# InterComm Toolkit

Norman Lo and Alan Sussman
University of Maryland

# Motivation

- Simulation of complex systems requires multiple physical models
- Need the capability to move data between coupled models (MxN transfer)
- CCA components allow the reusability of common computational tasks
- With CCA-InterComm toolkit, along with the growing collection of CCA toolkits, complex systems can be developed in a simpler way.

# Creating the InterComm Toolkit

- Bocca provides a simplified way to create and maintain the CCA components
- All the necessary SIDLs, skeleton files, makefiles, and etc are automatically generated.
- Only need to fill in the *.impl* files.
- InterComm and PVM libraries are linked by modifying *make.vars.user*.
- A pointer lookup table is used to pass pointers between components.

# Components

**Schedule**

- Prepares InterComm for schedule building
- Provides XJDPort and uses RegionPort

**ICRegion**

- Calculates the InferComm region information
- Provides RegionPort

**Communicate**

- Send or Receive data
- Provides CommPort

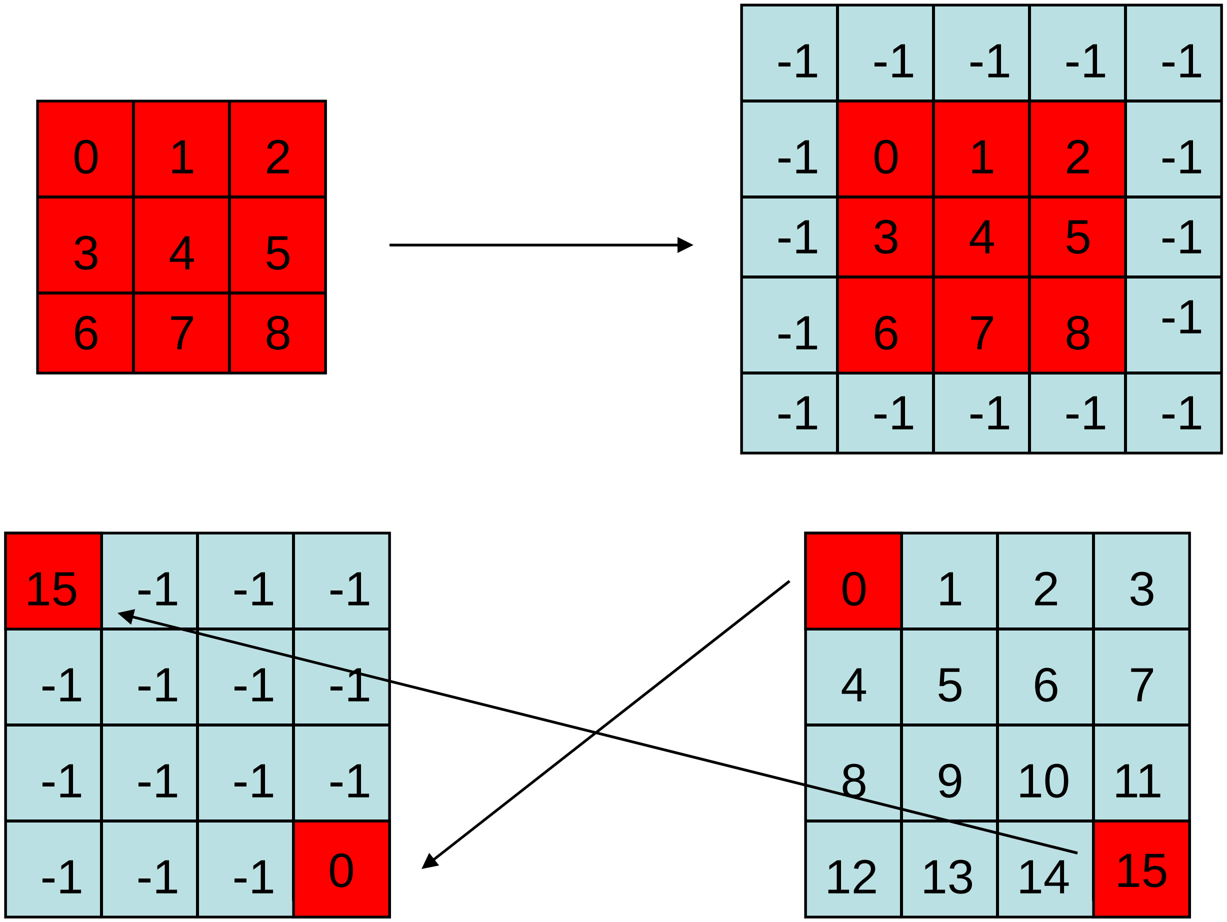
**CDriver/CDriverOther**

- Drives the InterComm components
- Uses XJDPort and CommPort

# Driver Components

**CDriver**

**CDriverOther**



| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

| | | | | |
|---|---|---|---|---|
| -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | 1 | 2 | -1 |
| -1 | 3 | 4 | 5 | -1 |
| -1 | 6 | 7 | 8 | -1 |
| -1 | -1 | -1 | -1 | -1 |

| | | | |
|---|---|---|---|
| 15 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 0 |

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

# Ports

- **XJDPort – returns the XJD_index contains InterComm information**

```
 int computeSchedule(in int progname_index, in int xjd_read,
                     in int rank,                          // Initialize info
                     in int ndims, in int arrayOrder, in int count,
                     in int tasks_index, in int indices_index,
                     in int type,                          // Descriptor info
                     in int rgnname_index, in int rgnset_index,
                     in int rgnsize, in int data_index); // Region info
```

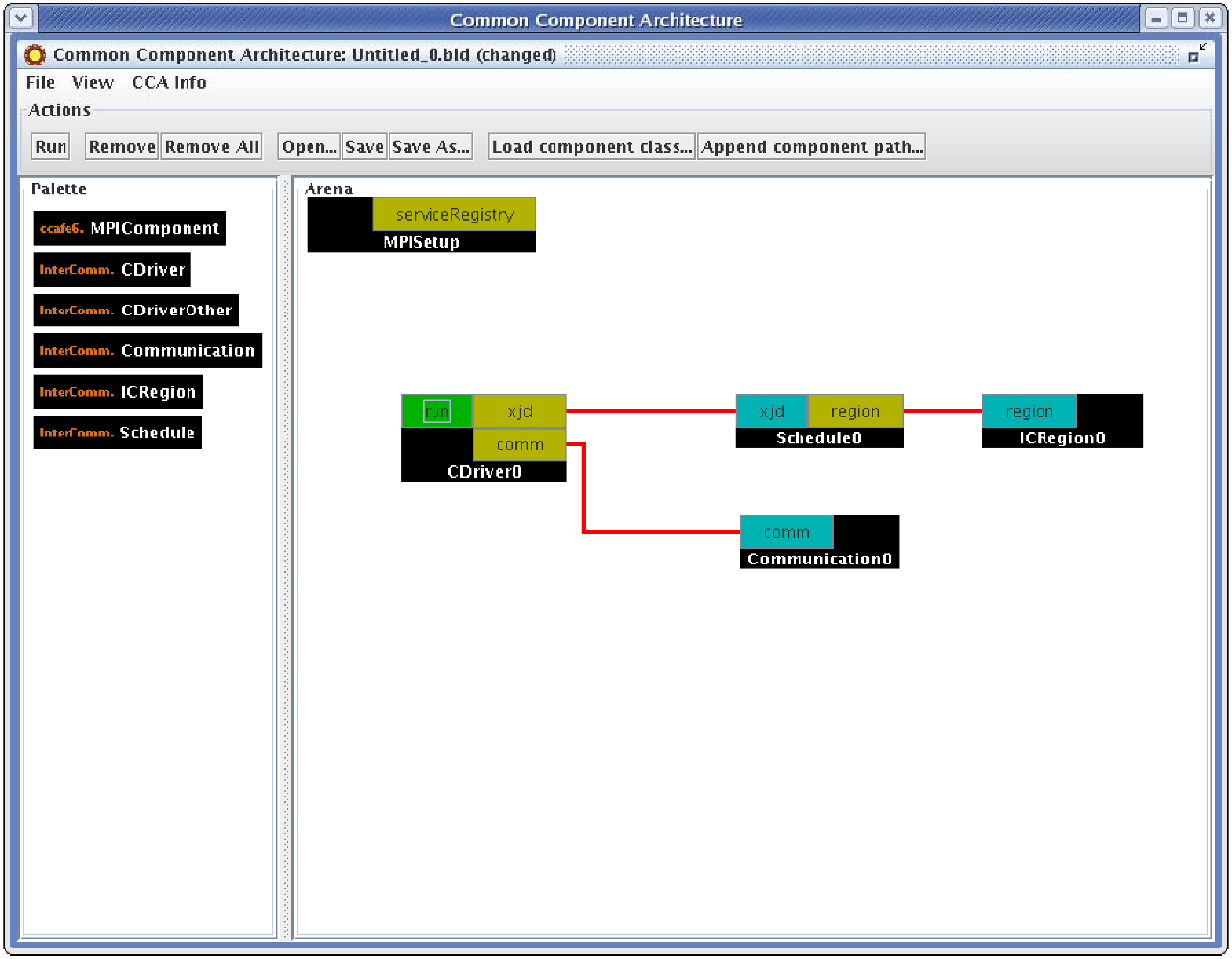
- **RegionPort – returns 0 if success, -1 otherwise**

```
int computeRegion(in int ndims, in int rgnset_index, in int xjd_index,
                  in int rgnsize, in int rgnname_index,
                  in int desc_index, in int data_index);
```

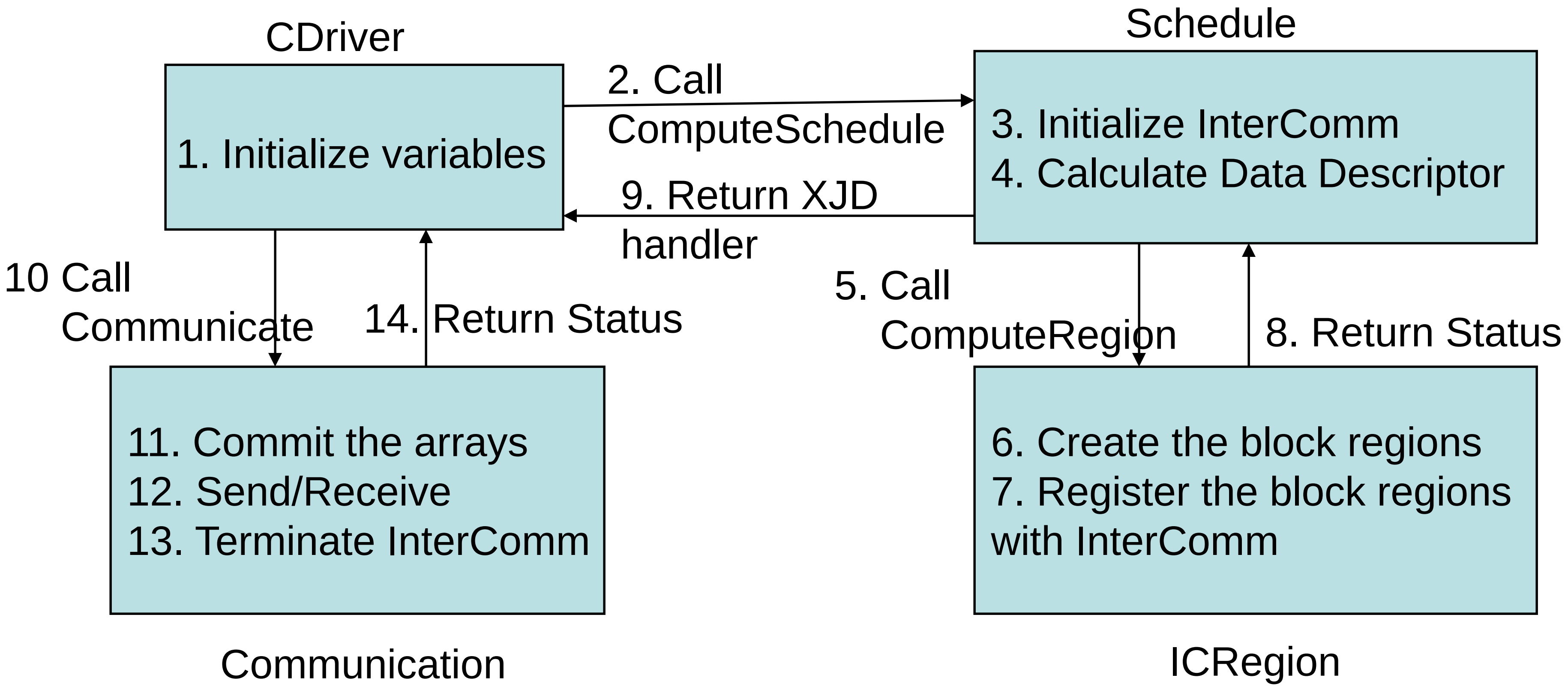
- **CommPort – returns 0 if success, -1 otherwise**

```
int communicate(in int xjd_index, in int rgnname_index, in int
                   direction, in int final_flag);
```

# Connection in GUI



# Flowchart



CDriver

1. Initialize variables

2. Call ComputeSchedule

Schedule

3. Initialize InterComm
4. Calculate Data Descriptor

5. Call ComputeRegion

ICRegion

6. Create the block regions
7. Register the block regions with InterComm

8. Return Status

9. Return XJD handler

10 Call Communicate

Communication

11. Commit the arrays
12. Send/Receive
13. Terminate InterComm

14. Return Status