Places.ExchangeBoundary( )

Three different Implementations:

(1) ExchangeBoundary function eb_exchangeBoundary() runs callMethod(exchangeWave_) on Shadowed Boundaries.
It stores the output in the outMessages variable of the shadow boundary.
Then runs callMethod(exchangeWave_) on all other place locations.
All inMessage variables are updated from either the callMethod() output or from the outMessages variable (if a boundary)

Pro: Does not require user to update the outMessage.
Runs user specified callMethod(*func*, *arg*) function on all locations.

Con: Since output of Shadow Boundary callMethod(*func*, *arg*) is stored in outMessages,
we can not use the arg variable since we do not know which arg will be calling at the time Shadow Boundary is called.

(2) User updates the Place.outMessages variable in the computeWave function (in the same fashion that the user reads the inMessages).
The user will still need to create a callMethod( ) function that reads the value from Place.outMessages instead of from Wave2DMASS.wave, since the callMethod( ) function will be called on all place locations (including on the shadow locations).

Pro: This implementation allows for each place to run the user specified callMethod( ) function on each of it's neighbors WITH a unique argument value.

Con: It requires the user to update outMessages on all place locations prior to calling exchangeBoundary( ).

(3) ExchangeBoundary function runs callMethod(exchangeWave_) on each place location (except for shadow bound)
to update the Place.outMessages variable from the Wave2DMASS.wave variable.
Then eb_exchangeBoundary() updates the outMessages of the shadow boundary from the outMessage of the shadowed location.
Then eb_update() updates all inMessage variables from the neighbors outMessages variable.

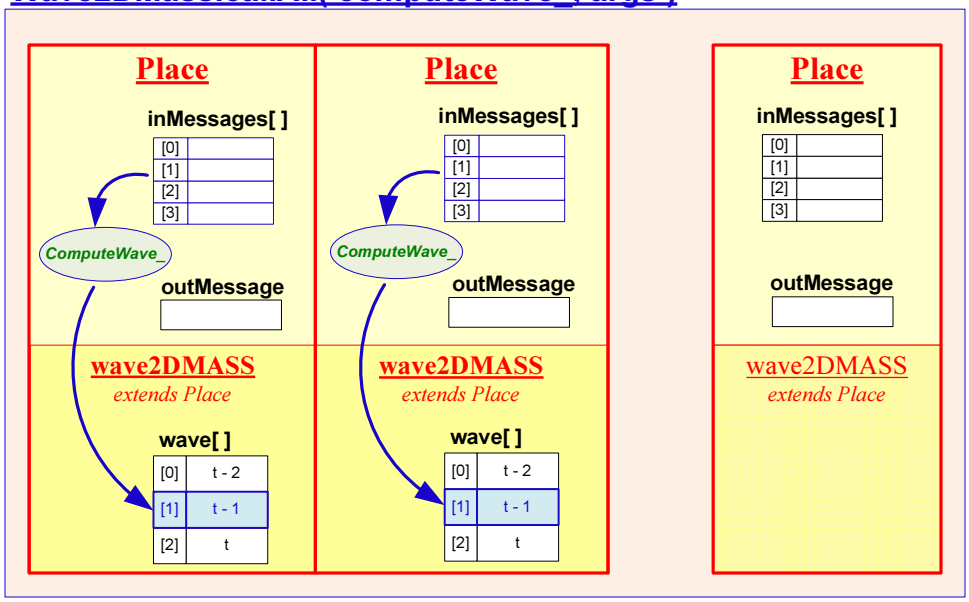
Pro: Does not require user to update the outMessage.
Runs user specified callMethod(*func, arg*) function on all locations.

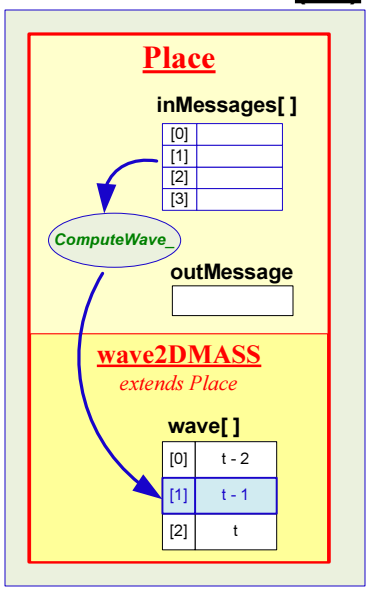
Con: Since each place locations outMessages variable is getting updated from it's own wave variable, there is no way of providing a unique argument value to the callMethod( ) function.

This has been very hard to get to work as it requires exchangeBoundary to call callMethod(exchangeWave, arg) on each location and then take the return value and update the outMessage of that location with that value.

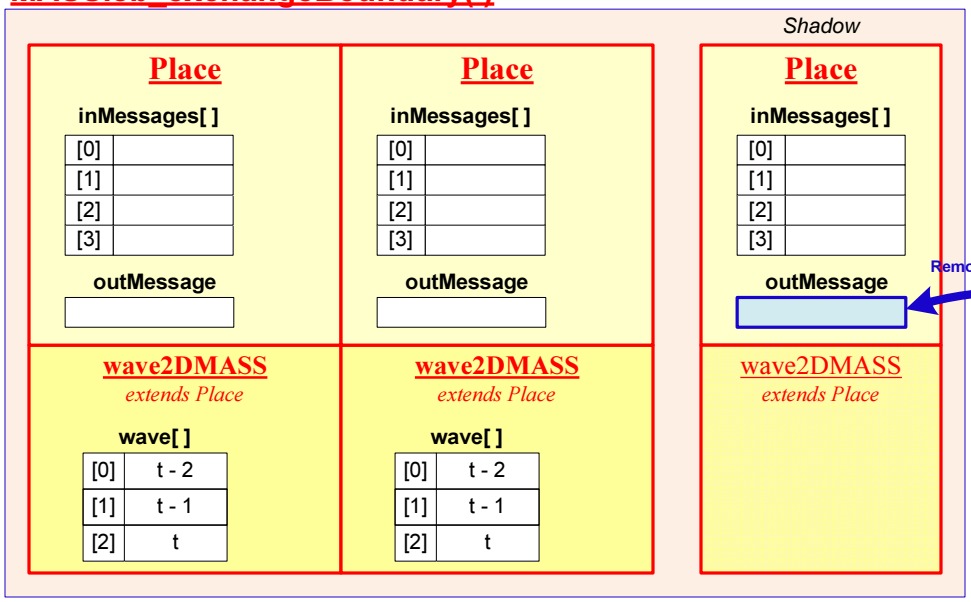
Though I have not run this implementation yet, it seems that with all of the overhead of running callMethod just to transfer the value of Wave2DMASS.wave to Place.outMessage, without being able to use the argument value, is not a very good idea.

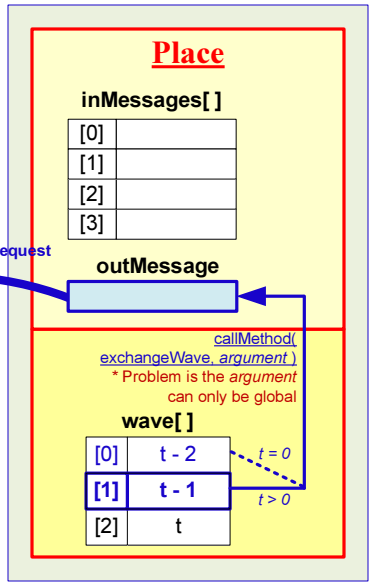
Wave2DMass.callAll(computeWave_, args)



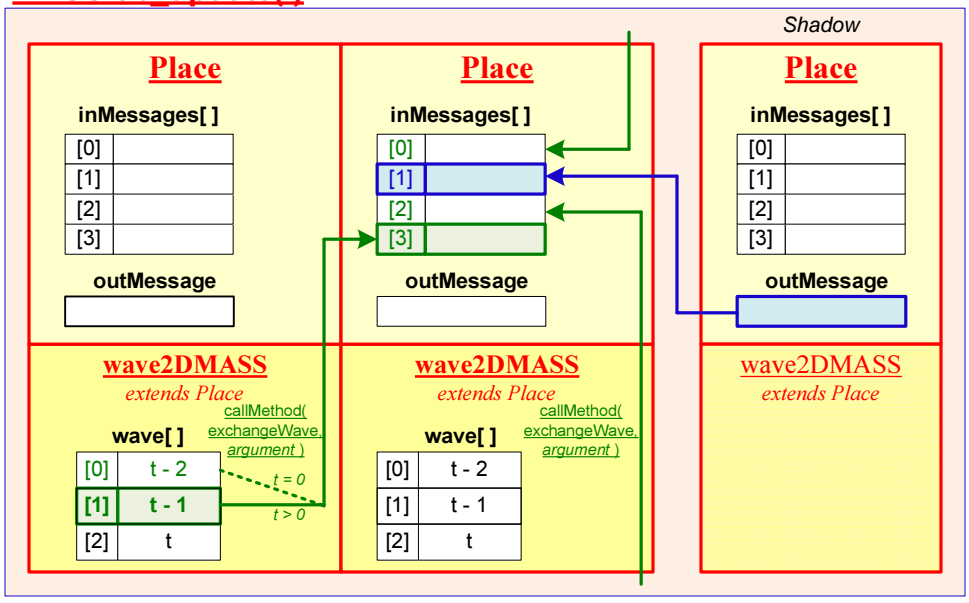


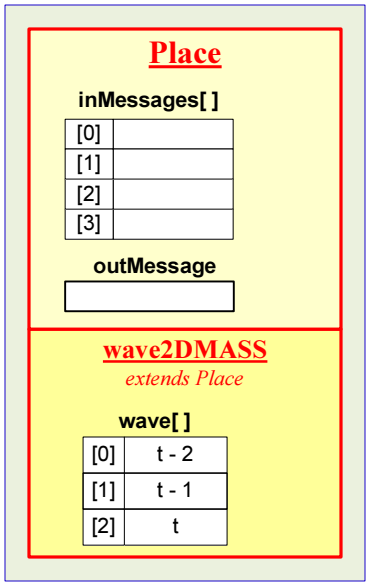
MASS.eb_exchangeBoundary( )



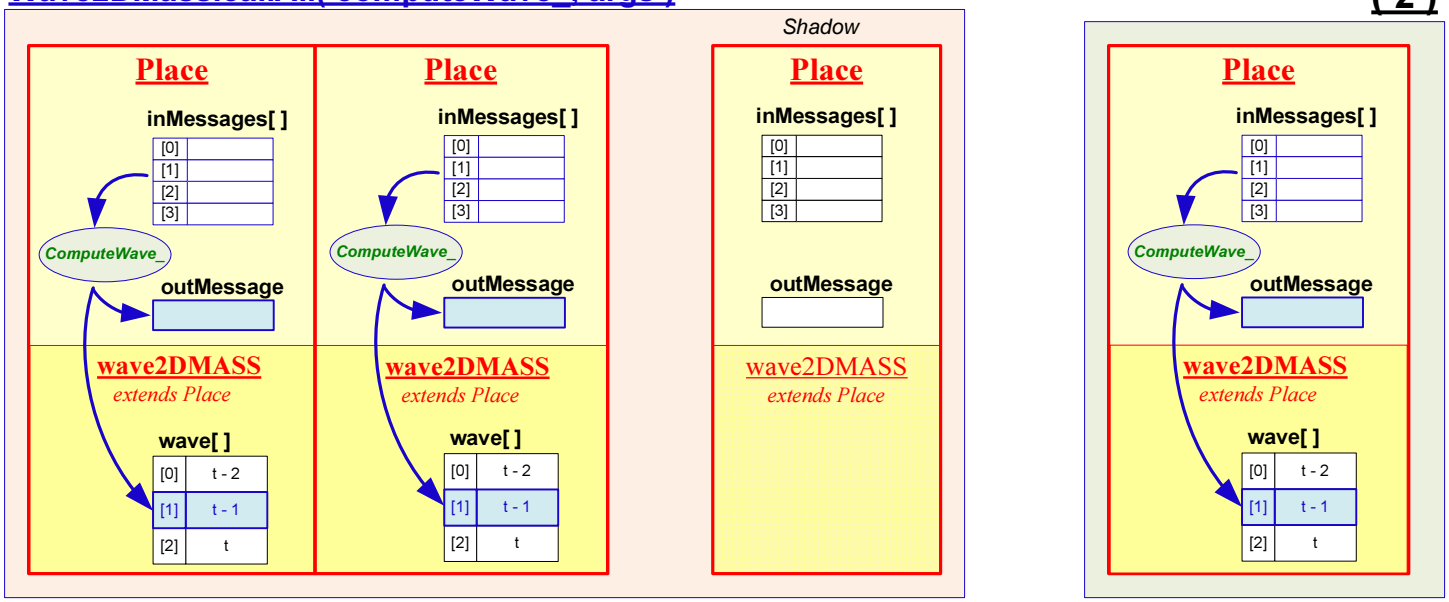


MASS.eb_update( )

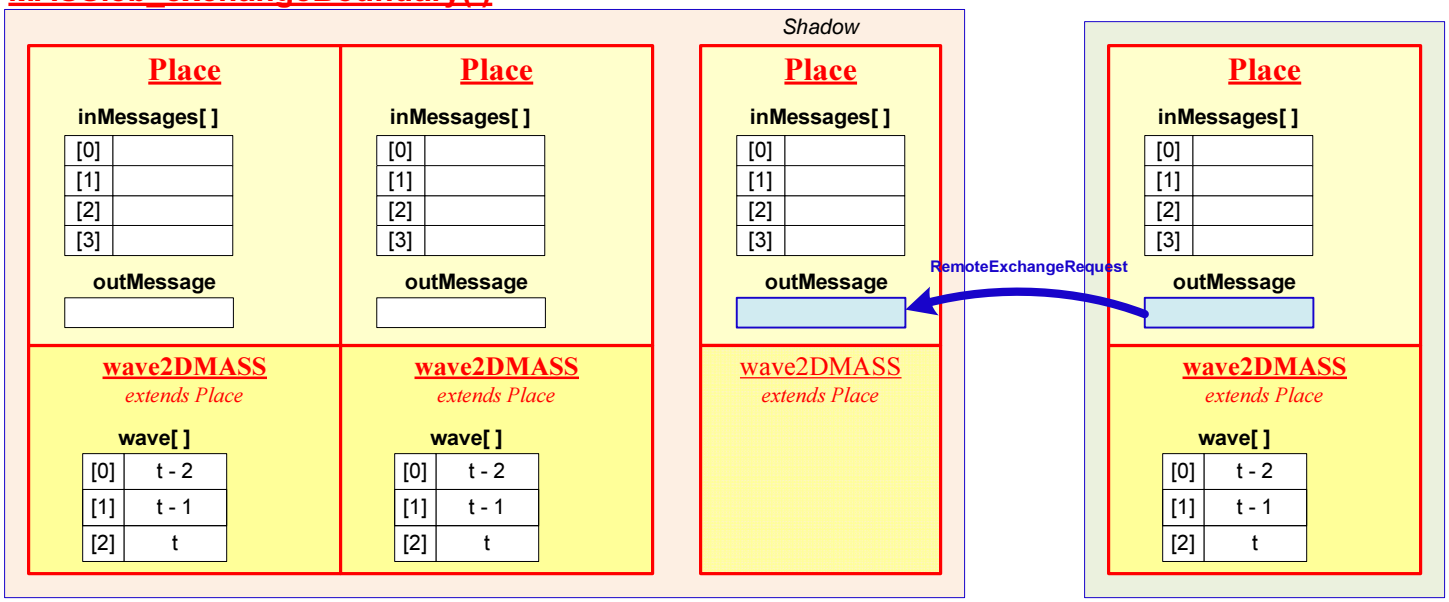




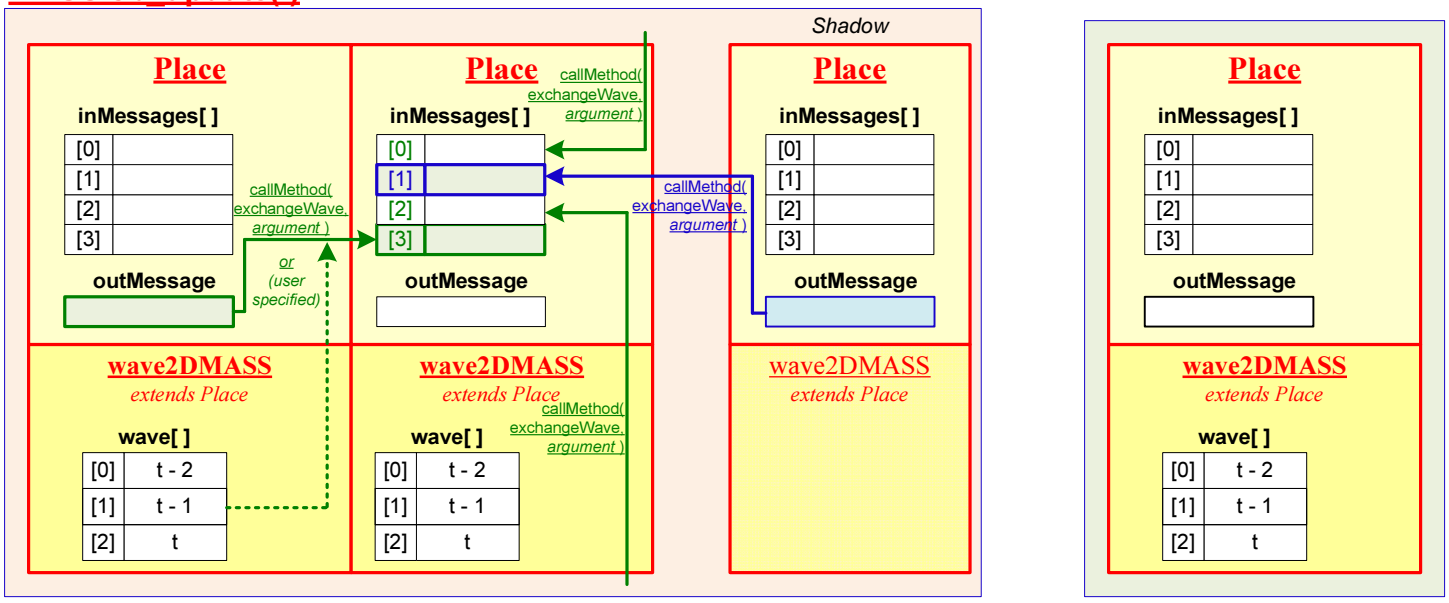
Wave2DMass.callAll(computeWave_, args)



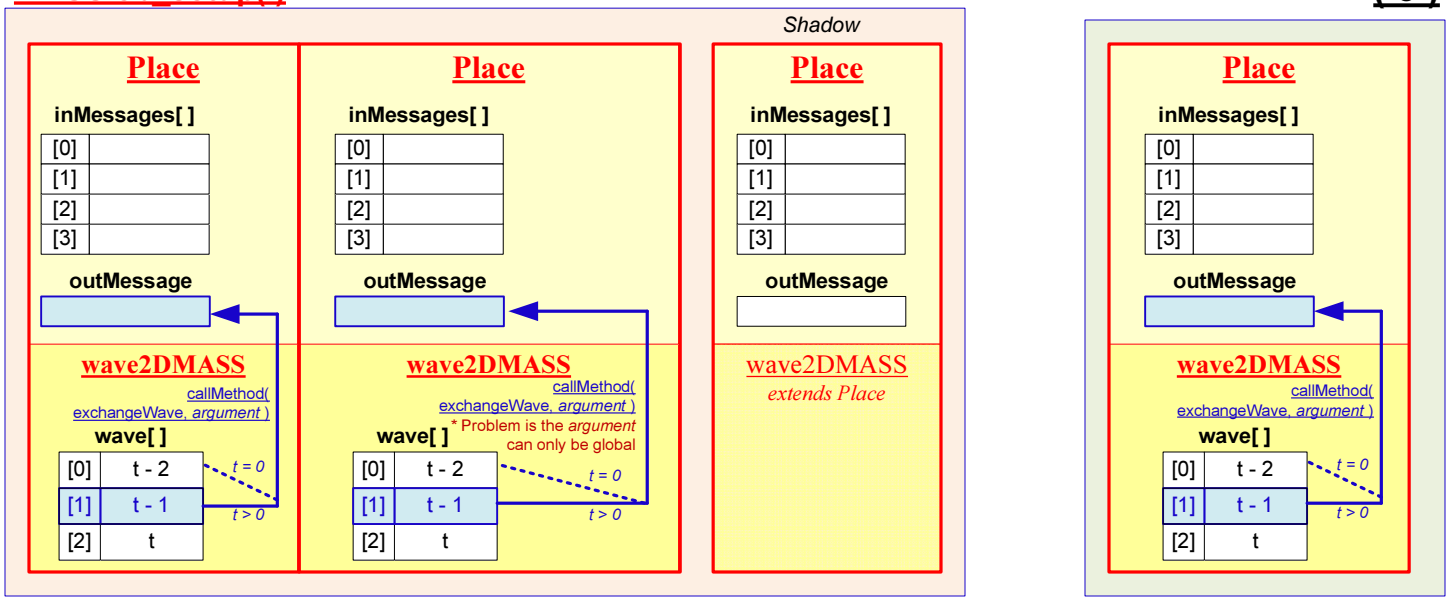
MASS.eb_exchangeBoundary( )



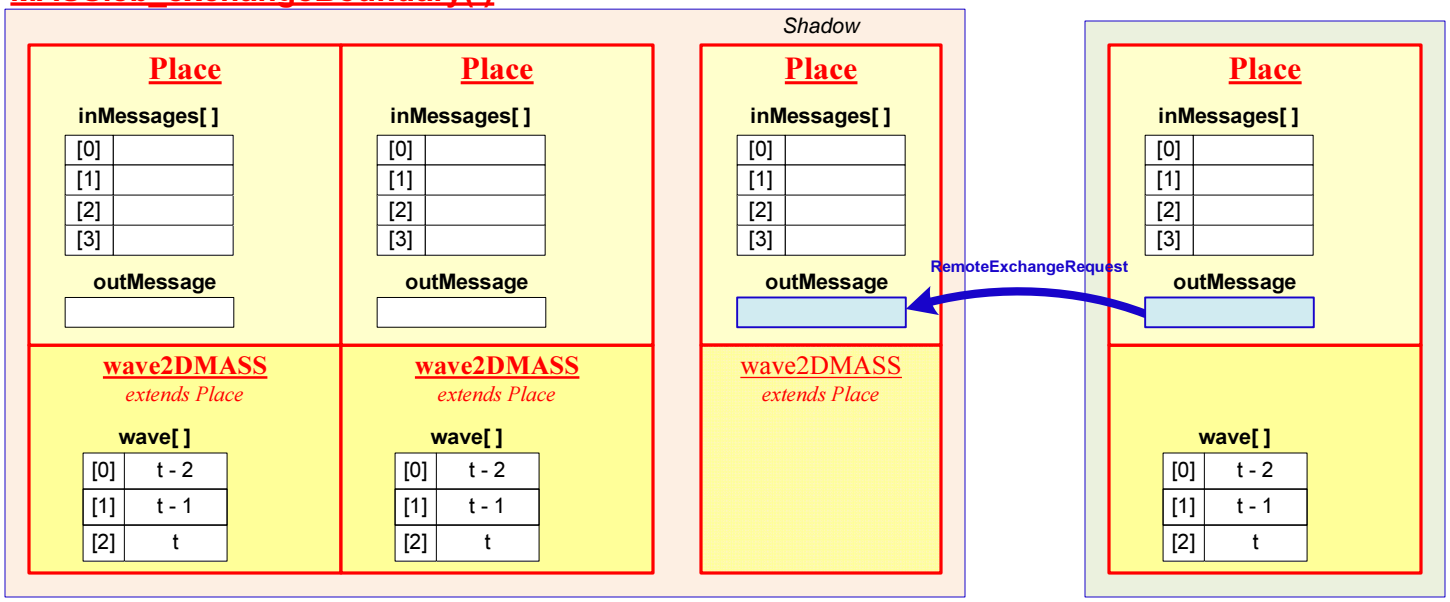
MASS.eb_update( )



MASS.eb_setup( )



MASS.eb_exchangeBoundary( )



MASS.eb_update( )