Electronic Supplementary Material (ESI) for Analyst.

Materials:

We purchased reagent-grade adenosine, sodium phosphate monobasic, sodium phosphate dibasic, and sodium chloride from Sigma-Aldrich (St. Louis, MI); glass slides from Fisher Scientific (Hampton, NH); unlabeled, desalt purified, DNA oligos from Sigma-Aldrich; labeled, HPLC purified DNA oligos from Integrated DNA Technologies (Coralville, IA); and aqueous red Fluoro-Max beads with 1.0 μm diameters from Thermo Scientific (Waltham, MA). We used all reagents as purchased.

The Y-DNA hydrogel architecture we employed (shown in **Figure SI 1**) contains two sets of trivalent, Y-shaped monomers connected on their ends by an adenosine aptamer. Each monomer is in turn composed of of three subunit strands ("a," "b", and "c") annealed to form a Y-shaped, double-stranded core with pendant, single-stranded ends complementary to either the 3' or 5' end of the adenosine aptamer. In this study, we varied the length of the pendant functional ends to hybridize with 6-9 bases on either end of the adenosine aptamer. The sequences we employed are as follows, with sequences comprising the double stranded core italicized and the pendant arms underlined.

3' monomer subunit strands:
3'-6a: 5' *CTTACGGCGAATGACCGAATCAGCCT* ACCTTC
3'-6b: 5' *AGGCTGATTCGGTTCATGCGGATCCA* ACCTTC
3'-6c: 5' *TGGATCCGCATGACATTCGCCGTAAG* ACCTTC

3'-7a: 5' *CTTACGGCGAATGACCGAATCAGCCT* ACCTTCC
3'-7b: 5' *AGGCTGATTCGGTTCATGCGGATCCA* ACCTTCC
3'-7c: 5' *TGGATCCGCATGACATTCGCCGTAAG* ACCTTCC

3'-8a: 5' *CTTACGGCGAATGACCGAATCAGCCT* ACCTTCCT
3'-8b: 5' *AGGCTGATTCGGTTCATGCGGATCCA* ACCTTCCT
3'-8c: 5' *TGGATCCGCATGACATTCGCCGTAAG* ACCTTCCT

3'-9a: 5' *CTTACGGCGAATGACCGAATCAGCCT* ACCTTCCTC
3'-9b: 5' *AGGCTGATTCGGTTCATGCGGATCCA* ACCTTCCTC
3'-9c: 5' *TGGATCCGCATGACATTCGCCGTAAG* ACCTTCCTC

5' monomer subunit strands:
5'-6a: 5' CCAGGT *CTTACGGCGAATGACCGAATCAGCCT*
5'-6b: 5' CCAGGT *AGGCTGATTCGGTTCATGCGGATCCA*
5'-6c: 5' CCAGGT *TGGATCCGCATGACATTCGCCGTAAG*

5'-7a: 5' CCCAGGT *CTTACGGCGAATGACCGAATCAGCCT*
5'-7b: 5' CCCAGGT *AGGCTGATTCGGTTCATGCGGATCCA*
5'-7c: 5' CCCAGGT *TGGATCCGCATGACATTCGCCGTAAG*

5'-8a: 5' CCCCAGGT *CTTACGGCGAATGACCGAATCAGCCT*
5'-8b: 5' CCCCAGGT *AGGCTGATTCGGTTCATGCGGATCCA*
5'-8c: 5' CCCCAGGT *TGGATCCGCATGACATTCGCCGTAAG*

5'-9a: 5' CCCCCAGGT *CTTACGGCGAATGACCGAATCAGCCT*
5'-9b: 5' CCCCCAGGT *AGGCTGATTCGGTTCATGCGGATCCA*
5'-9c: 5' CCCCCAGGT *TGGATCCGCATGACATTCGCCGTAAG*

As a crosslinker, we employed the adenosine aptamer of Huizenga and Szostak [20]:

Unlabeled: 5' ACCTGG*G*GGAGTATTGCGGAGGAAGGT
Labeled: 5' Alexa488-ACCTGGGGGAGTATTGCGGAGGAAGGT-Black Hole Quencher-1

Methods:

We synthesized each monomer by combining a final concentration of 1.0 mM of component strands a, b, and c in 75 mM sodium chloride, 25 mM sodium phosphate, pH 7.0 buffer, then annealing by heating at 95°C for 5 min, then cooling to 4°C at a rate of 1°C/min. We confirmed that the desired products were formed by running products on a 4-20% gradient acrylamide Tris/Borate/EDTA gel (**Figure SI 2**).

We synthesized hydrogels in a specifically designed imaging flow cell composed of a 1.8 mm diameter, 0.75 mm deep cylindrical well drilled into a 75 x 25 x 1.0 mm glass slide overlaid by a polydimethylsiloxane (PDMS) channel consisting of a square hole placed over the cylindrical well flanked by two 0.8 mm by 20 mm channels (see [15] for schematic). To form the hydrogel we mixed final concentrations of 0.18 mM of each Y-monomer with 0.48 mM unlabeled aptamer, 480 µM labeled aptamer, and 0.002% by volume red fluoromax beads in 60 mM sodium chloride, 20 mM sodium phosphate, pH 7.0 buffer. After mixing we covered the well but not the ends of the channel by placing a glass coverslip on top of the PDMS layer. After ~15 minutes of equilibration, we added 20 µl of 10 mM adenosine solution to the top of the gel by pipetting it from one side of the channel. We equilibrated and imaged in an enclosed hutch maintained at 20°C with high humidity to minimize evaporation.

We employed an upright Olympus Fluoroview FV1000 MPE laser scanning confocal microscope with a 25x magnification, 1.05 numerical aperture Olympus X Plan N lens to image the gels before, during, and after dissolution. We excited the aptamer fluorophore and fluorescent beads simultaneously with 473 and 559 nm lasers, respectively, and detected each with PMT detectors. We collected frames at a 1.644 s^{-1} rate using raster scanning. Each frame had a 169 µm x 169 µm field of view and 512 x 512 pixel resolution. We measured each permutation at least in triplicate.

We analyzed molecular- and micron- scale dissolution as described in detail in our previous paper [15]. Briefly, to measure molecular-scale dissolution we employed a small amount of "tracer" aptamer labeled on its termini with a fluorophore-quencher pair that produces high fluorescent yield in the crosslinking conformation but not in the adenosine bound (or otherwise non-crosslinking) conformation. In our previous study, we confirmed that the tracer aptamer does not behave differently from the unlabeled aptamer, indicating that its fluorescence is proportional to the extent of the gel's overall crosslinking [15]. We measured tracer aptamer fluorescence at each frame (i.e., timepoint) as the average intensity of each frame in the 473 nm channel, using ImageJ's batch measure function, obtaining fluorescence decay vs. time curves. For permutations in which

fluorescence decreased to near background levels (i.e., for 12-16 complementary bases), we normalized each decay curve to the maximal fluorescence, obtained by manually selecting the highest fluorescence at or shortly after adenosine addition and the estimated baseline fluorescence value, obtained by fitting the last ~50% of the decay curve to a stretched exponential equation

$$fluorescence = background + (max\ signal - background) * [1 - exp(\tau^{-1}) * (time\ since\ aden)]$$

using the Matlab cftool. We obtained the characteristic exponential dissolution constant τ by fitting the portion of the fluorescence vs. time curve after fluorescence had decreased 25% from the maximal value to a simple exponential equation

$$fluorescence = background + (max\ signal - background) * [1 - \exp(\tau^{-1}) * (time\ since\ adeno]$$

using GraphPad plotting software. We manually extracted t_{half} and t_{lag} for each normalized fluorescence trace. We averaged replicates to obtain averages and standard errors. For permutations that did not show substantial molecular-scale dissolution (i.e., 17 and 18 basepairs) we estimated background fluorescence as the average background value of measurements taken that day and obtained τ by fitting and t_{half} and t_{lag} manually, when appropriate.

We measured molecular-scale dissolution by employing two-point passive rheology of entrapped beads, as detailed in our previous paper [15] and in the seminal two-point passive rheology paper [24]. Briefly, we measured the average squared distance (i.e., mean squared displacement, "MSD") that 1.0 μm diameter beads embedded in the gel over moved over one 1.644 s imaging frame, over the course of dissolution. We obtained bead tracks using the particle tracker function on Bitplane Imaris and calculated the square change in distance over one frame for all pairs of beads present in both frames and separated by a distance less than 10x the bead diameter. For plotting and analysis we binned the squared displacements for each trial into 100 consecutive bins. We obtained the characteristic exponential time dissolution constant τ by fitting the portion of these binned MSD vs time curves after the beads had achieved a binned mean square displacement value equal to 10% of that expected in buffer to the simple time delayed exponential equations

$$MSD = IF[(t > t_{lag}): MSD_{plateau} * [1 - \exp(\tau^{-1}) * ((time\ since\ adenosine\ addition) - t_{lag})]$$

in GraphPad. We manually extracted t_{half} (the time at which the MSD is equal to half that expected in buffer) and t_{lag} (the time at which the MSD is equal to 10% that expected in buffer) from the binned data.

Matlab scripts:

1. Function for plotting raw fluorescence.

```
%Raw fluorescence data plotting function
%Plaxco Group

%This function will pull the fluoresence data from the CSV file generated
%by ImageJ. It will then normalize to a percent max format, and feed the
%final data matrix back into the program that called this function as the
%variable "fluoro_data", in which the first column is the time stamp and
%the second column is the normalized fluoresence data.



function fluoro_data = FNC_Fluoro_Plot_041415_rawdata_forpaper(filename,
maxfluor)
    %"filename" is the name CSV file obtained by imagej batch measure of the
images
    %"maxfluor" is the frame number which contains what we hold to be the
    %maximum fluorescence, which is the fluorescence obtained immediately
    %after the addition of adenosine

    %Read the data file "filename"
    RawdataALL = csvread(filename,1);

    %Extract the first and third columns, which contain the frame number and
the fluorescence.
    Rawdata = [RawdataALL(:,1) RawdataALL(:,3)];

    %Make a matrix with just the fluorescence values
    Fluor_vals=Rawdata(:,2);

    %Obtain the maximum fluorescence, which occurs at the manually-input
    %time "maxfluor"
    peak_fluor = Fluor_vals(maxfluor);

    %Obtain all of the fluorescence values as fractions of the maximal
    %fluorescence by dividing by the peak fluorescence
    NormF = (Fluor_vals/peak_fluor);


    %Make a matrix with just the frame values
    frame = Rawdata(:,1);


    %Compile the normalized data into a matrix for (optional) plotting
    NormalData=[frame(maxfluor:length(frame)) NormF(maxfluor:length(frame))];

    % Remove discontinuities due to moving the focus, etc for easier fitting
    % later:
    % Start by defining an empty matrix to do this
    no_disc_data = [];

    % for the whole data set:
    for position = 3:length(NormalData(:,1));
        %get the fluorescence for that data point
        num2=NormalData(position,2);
```

```
        %get the fluorescence for one point before that data point
        num1=NormalData(position-1,2);
        %get the fluorescence for two points before that data point
        num0=NormalData(position-2,2);
        %if the data point is not < 65% of the one before it
        if num2/num1 > 0.65;
            %if the data point that one is not < 65% of the one before it
            if num1/num0 > 0.65;
                %add to the data set without discontinuities
                no_disc_data = [no_disc_data; NormalData(position,1)
NormalData(position,2)];
            end
        end
    end


    % Get a data set only including the time after the addition of
    % adenosine for curve fitting
    % First, get a no-discontinuity data from the time that
    % fluorescence is added (assumed to be maxfluor) to ten points before
    % the end of the set
    FittingDataSet =
no_disc_data((round(maxfluor):round(length(no_disc_data)-10)),:);

    %Subtract the frame where fluorescence is added from all of the times
    %to get the frame number after the addition of fluorescence
    frame = FittingDataSet(:,1)-maxfluor;
    %Get the fluorescent values with the discontinuities removed
    fluoro =FittingDataSet(:,2);
    %The final output is the zero corrected time and the normalized
    %fluorescence values, with the discontinuities removed
    fluoro_data = [frame fluoro];

    % optional plot toggle on/off
    % plot(fluoro_data(:,1),fluoro_data(:,2), color);
```

2. Function for normalized raw fluorescence.

```
%Background subtracted plotting function
%Plaxco group

%This function will pull the fluoresence data from the CSV file generated
%by ImageJ. It will then normalize the fluorescence to the percent maximum
fluorescence (the
%fluoresence at the chosen max time) and the background levels previously
%obtained by fitting the previous data curves to find their background. It
%will also correct the time so that the frame in which the adenosine is
%added is equal to time zero, even if the maximal fluorescence occurs
%later.


function fluoro_data = FNC_Fluoro_Plot_060415_baselinesubtr(filename,
maxfluor,baseline, realstarttime);
```

```
    %"filename" is the name CSV file obtained by imagej batch measure of the
images
    %"maxfluor" is the frame number which contains what we hold to be the
    %maximum fluorescence, which is the fluorescence obtained immediately
    %after the addition of adenosine
    %"baseline" is the baseline calculated by fitting the
    %non-baseline-subtracted raw data to an exponential function to find
    %the baseline
    %realstarttime is the time at which we add the adenosine, which is
    %generally a few frames before the time of maximal fluorescence due to
    %the gel swelling and settling before then.

    %Read the data file "filename"
    RawdataALL = csvread(filename,1);

    %Extract the first and third columns, which contain the frame number and
the fluorescence.
    Rawdata = [RawdataALL(:,1) RawdataALL(:,3)];

    %Make a matrix with just the fluorescence values
    Fluor_vals=Rawdata(:,2);

    %Obtain the maximum fluorescence, which occurs at the manually-input
    %time "maxfluor"
    peak_fluor = Fluor_vals(maxfluor);

    %Obtain all of the fluorescence values as fractions of the maximal
    %fluorescence by dividing by the peak fluorescence
    NormF = (Fluor_vals/peak_fluor);

    %Correct to set the baseline at zero
    NormF_baselinesub = (NormF-baseline)/(1-baseline);


    %Make a matrix with just the frame values
    just_frame_numbers = Rawdata(:,1);


    %Compile the normalized data into a matrix for (optional) plotting
    NormalData=[just_frame_numbers(maxfluor:length(just_frame_numbers))
NormF_baselinesub(maxfluor:length(just_frame_numbers))];


    %Compile the normalized data into a matrix for (optional) plotting
    NormalData=[just_frame_numbers(maxfluor:length(just_frame_numbers))
NormF(maxfluor:length(just_frame_numbers))];

    % Remove discontinuities due to moving the focus, etc for easier fitting
    % later:
    % Start by defining an empty matrix to do this
    no_disc_data = [];

    % for the whole data set:
    for position = 3:length(NormalData(:,1));
```

```
        %get the fluorescence for that data point
        num2=NormalData(position,2);
        %get the fluorescence for one point before that data point
        num1=NormalData(position-1,2);
        %get the fluorescence for two points before that data point
        num0=NormalData(position-2,2);
        %if the data point is not < 65% of the one before it
        if num2/num1 > 0.65;
            %if the data point that one is not < 65% of the one before it
            if num1/num0 > 0.65;
                %add to the data set without discontinuities
                no_disc_data = [no_disc_data; NormalData(position,1)
NormalData(position,2)];
            end
        end
    end


    % Get a data set only including the time after the addition of
    % adenosine for curve fitting
    % First, get a no-discontinuity data from the time that
    % fluorescence is added (assumed to be maxfluor) to ten points before
    % the end of the set
    FittingDataSet =
no_disc_data((round(maxfluor):round(length(no_disc_data)-10)),:);

    %Subtract the frame where fluorescence is added from all of the times
    %to get the frame number after the addition of fluorescence
    just_frame_numbers = FittingDataSet(:,1)-maxfluor;
    %Get the fluorescent values with the discontinuities removed
    fluoro =FittingDataSet(:,2);
    %The final output is the zero corrected time and the normalized
    %fluorescence values, with the discontinuities removed
    fluoro_data = [just_frame_numbers fluoro];

    % optional plot toggle on/off
    % plot(fluoro_data(:,1),fluoro_data(:,2), color);
```

3. Function for plotting bead mean square displacements.

```
%Bead plotting function
%Plaxco group

%This function will pull data from the particle tracking CSV file generated
%by the Imaris software. The function can then calculate the average squared
change
%in particle seperation between all particles as a function of time.
%The output is presented in the variable "MSD_InterBead_Distance", which
%contains the timestamp in the first column and the average squared change
%in interbead distance in the second column.
```

```
function MSD_InterBead_Distance =
Luke_msd_interbead_addstart061015_forpaper(filename, limit, pointadded,
bin_on_off, color, distancecutoff, maxdisplacement)
    %"filename" is the name of the file obtained from Imaris containing the
    %positions, times, and IDs for the beads. The format is [x position, y
    %position, z position (not tracked so arbitrary), time, bead id, and
    %point id
    %"limit"
    %"pointadded" is the frame at which we added the adenosine
    %"color" is the color for (optional) plotting
    %"distancecutoff" is the distance cutoff below which we do not consider
    %the change in distance between bead pairs. Generally we input ~10x
    %bead diameter (see [ref two point microscopy paper,
    %Crocker/Valentine/Weitz])
    %"maxdisplacement" is the maximal one-frame displacement of the bead
    %that we would assume to be real, i.e., not a tracking error

    %import data file
    datafile = csvread(filename,1);

    %get the total number of points
    numberpts = length(datafile);

    %Imaris sets the bead index to a nine digit number, in the format
    %"10000001, 10000002, 1e9 + n" for the 1st,
    % 2nd, nth beads. Set this to just 1, 2, n format for easier viewing
    for n = 1:numberpts;
        datafile(n,5) = datafile(n,5)-1e9;
    end

    %Sort the data by the bead I.D.
    datasorted = sortrows(datafile, 5);


    %Caculate the displacement of each bead over one frame intervals and
store those displacements in a list
    %and store those displacements in a list.
    %Make an empty list for displacements
    xdisplacements = [];
    %For each row in the matrix (i.e., for each datapoint)
    for line = 2:numberpts;
        % if the bead ID of that point is the same as that of the previous
point
        if datasorted(line,5) == datasorted(line-1,5);
            %get the time of that point
            timeindex = datasorted(line, 4);
            %if the time is after the time when we added the adenosine
            if timeindex >= pointadded;
                %obtain the x and y positions of that bead at the given
                %point
                xposition = datasorted(line,1); yposition =
datasorted(line,2);
                %calculate the displacement, from the frame before the
                %current one to the current one
                xdisp = datasorted(line,1)-datasorted(line-1,1);
                %get the bead number
```

```
                bead_number = datasorted(line, 5);
                % if the displacement of the bead is less than the upper
                % limit maxdisplacement:
                if abs(xdisp) < maxdisplacement;
                    %append the bead's timeindex, x displacement, ID, x
                    %position, and y position to the list
                    xdisplacements = [xdisplacements; timeindex xdisp
bead_number xposition yposition];
                end
            end
        end
    end


    %Sort the xdisplacements by time
    sort_by_time = sortrows(xdisplacements, 1);

    %Find where the frame number changes in order to find the row numbers
    %in xdisplacements that correspond to each frame.
    %Make an empty list for these points
    cut_points = [];
    %for each row but the first in the list of displacements sorted by time
    for row = 2:length(sort_by_time);
        %if the time index of a data point is NOT equal to the time index
        %of the next in the sort_by_time matrix, that means that the bead
        %ID is different, so that it is the start of a new series of beads.
        %If this is the case, then add this index to the list of
        %cut_points.
        if sort_by_time(row, 1) ~= sort_by_time(row-1,1);
            point = row;
            cut_points = [cut_points; point];
        end
    end

    %Extract the single frame delta displacements from the xdisplacements
list,
    %using the cut_points values as row cut-offs.
    % Make an empty matrix to do this
    Single_Frame_Delta_Disps = [];
    %For each line between 3 and the number of cut points
    for line = 3:length(cut_points);
        %the end of displacements for those time intervals
        end_time = cut_points(line);
        %the start of the set of that time intervals
        start_time = cut_points(line-1);
        %the displacements at those time intervals
        data_thatcutpoint = sort_by_time(start_time:end_time-1,:);
        %if the frame has more than one bead in it, then continue (if not,
        %skip)
        if length(data_thatcutpoint(:,1)) ~= 1;
            %Find the difference in displacement for each combination of
            %beads, which corresponds to the change in the inter-bead
            %distance over one frame for that pair.
            %First, set a temporary matrix to do this
            temp_inter_dist_storage =[];
            %for "bead one" present at the time points
```

```
            for bead_one = 1:length(data_thatcutpoint(:,1));
                %for each "bead two" also present at that time point and
further down on the list
                for bead_two = bead_one+1:length(data_thatcutpoint(:,1));
                    %get the x position of both beads and the distance
                    %between them
                    xpos_one = data_thatcutpoint(bead_one,4); xpos_two =
data_thatcutpoint(bead_two,4); xdistance = xpos_one - xpos_two;
                    %get the y position of both beads and the distance
                    %between them
                    ypos_one = data_thatcutpoint(bead_one,5); ypos_two =
data_thatcutpoint(bead_two,5); ydistance = ypos_one - ypos_two;
                    %get the horizontal distance between the beads
                    %(pythagorean theorum)
                    distancebtbeads = sqrt(xdistance.^2 + ydistance.^2);
                    %if the distance is greater than required by the cutoff
                    if distancebtbeads > distancecutoff;
                        inter_dist = data_thatcutpoint(bead_two, 2)-
data_thatcutpoint(bead_one,2);
                        %get the bead IDs
                        bead_one_id = data_thatcutpoint(bead_one,3);
                        bead_two_id = data_thatcutpoint(bead_two,3);
                        %Square the distance and place it in a storage matrix
with the bead IDs.
                        temp_inter_dist_storage = [temp_inter_dist_storage;
inter_dist^2 bead_one_id bead_two_id];
                    end
                end
            end
            %Average the change in displacement for all the combinations of
            %beads in that frame.
            ave_inter_dist_change = mean(temp_inter_dist_storage(:,1));
            if ave_inter_dist_change < 10; %to prevent artifacts
                Single_Frame_Delta_Disps = [Single_Frame_Delta_Disps;
data_thatcutpoint(1,1) ave_inter_dist_change];
            end
        end
    end

    %If binning is toggled "on," bin the data over 100 bins to make it
    %easier to interpret
    if bin_on_off == 1;
        %set the number of bins to 100
        num_bins = 100;
        %find the bin step size by dividing the number of frames by 100
        bin_step_size = int64(length(Single_Frame_Delta_Disps)/num_bins)-1;
    end

    %If binning is toggled "off," the bin step size is set as zero
    if bin_on_off == 0;
        bin_step_size = 1;
        limit = limit/1;
    end

    %set an empty matrix for the binned data
    binned_data_mat = [];
```

```
    %step through the data looking at each window, the size of which is
    %defined by bin_step_size
    for row = bin_step_size:bin_step_size:length(Single_Frame_Delta_Disps);
        %put the data for each bin window in a temporary matrix for
        %averaging
        temp_bin_matrix = [];
        %for each
        for group = row-bin_step_size+1:row;
            temp_bin_matrix = [temp_bin_matrix;
Single_Frame_Delta_Disps(group,1) Single_Frame_Delta_Disps(group,2)];
        end
        %average the data in the bin window, then store the result in the
        %main data matrix
        binned_data_mat = [binned_data_mat; mean(temp_bin_matrix(:,1) -
pointadded) mean(temp_bin_matrix(:,2))];
        %setting this number (below) will change the max length of the data
        %set.  If you get a "horzcat", lower this number to your smallest
        %matrix dimension
        if length(binned_data_mat) >= round(limit);
            break
        end
    end
    %plot the binned results (optional)
    plot(binned_data_mat(:,1), binned_data_mat(:,2), color);
    %return the binned interbead distances
    MSD_InterBead_Distance = binned_data_mat(2:length(binned_data_mat),:);
end
```

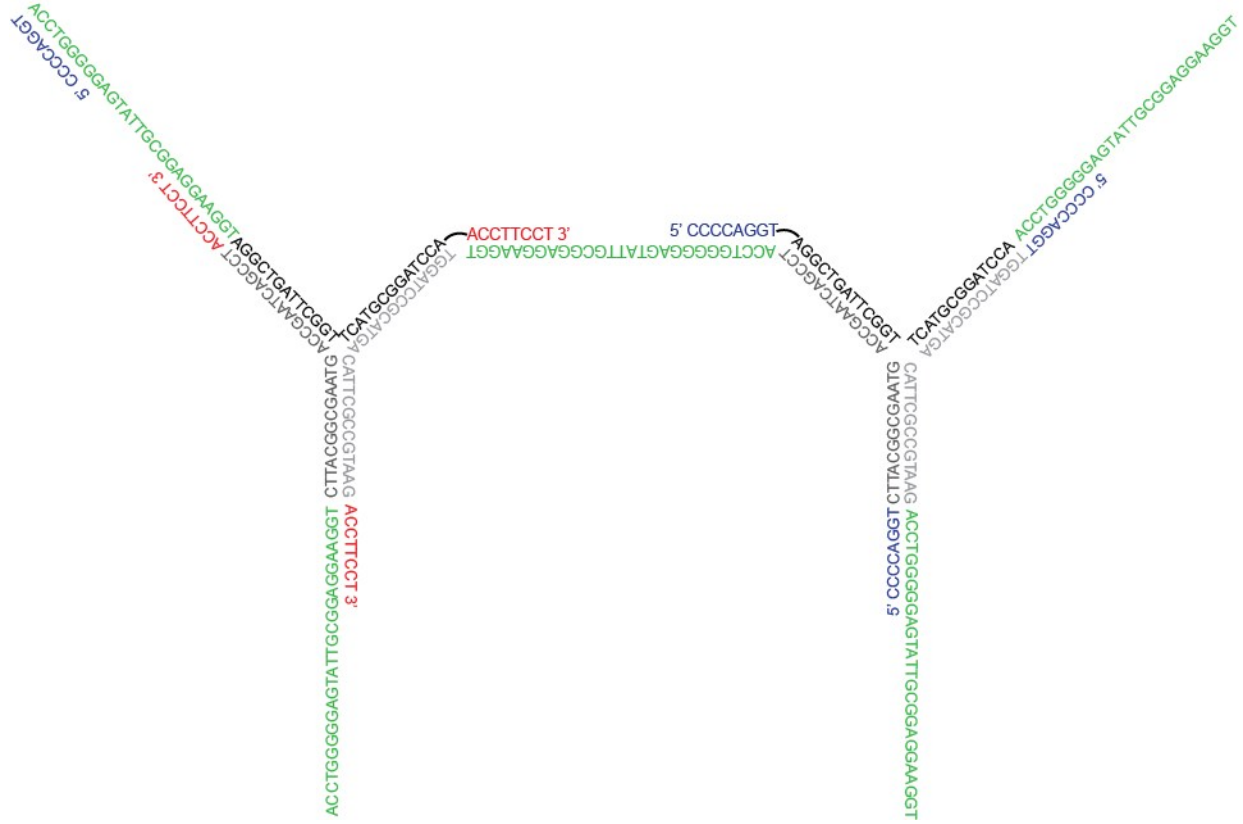
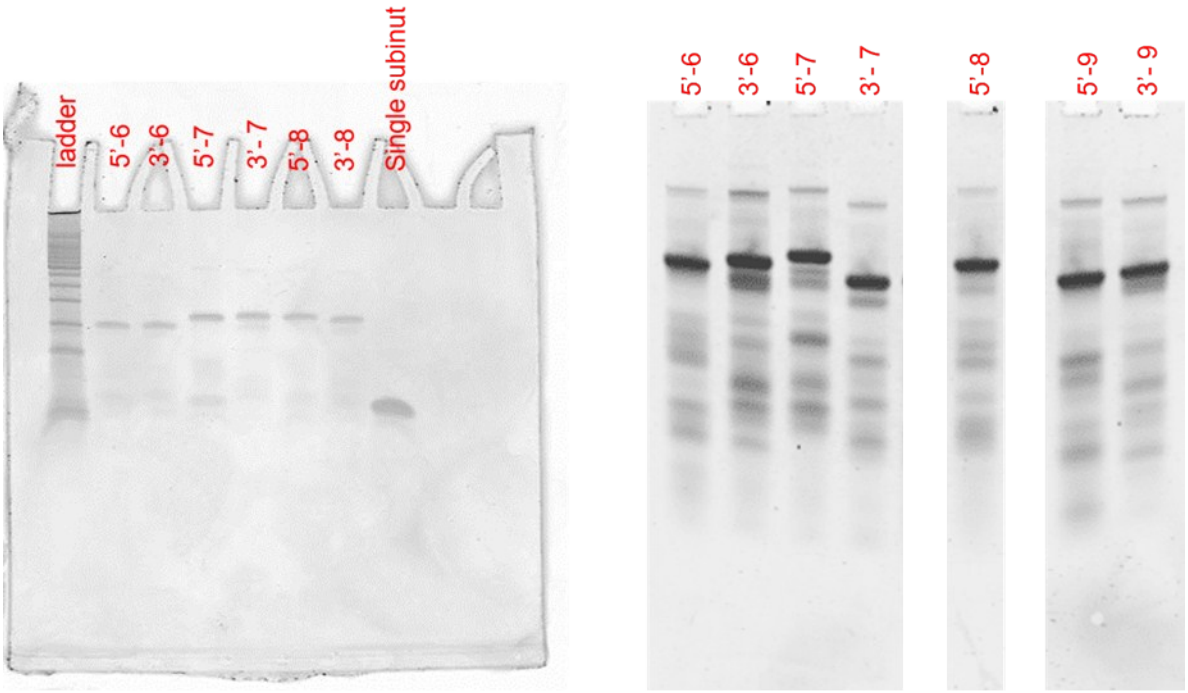


Figure SI 1. Sequence architecture of our Y-DNA model hydrogels. Our Y-DNA architecture consists of two "monomers", each themselves composed of three distinct strands of DNA, containing a double-stranded, Y-shaped core, and pendant ends complementary to bases on the 5' and 5' ends of the adenosine aptamer. In this figure, the aptamer is in green text, the pendant Y-monomer ends are in red and blue, and the three strands comprising the aptamer core are in black, medium gray, and light gray.



Left gel	% Y-monomer		Right gel	% Y-monomer
5'-6	0.8750624		5'-6	0.867541827
3'-6	0.783393188		3'-6	0.651750216
5'-7	0.94274425		5'-7	0.716868367
3'-7	0.873951949		3'-7	0.930305839
5'-8	0.96038657		5'-8	0.899188287
3'-8	0.770580266		3'-8	0.901937573
			5'-9	0.945856924
			3'-9	0.813171513

Figure SI 2. (**Top**) Acrylamide (4-20% gradient) gels confirming production of each Y-monomer permutation from their component subunits. The gel at left includes a single subunit; the gel on the right contains the 5'-9 and 3'-9 Y-monomers; gels were imaged on different dates with different background. (**Bottom**) We estimated the yield by taking the ratio of average background-subtracted intensity between the main, Y-monomer band to the background-subtracted intensity of the entire row, using ImageJ to measure intensities. We have expanded the figure caption to incorporate these suggestions. The yield of desired product Y-monomers is consistently ~80-90%. We suspect that the other products formed consist of either combinations of two but not three of the monomers in varying secondary structures and thus migrating at different rates.

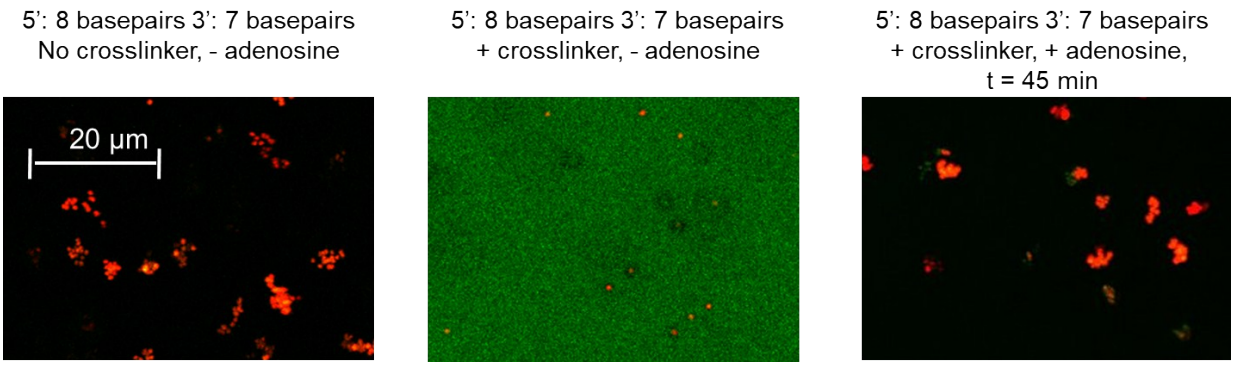


Figure SI 3. Gels containing monomers but no crosslinker show background aptamer fluorescence and high bead mobility, similar to dissolved gels. (Left) 20-frame (~30s) composite image of a mixture containing monomers that bind 8 and 7 base pairs on the aptamer's 5' and 3' termini, but not aptamers. (Center) 20 frame composite image of the same combination of monomers as at left with aptamer added shows high aptamer fluorescence and low bead mobility. (Right) ~45 minutes after the addition of aptamer to this gel, aptamer fluorescence decreases to near background levels and bead mobility approaches that of the monomer-only mixture.