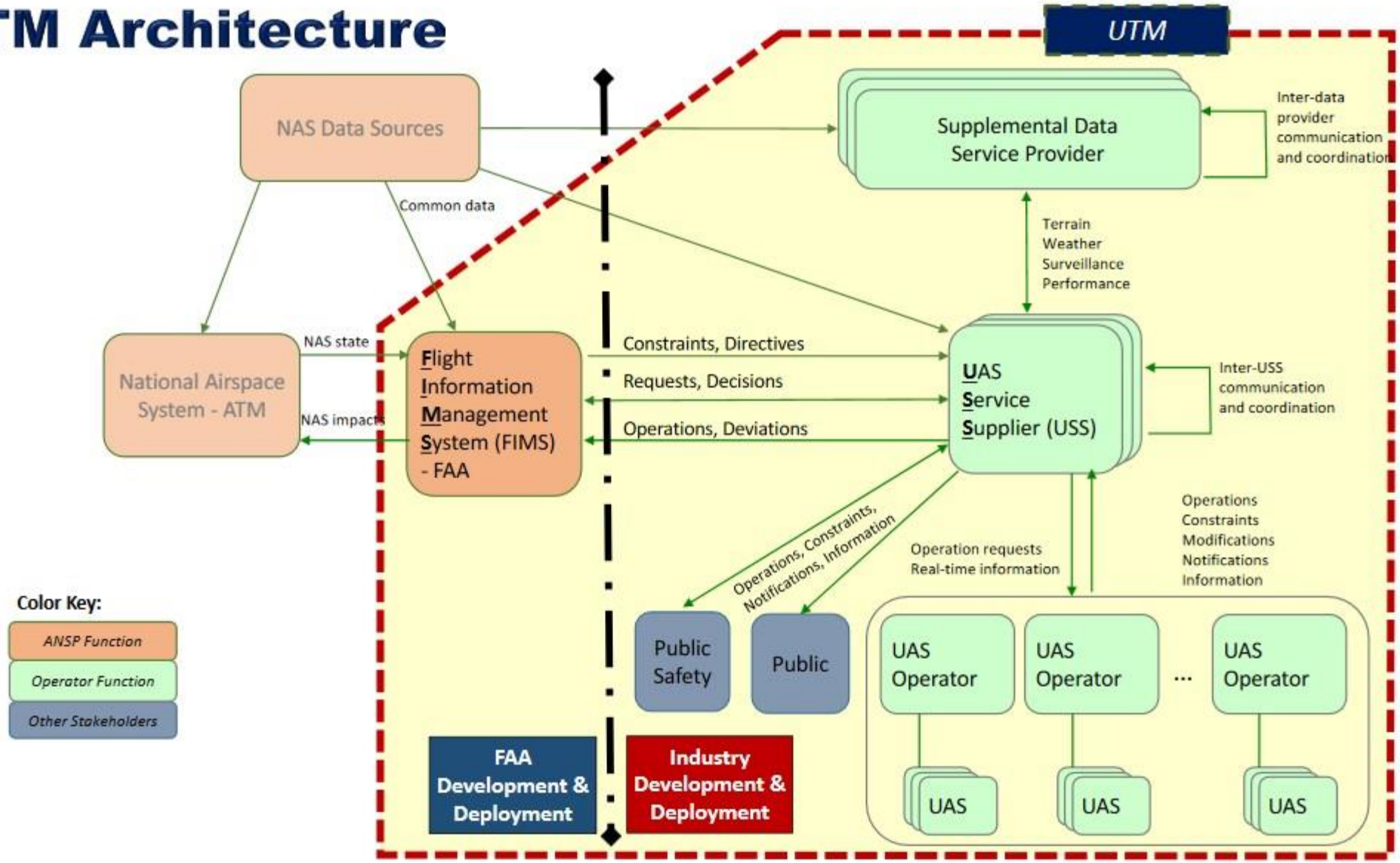
CS 6380 11 Feb 2020
Agent Communication

UAS Traffic Management

UAS Traffic Management (UTM)
Urban Air Mobility (UAM)

How to manage thousands of UAS flights in an urban area?

UTM Architecture

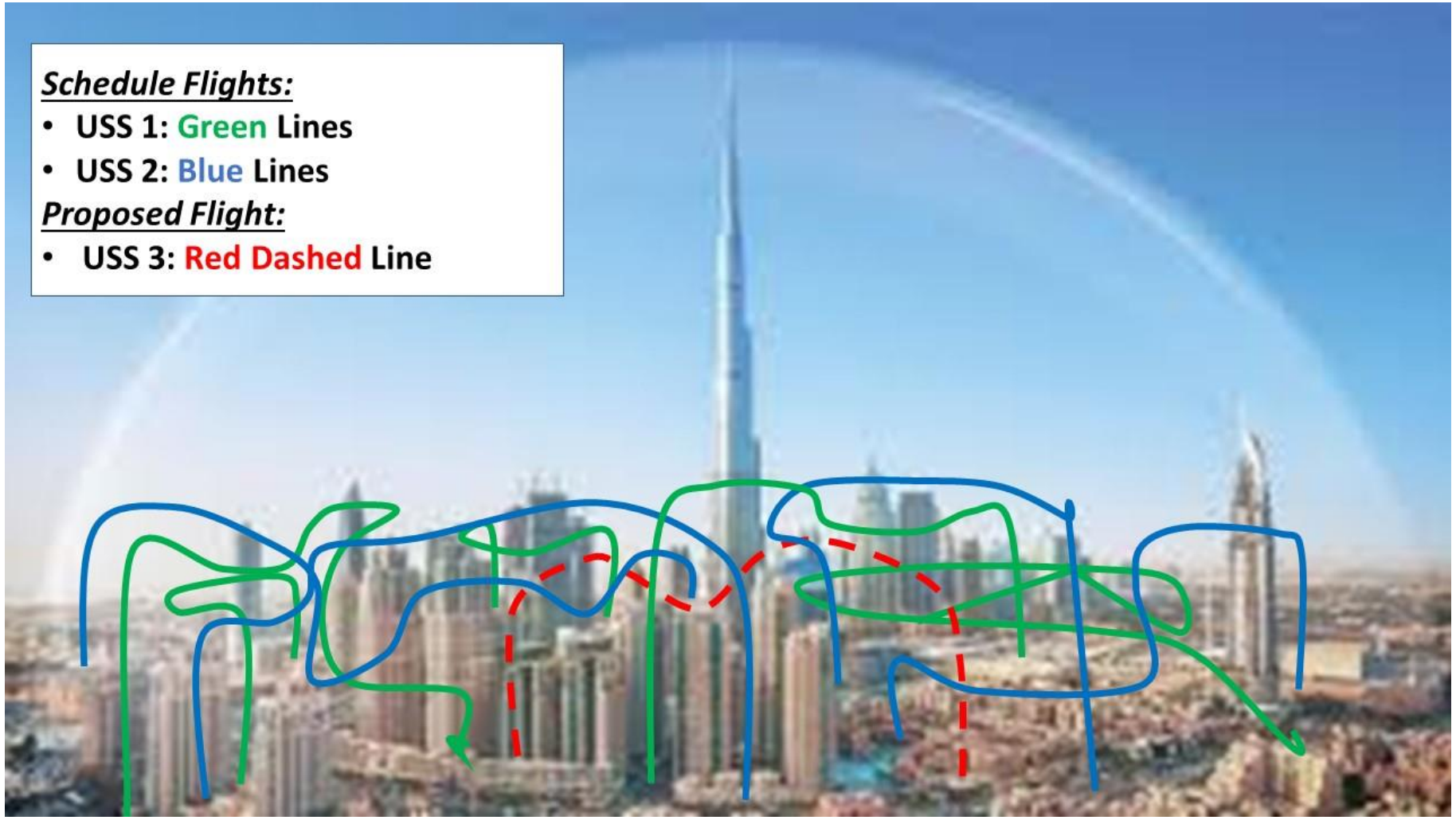


Strategic Deconfliction (SD)

Flights are scheduled so that no two flights ever get within a specified safety distance of each other.

This is specified either in space or in time (or space-time!).

FAA-NASA Strategic Deconfliction Protocol



FAA-NASA SD Method (FNSD)

- USS_new declares domain (grid elements) to GRS
- GRS provides USS's in that domain
- USS_new gets all flights from USS's whose domains intersect the desired new flight path and notify me if new flights scheduled or trajectory changes
- USS_new produces new deconflicted flight path
  - Checks if state assumptions changed (e.g., GRS for new USS, some USS tells it the domain info has changed, and starts over if so)

Ben: Develop and demonstrate an FNSD protocol in IAM

USS will Contract with UAS for Flight

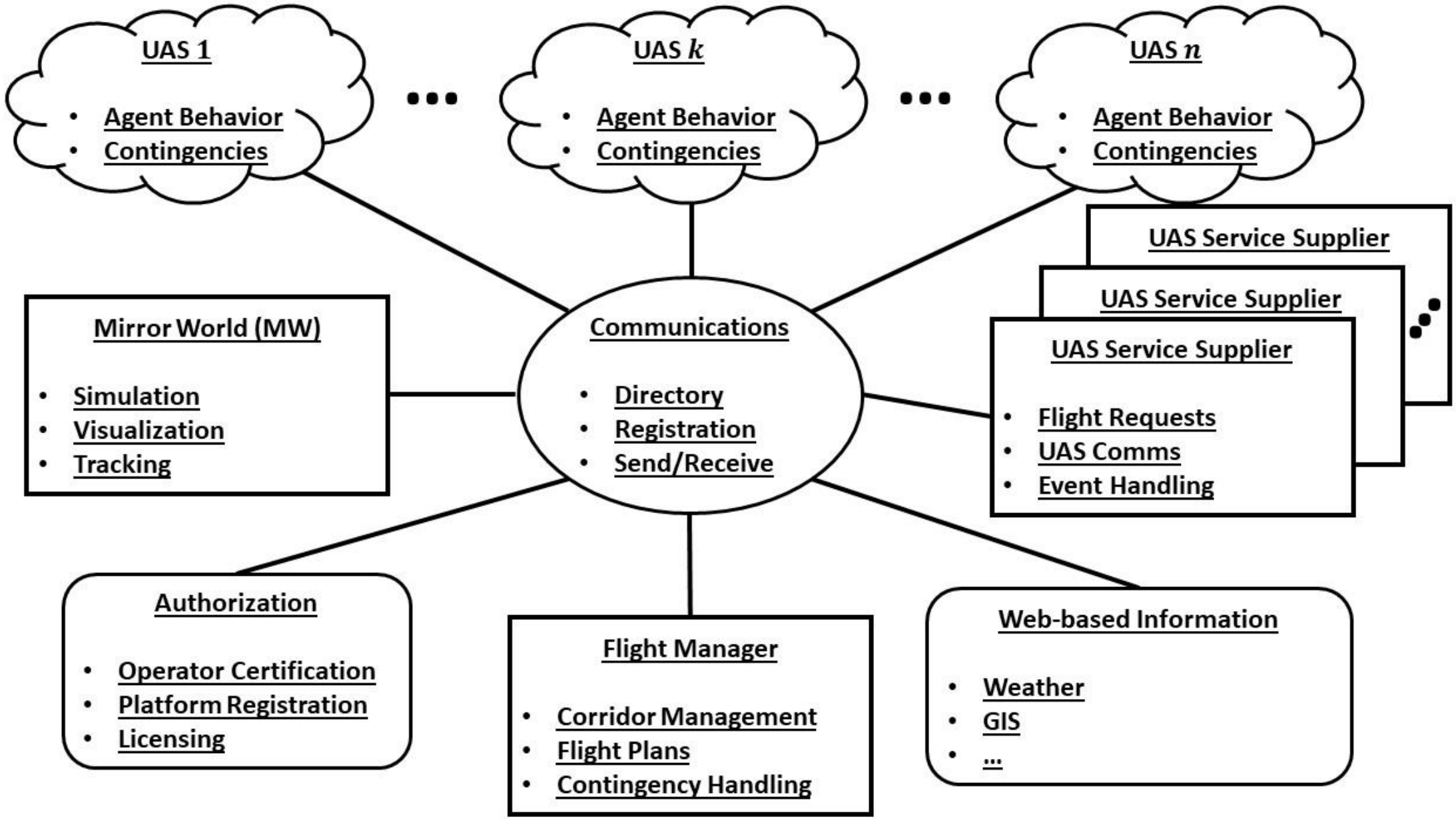
- USS announces call for bids
- UAS sends bid
- USS accepts bid
- UAS accepts contract
- USS sends flight path info

Michael: Develop and demonstrate this protocol in IAM

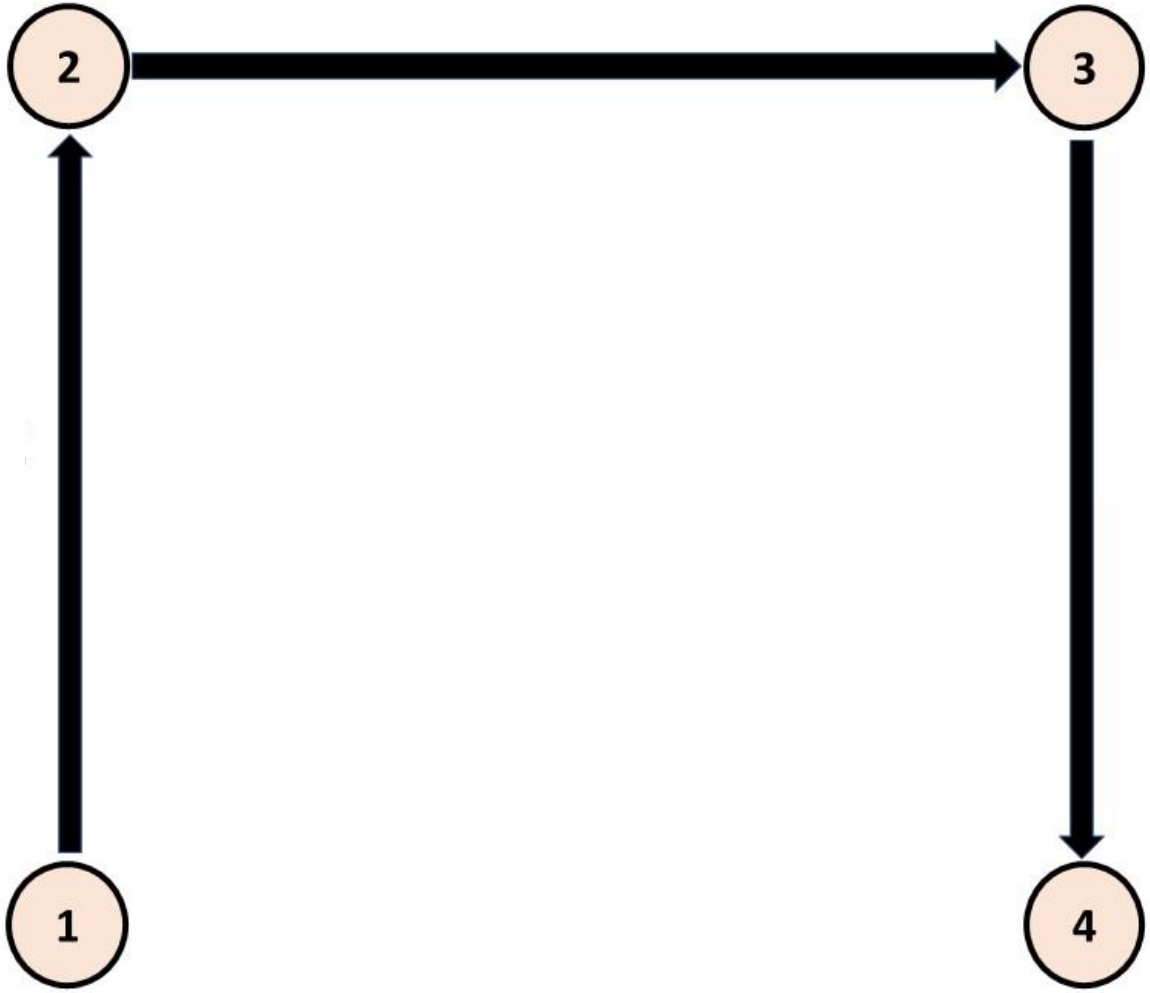
Overall UAM Goal

1. Simulate Aircraft
    1. Simulate aircraft
    2. Have a UAM model
    3. Define performance measures
    4. Interactive simulation
    5. Allow flight modifications
        1. En route
        2. Pre-departure
        3. Re-route
    6. Model takeoff-landing
2. USS Operations posting capability
3. Multi-UAS plan, launch, simulate
4. Load airspace definitions (routes, fixes, Vertiport, etc.)
5. Include weather
6. IT Security

URBAN UTM


UAS 1
Agent Behavior
Contingencies
UAS k
Agent Behavior
Contingencies
UAS n
Agent Behavior
Contingencies
UAS Service Supplier
UAS Service Supplier
UAS Service Supplier
Flight Requests
UAS Comms
Event Handling
Mirror World (MW)
Simulation
Visualization
Tracking
Communications
Directory
Registration
Send/Receive
Authorization
Operator Certification
Platform Registration
Licensing
Flight Manager
Corridor Management
Flight Plans
Contingency Handling
Web-based Information
Weather
GIS
...

Flights (without Lanes!)



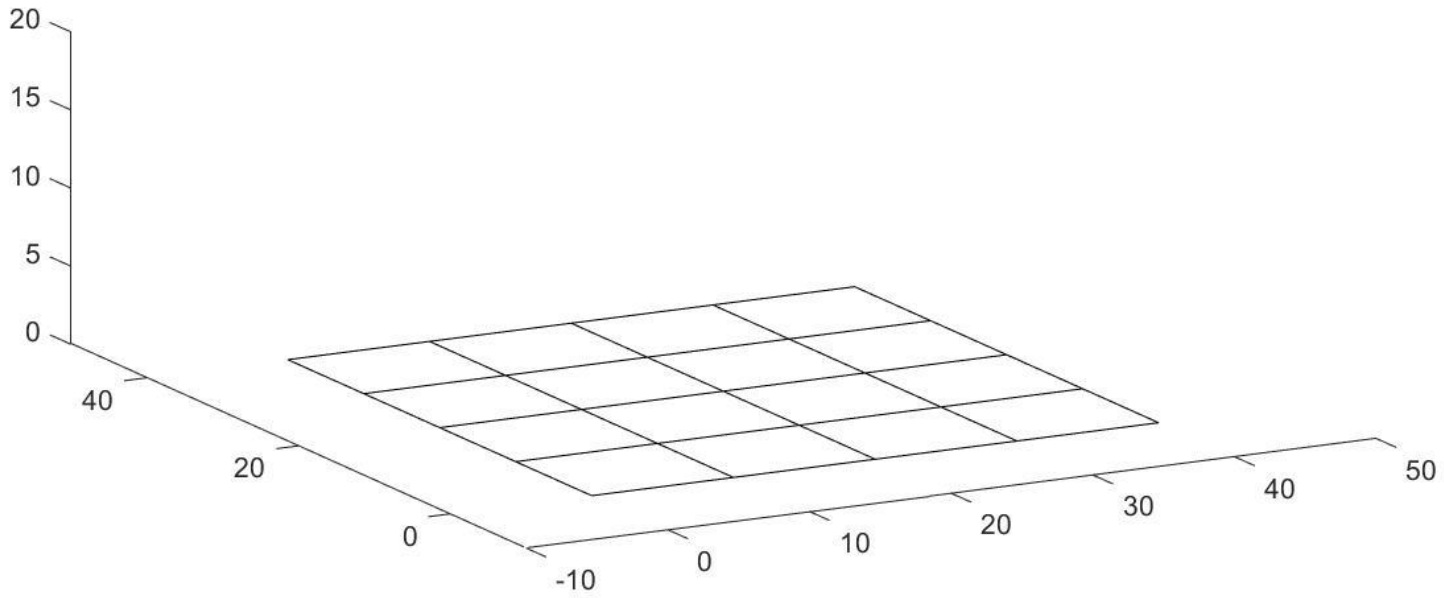
In FNSD flights will go to an altitude between 10 and 20 then across to their destination, then down.

Only rule: never get within headway distance of other flights.

IAM UTM Driver

- function res = CS6380_drive_A3_ABMS(max_t,draw,film)
- % CS6380_drive_A3_ABMS - overall driver for A3 ABMS
- % On input:
- % max_t (float): max simulation time
- % draw (Boolean): draw during simulation
- % film (Boolean): make a movie (not implemented)
- % On output:
- % res (struct vector): results
- % Call:
- % r1 = CS6380_drive_A3_ABMS(100,0,0);

IAM World



IAM Driver

- % clear persistent variables
- num_agents = length(fnames);
- for a = 1:num_agents
- clear(fnames(a).name);
- end
- clear('CS6380_A3_ABMS');
-
- res = CS6380_A3_ABMS(fnames,ports,max_t,del_t,draw);

```
function res = CS6380_A3_ABMS(fnames,ports,max_t,del_t,draw,film)
% CS6380_A3_ABMS - A3 ABMS simulator
% On input:
%    fnames (struct vector): names of agent function (filenames)
%    ports (nx2 array): launch/land ports (x1 y1)
%    max_t (float): max time to simulate
%    del_t (float): time step increment
%    draw (Boolean): display each simulation step
%    film (Boolean): make a movie from data
% On output:
%    res (struct vector): agent info at each step
%      .agents (px9 array): info for each agent
%         col 1: agent type (1: USS; 2: UAS: 3: ATOC)
%         col 2: x coord
%         col 3: y coord
%         col 4: z coord
%         col 5: dx heading in x
%         col 6: dy heading in y
%         col 7: dz heading in z
%         col 8: ground speed
%         col 9: last time called
% Call:
%    r1 = CS6380_A3_ABMS(fnames,roads,max_t,del_t,0,0);
```

IAM ABMS Simulator

```
% Set up percept
    xa = agents(a,2);
    ya = agents(a,3);
    za = agents(a,4);
    dx = agents(a,5);
    dy = agents(a,6);
    dz = agents(a,7);
    sa = agents(a,8);
    ta = agents(a,9);
    percept.x = xa;
    percept.y = ya;
    percept.z = za;
    percept.dx = dx;
    percept.dy = dy;
    percept.dz = dz;
    percept.speed = sa;
    percept.time = cur_time;
    percept.del_t = del_t;
    percept.messages = messages_out;
```

Set up percept

Eventually, need radar, cameras, lidar, etc.

Actions

```
% update heading
        dx = actions(a).dx;
        dy = actions(a).dy;
        dz = actions(a).dz;
        agents(a,5) = dx;
        agents(a,6) = dy;
        agents(a,7) = dz;
        % update speed
        speed = max(min(MAX_SPEED,actions(a).speed),0);
        agents(a,8) = speed;
        % update position
        xa = agents(a,2);
        ya = agents(a,3);
        za = agents(a,4);
        pt = agents(a,2:4)' + speed*[dx;dy;dz]*del_t;
        pt(3) = max(0,pt(3));
        agents(a,2) = pt(1);
        agents(a,3) = pt(2);
        agents(a,4) = pt(3);
```

Eventually, need physics model

Let’s Consider the ATOC Agent

```
MAX_SPEED = 10;
BROADCAST = '*';
MY_ID = 'ATOC_tom_1';
TELEMETRY = 'TELEMETRY';
ANNOUNCE_SELF = 'ANNOUNCE_SELF';
USS_TYPE = 'USS';
UAS_TYPE = 'UAS';
ATOC_TYPE = 'ATOC’;
```

persistent state USS UAS flights AgentNames AgentTypes A_table uit fig

Handling Messages

```
if ~isempty(messages_in)
        num_messages_in = length(messages_in);
        for m = 1:num_messages_in
            mess_from = messages_in(m).From;
            mess_to = messages_in(m).To;
            mess_type = messages_in(m).Type;
            mess_data = messages_in(m).Data;
```

Handling Messages (cont'd)

```
if ~strcmp(mess_from,MY_ID) % not from myself
        if strcmp(mess_from(1:3),USS_TYPE) % from USS
            % handle USS
            [USS,index] = CS6380_index_USS(USS,mess_from);
        elseif strcmp(mess_from(1:3),UAS_TYPE) % from UAS
            [UAS,index] = CS6380_index_UAS(UAS,mess_from);
            if strcmp(mess_type,TELEMETRY)
                flights = [flights;mess_data];
            end
        end
        [AgentNames,AgentTypes] = ...
            CS6380_insert_agent_info(AgentNames,...
            AgentTypes,mess_from);
    end
```

Having Multiple States during One Time Step

```
done = 0;
while done==0
   switch state
…
       case 3  % Flight Display
          CS6380_display_flights(flights);
          state = 4;
       case 4 % exit state
          state = 1;
          return
   end
```

USS Agent

```
persistent state USS UAS

messages_out = [];

if isempty(state)
    state = 1;
    USS = [];
    UAS = [];
    messages_out = CS6380_make_message(BROADCAST,MY_ID,ANNOUNCE_SELF,[]);
end
```

USS Agent

```
if ~isempty(messages_in)
        num_messages_in = length(messages_in);
        for m = 1:num_messages_in
            mess_from = messages_in(m).From;
            mess_to = messages_in(m).To;
            mess_type = messages_in(m).Type;
            if ~strcmp(mess_from,MY_ID) % not from myself
                if strcmp(mess_from(1:3),USS_TYPE) % from USS
                    % handle USS
                    [USS,index] = CS6380_index_USS(USS,mess_from);
                elseif strcmp(mess_from(1:3),UAS_TYPE) % from UAS
                    % handle UAS
                    [UAS,index] = CS6380_index_UAS(UAS,mess_from);
                end
            end
        end
    end
```

UAS Agent

persistent state USS UAS state_vars
persistent lanes cur_lane num_lanes

```
if isempty(state)
    state = 1;
    USS = [];
    UAS = [];
    messages_out(1).To = BROADCAST;
    messages_out(1).From = MY_ID;
    messages_out(1).Type = ANNOUNCE_SELF;
    state_vars(1) = 0; % x coord
    state_vars(2) = 0; % y coord
    state_vars(3) = 0; % z coord
    state_vars(4) = 1; % dx
    state_vars(5) = 0; % dy
    state_vars(6) = 0; % dz
    state_vars(7) = 0; % speed
    lanes = [0 0 0 0 0 20; 0 0 20 10 0 20; 10 0 20 10 0 0];
    num_lanes = length(lanes(:,1));
    cur_lane = 1;
end
```

UAS Agent

```
cur_loc = [xa;ya;za];
    goal = lanes(cur_lane,4:6)';
    dist = norm(goal-cur_loc);
    if dist<DIST_THRESH
        cur_lane = cur_lane + 1;
        if cur_lane<=num_lanes
            goal = lanes(cur_lane,4:6)';
        end
    end
```

UAS Agent:

Set Goal

```
if cur_lane<=num_lanes
        dir = goal - cur_loc;
        dir = dir/norm(dir);
        dx = dir(1);
        dy = dir(2);
        dz = dir(3);
        dist = norm(goal-cur_loc);
        speed = min(MAX_SPEED,dist/del_t);
```

UAS Agent:

Set heading & speed

```
else
        dx = 0;
        dy = 0;
        dz = 0;
        speed = 0;
    end
```

UAS Agent:

Set post-flight vals

UAS Agent:

Telemetry Message

```
messages_out = CS6380_make_message(BROADCAST,MY_ID,TELEMETRY,...
        [xa,ya,za,dx,dy,dz,speed]);
```