Supporting Information for

A Dissipative Particle Dynamics Simulation of the Controlled Loading and Responsive Release of Theranostic Agents from Reversible Crosslinked Triblock Copolymer Vesicle

Zhikun Wang[ab‡], Fengting Li[ab‡], Li Wang[ab], Yueqi Liu[ab], Miantuo Li[ab], Nannan Cui[ab], Chunling Li[ab], Shuangqing Sun[ab*], Songqing Hu[ab*]

CONTENTS

[a.] *School of Materials Science and Engineering, China University of Petroleum (East China), Qingdao 266580, China. E-mail: sunshuangqing@upc.edu.cn, songqinghu@upc.edu.cn*
[b.] *Institute of Advanced Materials, China University of Petroleum (East China), Qingdao 266580, China.*
[‡] *The authors contribute equally to this work.*

1. Grouping method for DPD simulations

Table S1 indicates the detailed grouping method of DPD beads in this work. The relative molecular weight per molecule or repeat unit was provided. In DPD simulations, the masses for all grouped beads are consistent. To maximize the consistency in the mass of grouped beads, 8 water molecules were selected to be grouped into one bead W (mass=144 amu), and the number of molecules or repeat units for other beads are defined to be close to this value. In other words, the defined bead types may represent more than the defined group number. For example, a PIBMA DPD bead represents 1.02 real PIBMA segments (144/141=1.02). In the main text of the manuscript, we use integers to describe the equivalent numbers.

Table S1. Grouping method for all DPD beads in the studied system.

	PEG (A)	POPMA (B)	PIBMA (C)	DTP (L)	DOX-HCl (D)	Nilered (N)	Water (W)
Relative molecular weight per molecule or repeat unit / amu	44	142	141	238	580	318	18
Number for one grouped DPD bead	3	1	1	1/2	1/4	1/3	8
Relative mass of grouped DPD beads / amu	132	142	141	119	145	106	144
Equivalent grouped number	1.09	1.01	1.02	1.21	0.99	1.36	1

2. Self-Assembly of Singular Block Copolymer in Solution

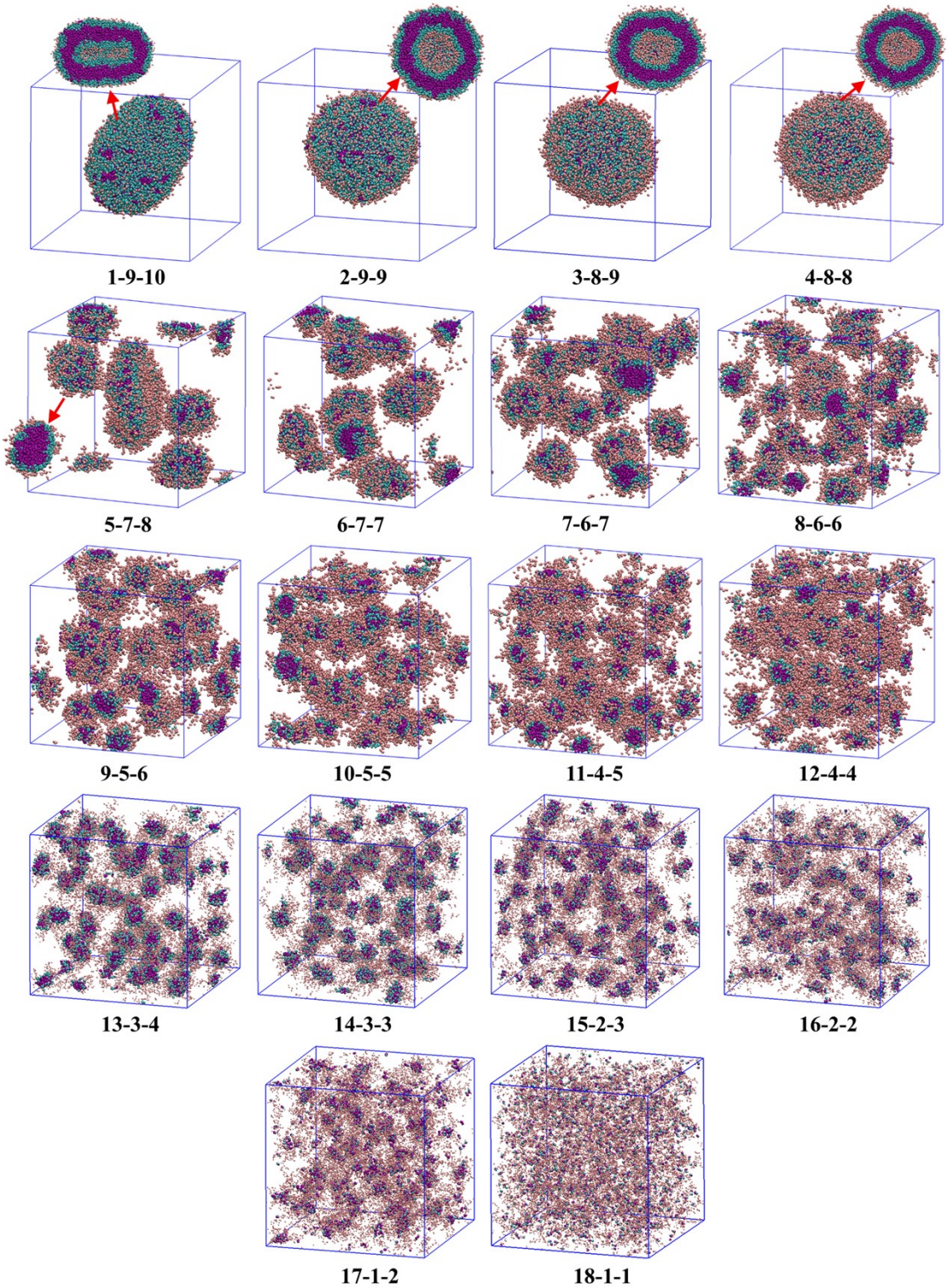

Fig. S1. Self-assembly of block copolymer $A_mB_nC_l$ as a function of the length of hydrophilic block A (a_{AW}=25). The three numbers connected by short dashes indicate the subscripts in $A_mB_nC_l$, respectively (*m* varies from 1 to 18, *m*+*n*+*l*=20, n equals to the integral part of (20-*m*)/2).

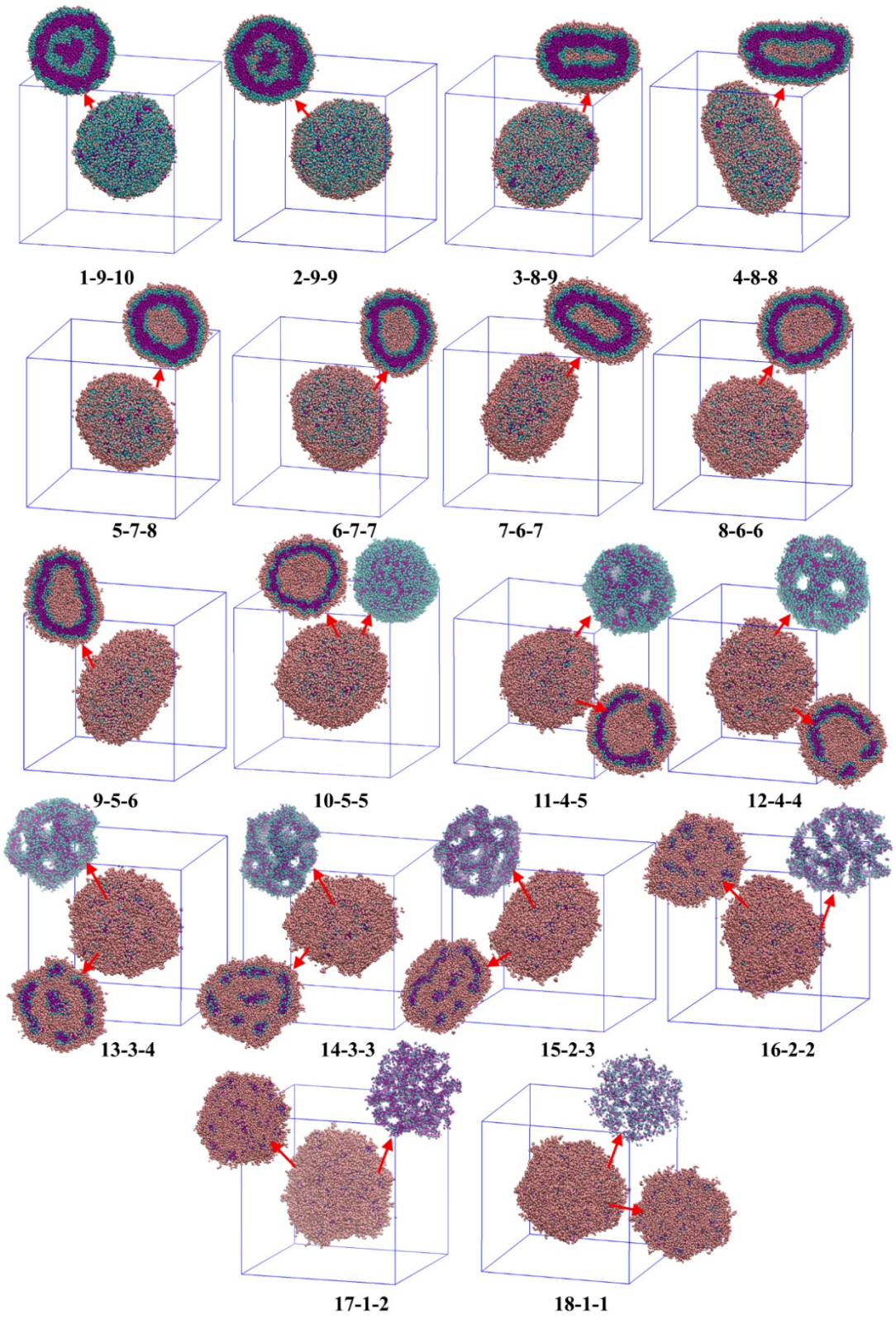

Fig. S2. Self-assembly of block copolymer $A_mB_nC_l$ as a function of the length of hydrophilic block A (a_{AW}=30). The three numbers connected by short dashes indicate the subscripts in $A_mB_nC_l$, respectively (*m* varies from 1 to 18, *m*+*n*+*l*=20, n equals to the integral part of (20-*m*)/2).

3. Crosslinking Perl Script in Materials Studio

#NOTE: To achieve a certain crosslinking degree of the coarse-grained model, this script needs to be performed a few cycles. After each cycle, geometry optimization and dynamical equilibrium are needed.
#LAST DEBUGGING DATE: April 10, 2023
#CONTACT: Zhikun Wang, China University of Petroleum (East China), wangzhikun@upc.edu.cn

```
#!perl

use strict;
use Getopt::Long;
use MaterialsScript qw(:all);

my $DOCUMENT ="80%.xsd";    # name of the model file
my $nameset ="set-MMA";     # name of the defined set
my $cutoff=10;              # cutoff distance used to search adjacent beads
my $namelinker = "B";       # bead type of the linker
my $namesite = "MMA";       # bead type of the crosslink site
my $ThresholdNum = 200;     # limit the number of crosslink sites for one crosslinking loop
my $blockNum = 8;           # number of beads on the crosslinking block

my $doc = $Documents{"$DOCUMENT"};
my $set = $doc->UnitCell->Sets("$nameset");
my $i = 1;
foreach my $bead0(@$set){$bead0->Name = "$i"; $i = $i + 1;}

Tools->BondCalculation->ChangeSettings([DistanceCriterionMode=> "Absolute" ,ExclusionMode
=> "Set", MaxAbsoluteDistance => $cutoff]);
my $closeContacts = $set->Beads->CalculateCloseContacts;

my $stop = 0;
foreach my $closeContact (@$closeContacts) {
    if ($stop < $ThresholdNum){
        my $bead1 = $closeContact->Terminal1;
        my $bead2 = $closeContact->Terminal2;

        my $beadname1 = $bead1->Name;
#       $beadname1 = substr($beadname1,3);
        my $beadname2 = $bead2->Name;
#       $beadname2 = substr($beadname2,3);
        my $judge = $beadname1 - $beadname2;
        if(($judge > $blockNum)and($bead1->BeadTypeName eq $namesite)and($bead2-
>BeadTypeName eq $namesite)) {
```

```
                my $connectedBeads1 = $bead1->AttachedBeads;
                my $n = 0;
                foreach my $bead (@$connectedBeads1) {
                    if ($bead->BeadTypeName eq $namelinker){ $n = $n + 1;}
                }
                my $connectedBeads2 = $bead2->AttachedBeads;
                my $m = 0;
                foreach my $bead (@$connectedBeads2) {
                    if ($bead->BeadTypeName eq $namelinker){ $m = $m + 1;}
                }
                if (($n eq 0)and($m eq 0)){
                    my $newX1 = 0.5*($bead1->XYZ->X + $bead2->XYZ->X) - 0.2*abs($bead2->XYZ->X - $bead1->XYZ->X);
                    my $newY1 = 0.5*($bead1->XYZ->Y + $bead2->XYZ->Y) - 0.2*abs($bead2->XYZ->Y - $bead1->XYZ->Y);
                    my $newZ1 = 0.5*($bead1->XYZ->Z + $bead2->XYZ->Z) - 0.2*abs($bead2->XYZ->Z - $bead1->XYZ->Z);
                    my $newX2 = 0.5*($bead1->XYZ->X + $bead2->XYZ->X) + 0.2*abs($bead2->XYZ->X - $bead1->XYZ->X);
                    my $newY2 = 0.5*($bead1->XYZ->Y + $bead2->XYZ->Y) + 0.2*abs($bead2->XYZ->Y - $bead1->XYZ->Y);
                    my $newZ2 = 0.5*($bead1->XYZ->Z + $bead2->XYZ->Z) + 0.2*abs($bead2->XYZ->Z - $bead1->XYZ->Z);

                    my $newbead1 = $doc->CreateBead("$namelinker", Point(X => $newX1, Y => $newY1, Z => $newZ1));
                    my $newbead2 = $doc->CreateBead("$namelinker", Point(X => $newX2, Y => $newY2, Z => $newZ2));
                    my          $newconnector1=$doc->UnitCell->CreateBeadConnector($bead1, $newbead1);
                    my     $newconnector2=$doc->UnitCell->CreateBeadConnector($newbead1, $newbead2);
                    my          $newconnector3=$doc->UnitCell->CreateBeadConnector($bead2, $newbead2);
                    $newconnector1->Name="NewConnector";
                    $newconnector2->Name="NewConnector";
                    $newconnector3->Name="NewConnector";
                    print "build new connector for this closeconnector","\n";
                    $stop = $stop +1;
}}}}
$closeContacts->Delete;

print "############################################################\n";
my $Numberofcrosslink=0;
```

```
foreach my $con(@{$doc->UnitCell->BeadConnectors}){
	if($con->Name eq "NewConnector"){
		#$doc->UnitCell->CreateDistance([$con->Terminal1, $con->Terminal2]);
		$con->Terminal1->Style = "Ball and stick";
		$con->Terminal2->Style = "Ball and stick";
		$Numberofcrosslink=$Numberofcrosslink+1;
}}
my $total = 0;
foreach my $beadA (@{$set->Beads})    {
	$total = $total +1;
}

$Numberofcrosslink = $Numberofcrosslink*2/3;
my $crosslinkdensity = $Numberofcrosslink / $total;
my $numoflinker = $Numberofcrosslink / 2;

print "total bead number= $total \n";
print "crosslinked bead number= $Numberofcrosslink \n";
print "crosslink density = $crosslinkdensity \n";
print "number of linker molecules = $numoflinker";
```