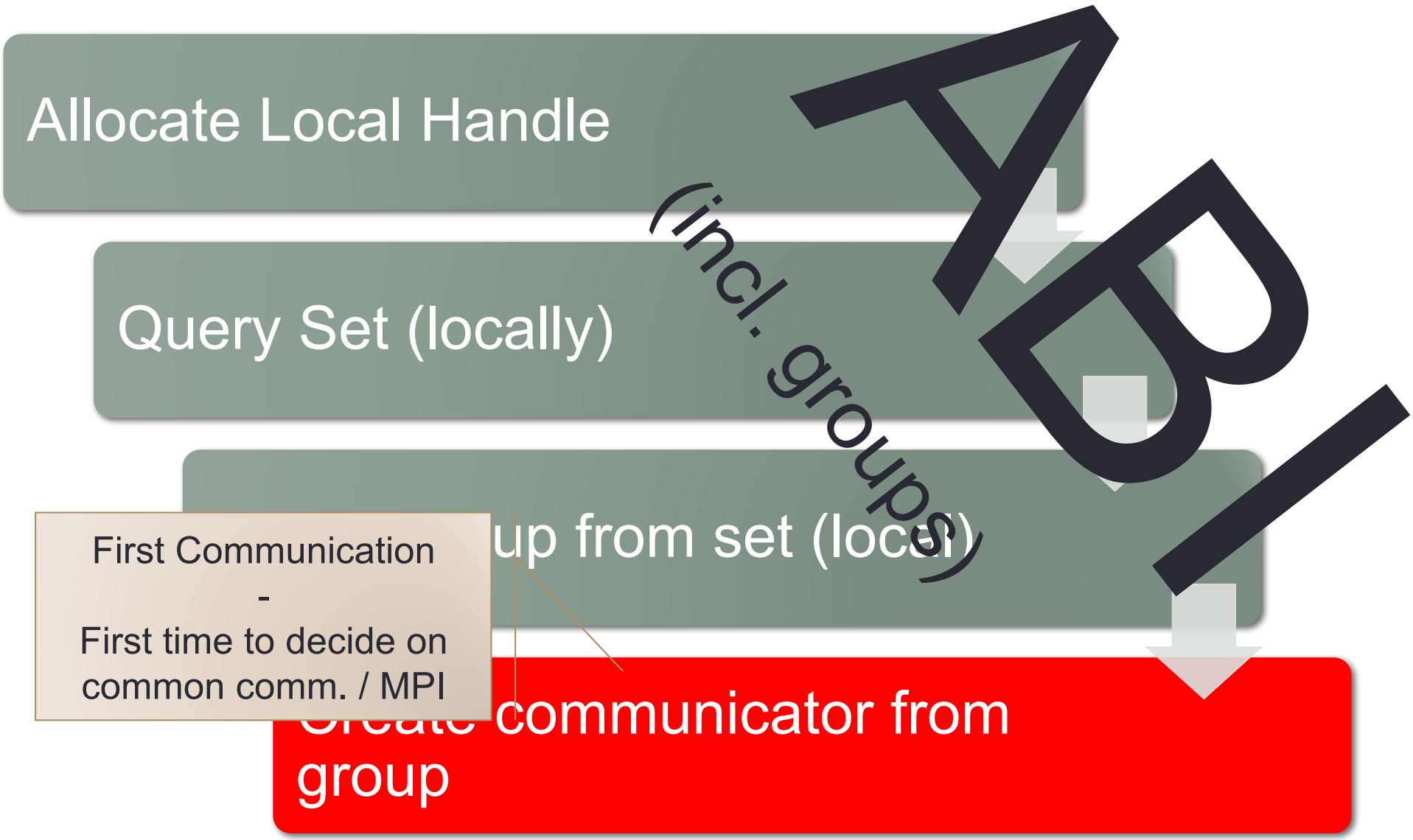
(By allowing multiple sets/comms per session)



Issue: Multiple MPIs

- **Which MPI am I using?**
  - MPI implementation and version
  - Threading options
  - FT approach (in the future)
- **Determined by set that am I using**
  - Different resources require different MPIs
- **What happens when I query the second set?**
  - Are some sets no longer available, since I am bound to an MPI?
  - When am I bound to an MPI?
    - What if I create N groups first and then create N communicators?
  - Does this mean the order matters?
- **Forces ABI for all group operations**

Issue: Late Coordination



Issue Tool Stacks

- **A tool stack should be per ???**
  - Session: that's what it should be, but what is a local tool?
    - Also we don't know MPI, yet
  - Set: wrong concept
  - Communicator: doesn't match MPI concepts
- **Cannot be initialized before communicator creation**
  - Same late coordination problem
  - How to monitor MPI calls before that?
-

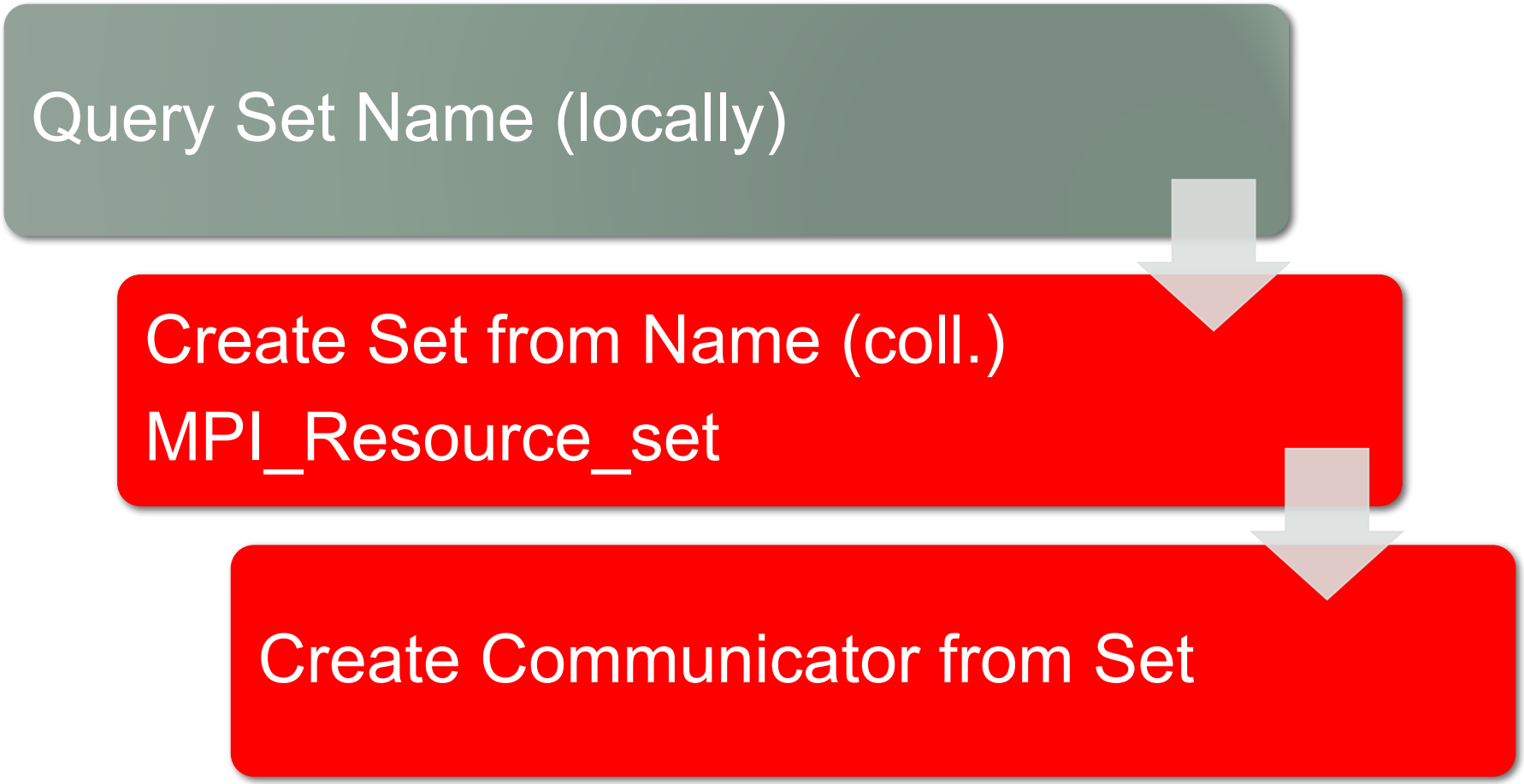
Common Issue

- **Local sessions don’t allow reasoning about what is in a session**
  - Resources across processes
  - What is affected by the session?
    - In case of a fault?
    - Which tool should be applied?
    - Which MPI am I using
- **Local sessions allow multiple sets/communicators**
  - Each can have different flavors
  - Can lead to inconsistencies
  - Further dilutes resource separation

Conceptual Proposal

- **Make Sessions match ideas of MPI_Init/Finalize**
  - That was the original intent anyway
  - One session = One set = One communicator
  - All other resources derived from that
- **Coordinate as early as possible**
  - Don’t keep state outside of sessions
  - ABI only for VERY few calls
- **Enforce consistent sets across nodes**
  - Make set creation a collective operation
- **Oh, and yes, the name should change – it’s taken ☹**

Option 1: Collective Sessions



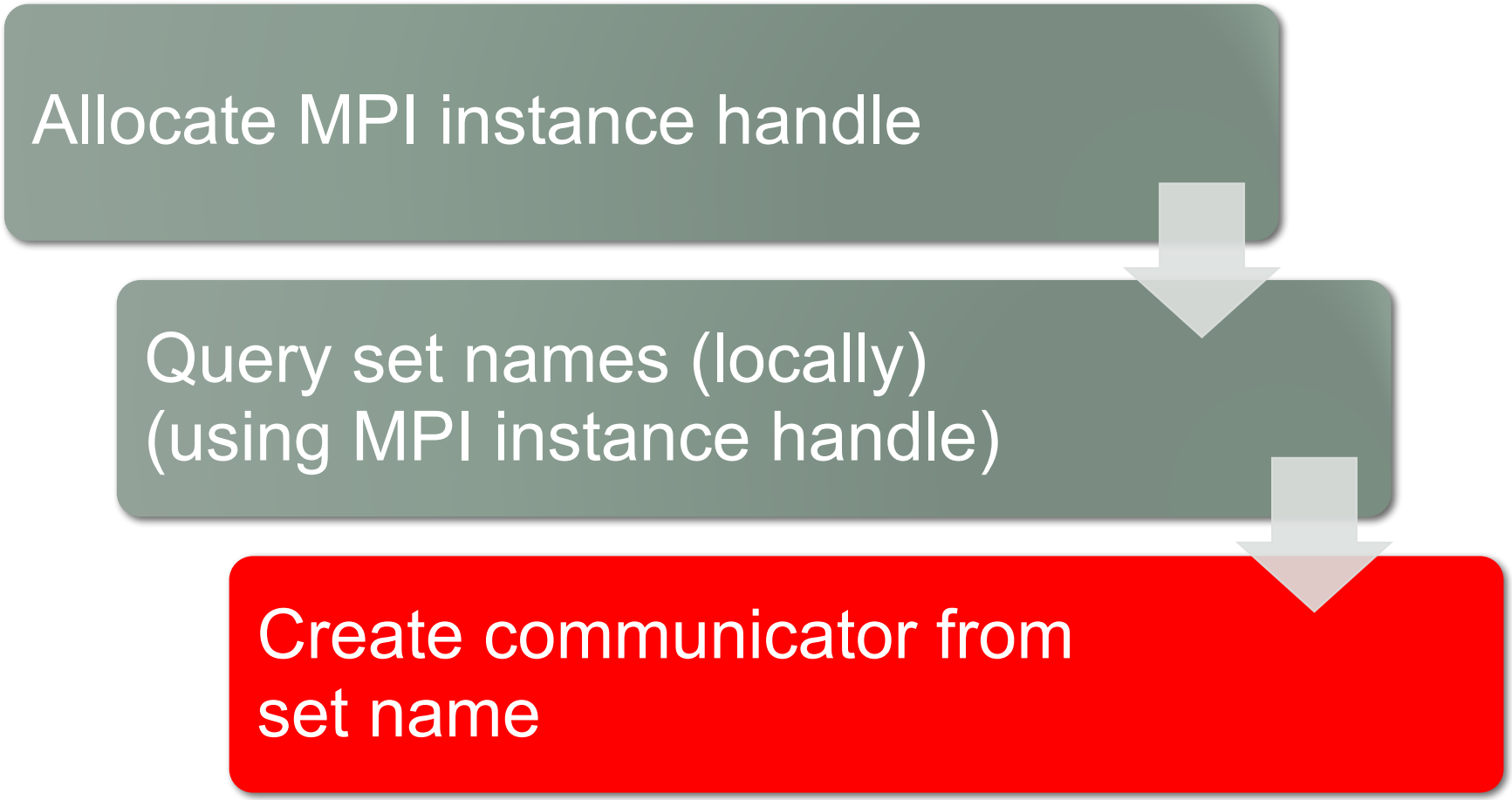
Option 1b: Even Simpler

Query Set Name (locally)

Create Communicator from Set Name (new init)

(new COMM_WORLD for that set)

Option 2: Hybrid Local/Collective



Advantages

- **One Set = One Session = One Communicator**
  - or –
  One Set = One Communicator
  - Clean resource isolation
  - Enables clean FT semantics
  - Assigns a single tool stack
- **Enable early coordination**
  - Minimizes ABI needs
  - Creates consistent sets
- **Side effect: we get rid of the name “Sessions”**