# Chapter 5

# Communication schedules

The type of the elements must be a POD type[1].

The constructor *does not* create copies of the DAD objects referenced by `dst` and `src`—the pointer values are simply copied. It is the caller's responsibility to ensure that DAD objects passed to schedule constructors persist as long as the constructed schedule may be used.

*But*, schedules never cache pointers to local data segments. The rationale for caching pointers to eg `DAD` objects but never to local data segments is ...

*All* communication schedules should be assumed to read the static variable `apg`, defining the active process group. The active process group in effect when a schedule is executed must be the same as the one in effect when it was created.

A schedule may be executed once or many times. In principle the `execute` member may be passed different base addresses in different calls, so long as the associated arrays are described by DAD structurally identical (shape, mapping *and* strides) to the ones passed to the constructor.

### 5.0.1 Plain 'Ol Data types

*[Define* POD.*]*

[1] *Plain Ol' Data* type, as defined in the draft ANSII C++ standard.

## 5.1 class Remap

A *remap schedule* is a communication schedule for copying the elements of one distributed array to another. The source and destination must have the same shape and same element-type, but no relation between the mapping of the two arrays is required. If the target array has a replicated mapping, the remap operation implements a broadcast.

A remap schedule is described by a collective object with local components of class `Remap`. The public interface of the `Remap` class is

```
class DAD ;

class Remap {
public :
  Remap(const DAD* dst, const DAD* src, const int len) ;

  void execute(void* dstDat, void* srcDat) ;
private :
  ...
} ;
```

### 5.1.1 Constructor

`Remap(const DAD* dst, const DAD* src, const int len)`

The source array is described by the DAD `*src` and the destination array is described by the DAD `*dst`. `len` specifies the size of each array element, in bytes.

**Type restrictions:** The elements of the source and destination arrays must have the same type. This must be a POD type (see section 5.0.1) of size `len` bytes.

**Shape restrictions:** The source and destination array must have the same shape (see section 4.4).

**Alignment restrictions:** None.

**Accessibility restrictions:** The source array must be value-accessible and the destination array must be accessible (see section 4.6).

**Argument persistence:** The `dst` and `src` arguments are stored in the schedule as references. The associated objects must persist for the lifetime of the constructed schedule.