V-REP

and its Matlab interface

Marc Freese

COPPELIA ROBOTICS

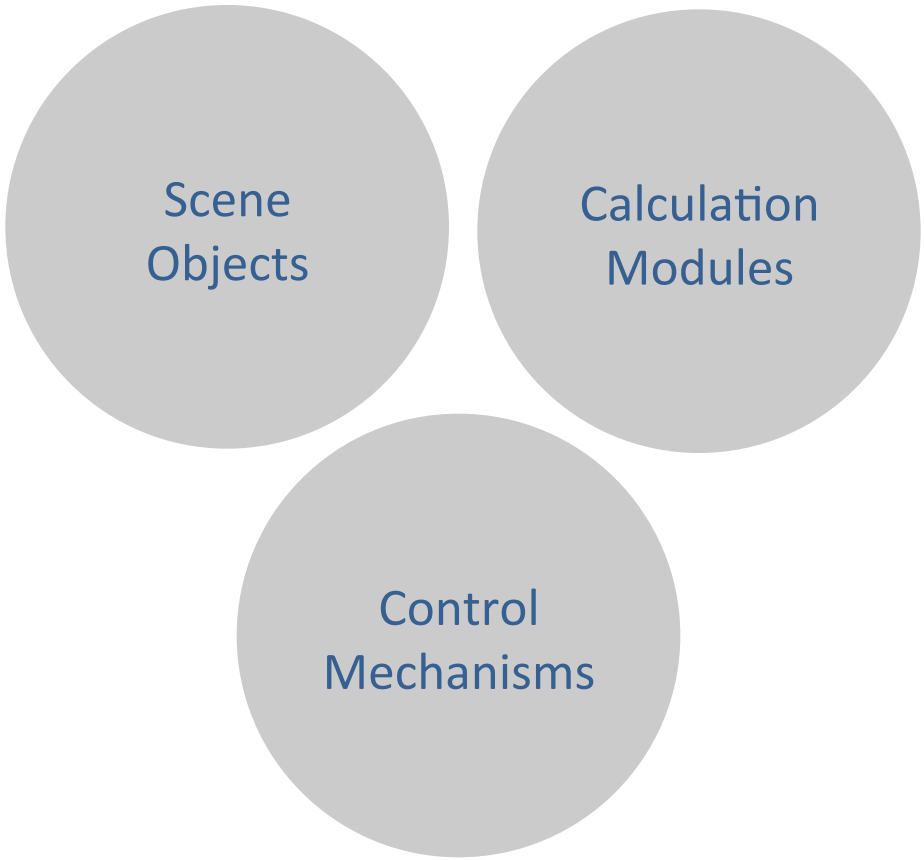
www.coppeliarobotics.com

Robot simulator

Why simulation ?

Some examples

3 Central Elements



Scene Objects

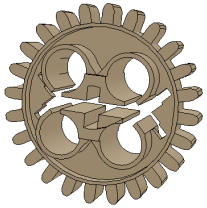
Scene Objects

Joints

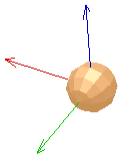
Cameras

Lights

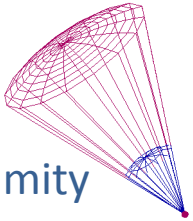
Shapes



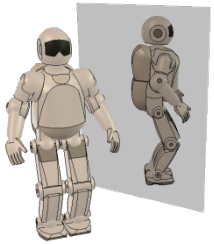
Dummies / reference frames



Proximity sensors

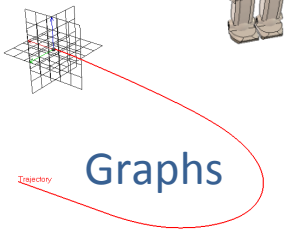


Mirrors

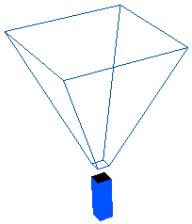


Force / torque sensors

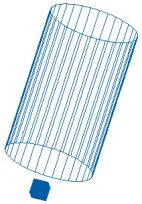
Graphs



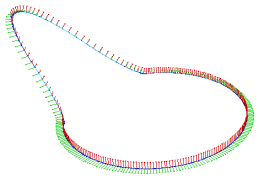
Vision sensors



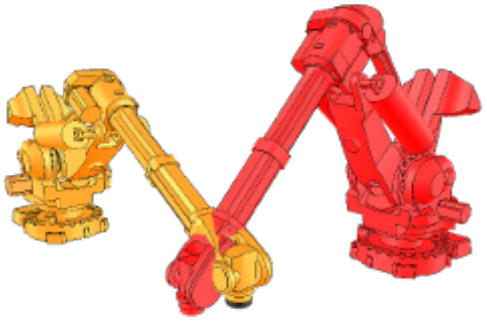
Mills



Paths / trajectories



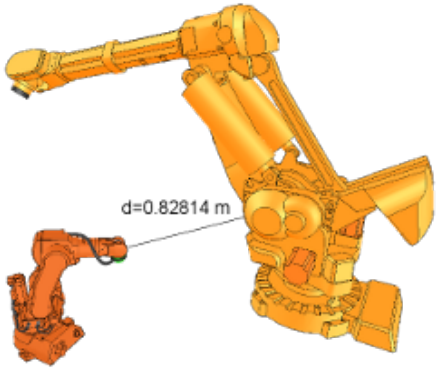
Calculation Modules



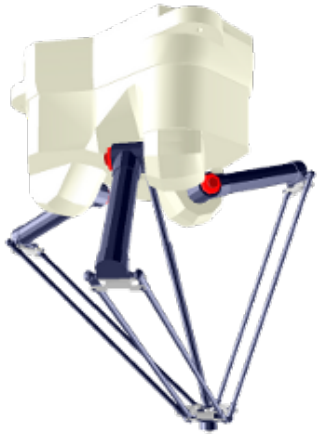
Collision detection

Calculation Modules

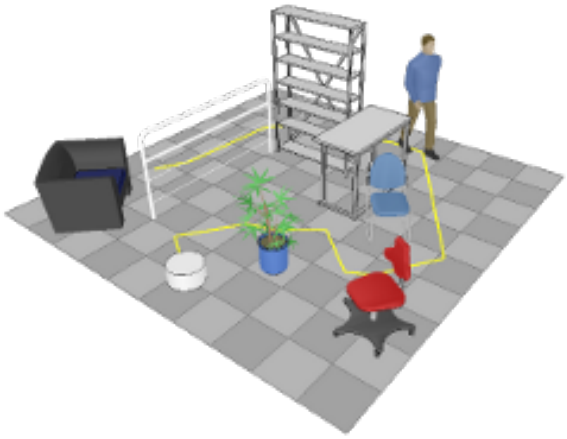
Physics / Dynamics



Minimum distance calculation

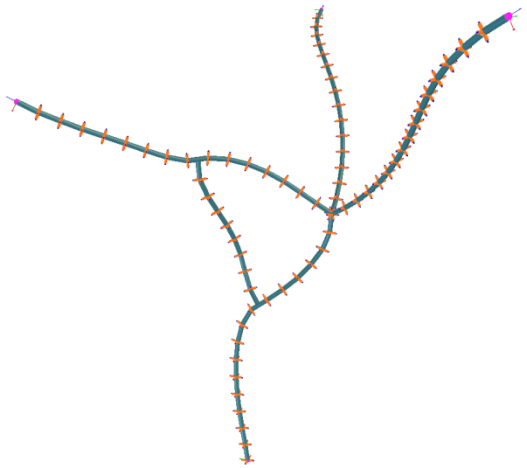


Forward / Inverse kinematics



Path / motion planning

Kinematics and Distance Calculation

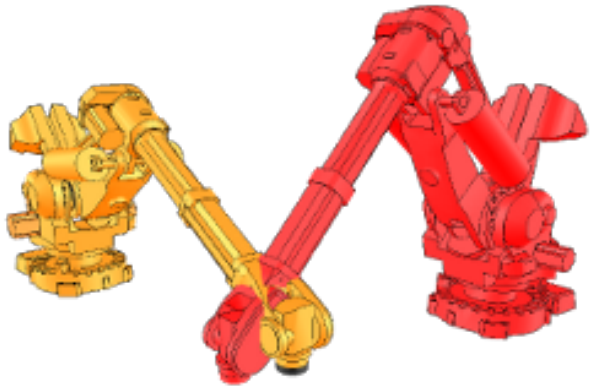


Inverse / forward Kinematics

- Any mechanism: redundant, branched, closed, etc.
- Damped / undamped resolution
- Weighted resolution
- Conditional resolution
- Obstacle avoidance

Collision Detection

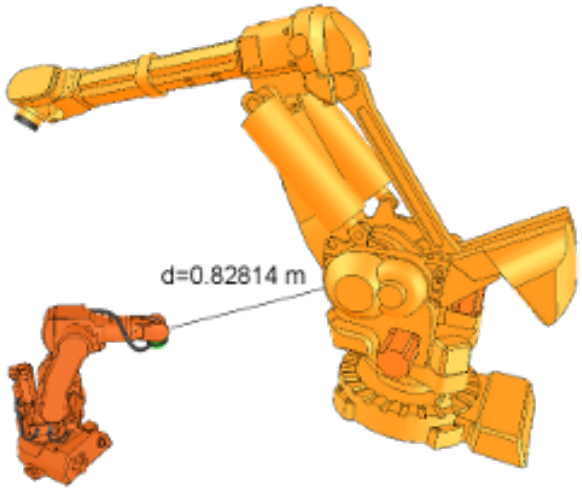
Any mesh (also open / concave / polygon soups)

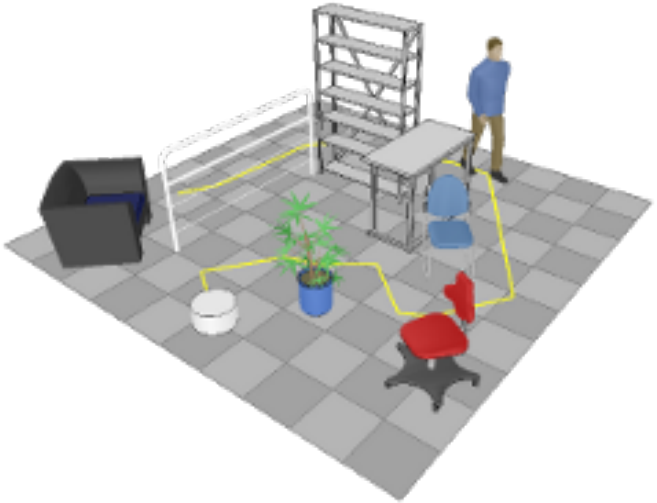


Collision Detection and Path Planning

Minimum Distance Calculation

Any mesh (also open / concave / polygon soups)

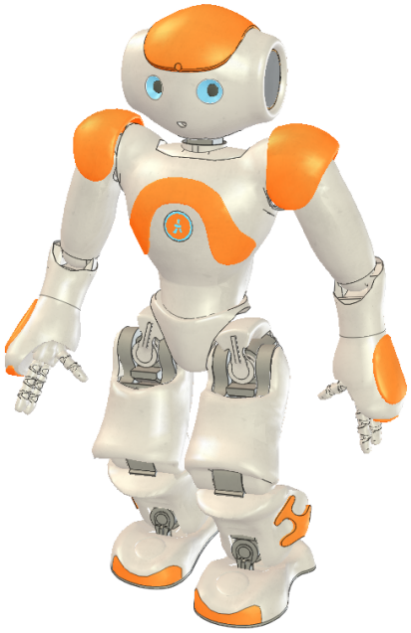




Path / Motion Planning

- Holonomic in 2-6 dimensions
- Non-holonomic for car-like vehicles
- Motion planning for kinematic chains

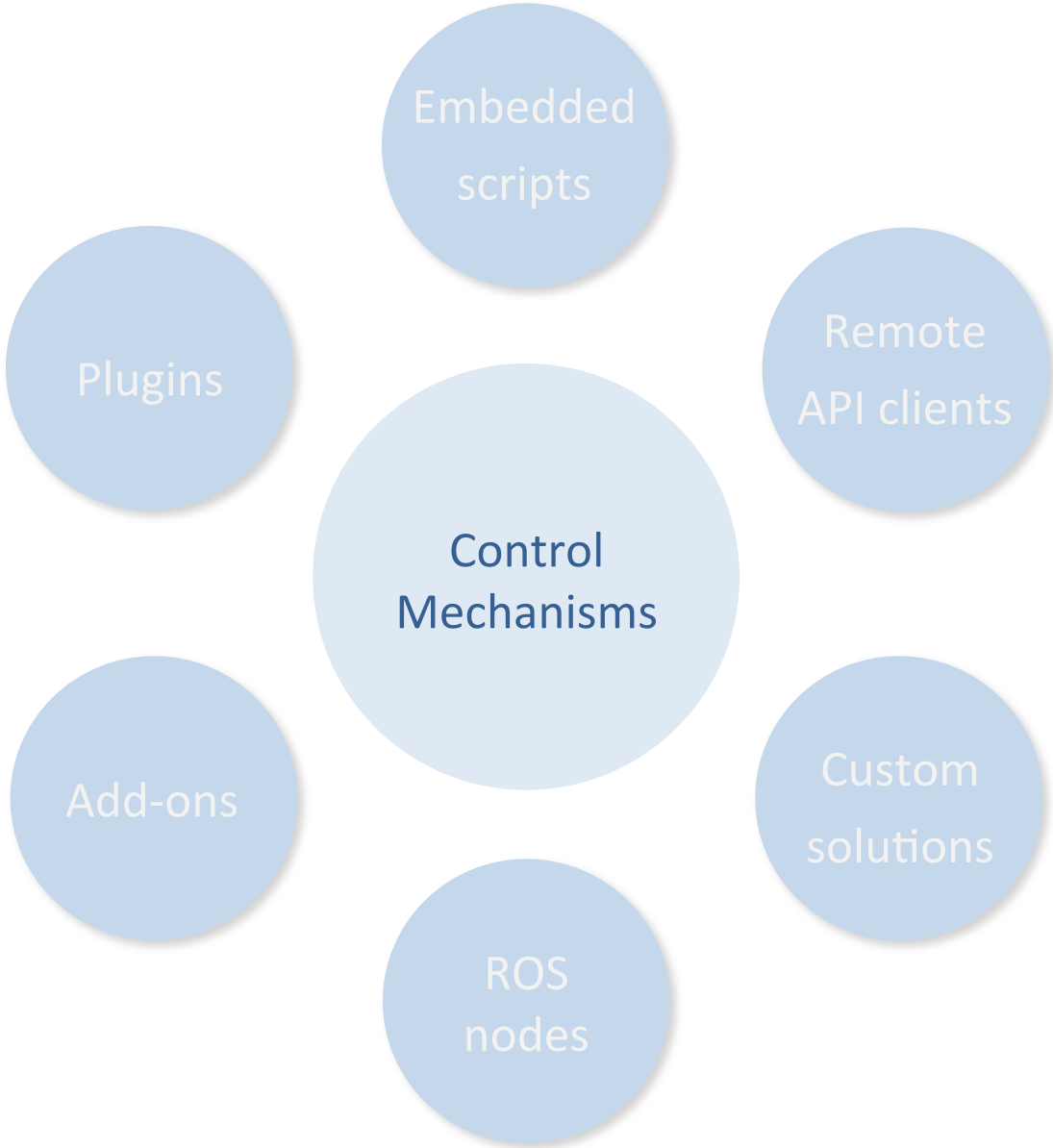
Dynamics



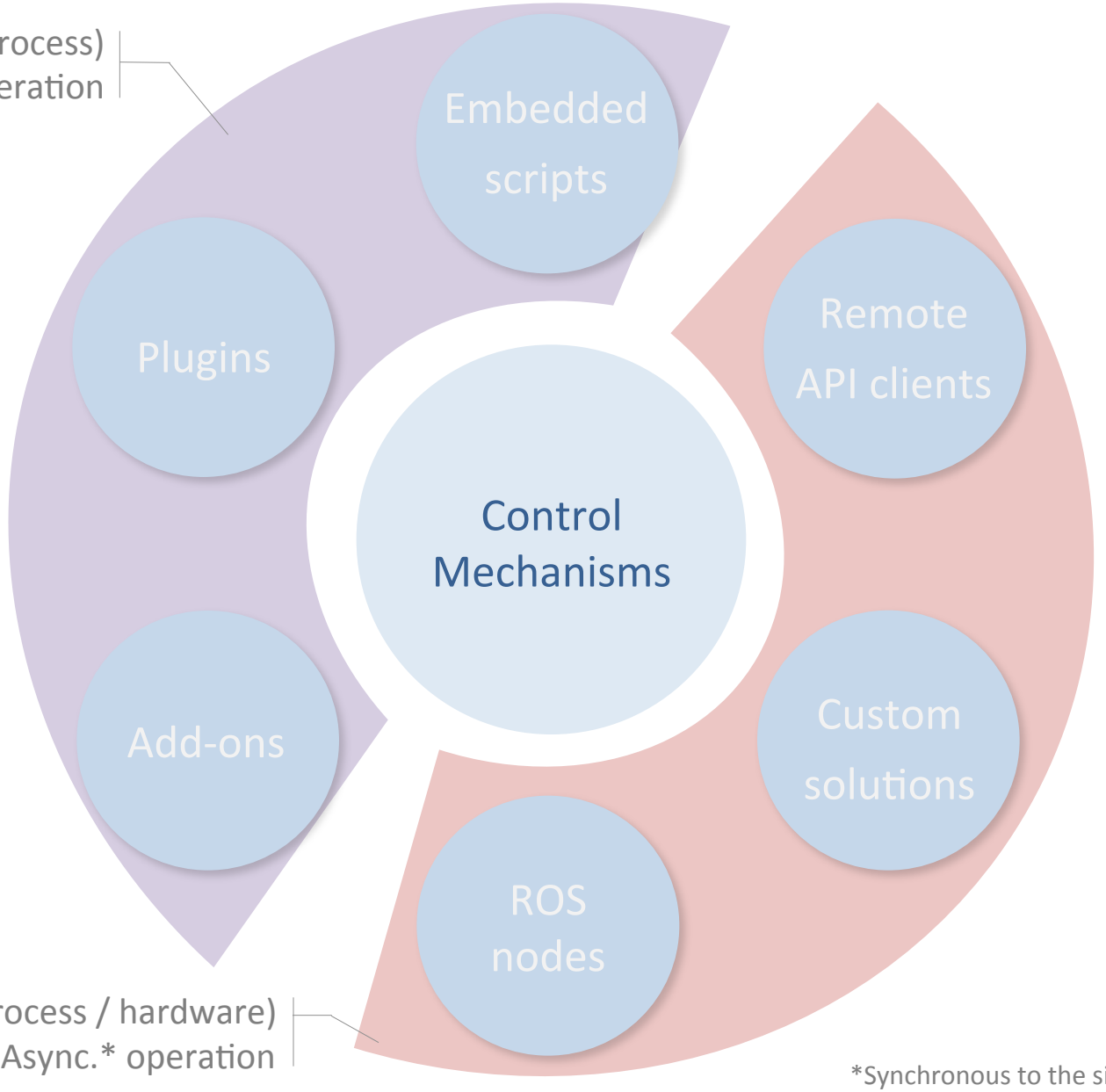
Dynamics / Physics

- 3 physics engines: Bullet Physics
  Open Dynamics Engine
  Vortex Dynamics
- Simple mouse click to switch
- Dynamic particles to simulate air or water jets
- Can work hand-in-hand with kinematics module

Control Mechanisms



Local and Remote Interfaces



*Synchronous to the simulation loop

Embedded Script Advantages 1/4

Controller Integration

Plugins

Simulation model

(meshes, joints, sensors, etc.)

+

Plugin

(controller)

2 items

Embedded scripts

Simulation model

(meshes, joints, sensors, etc.)

+

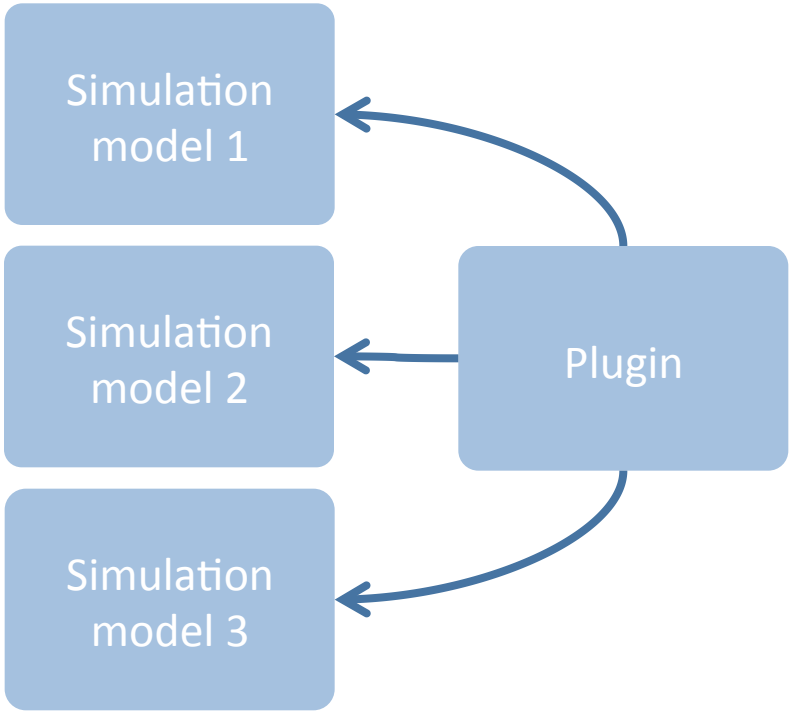
controller

1 item

Embedded Script Advantages 2/4

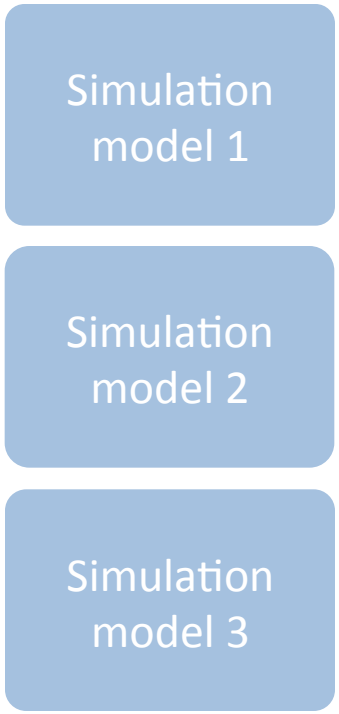
Scalability

Plugins



Plugin has to manage instances

Embedded scripts

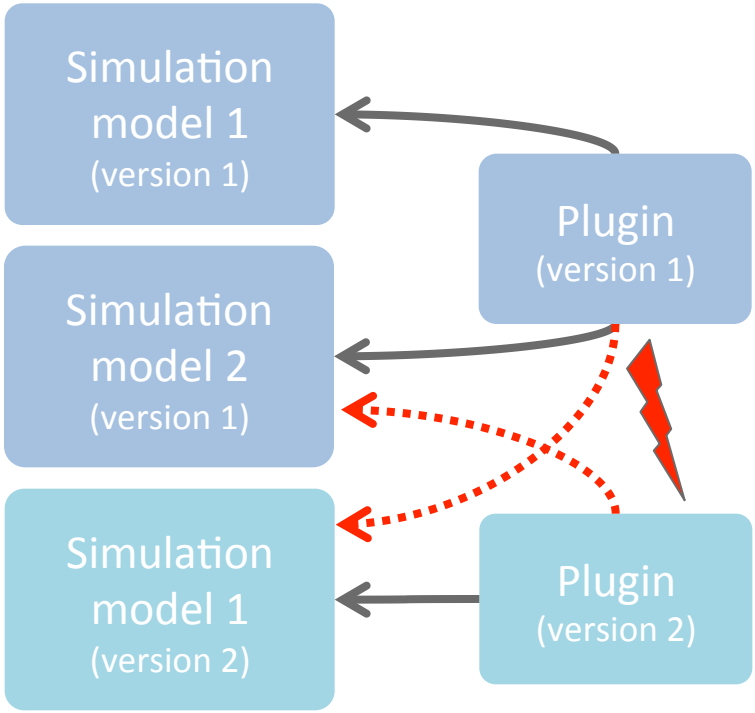


Scalability is inherent

Embedded Script Advantages 3/4

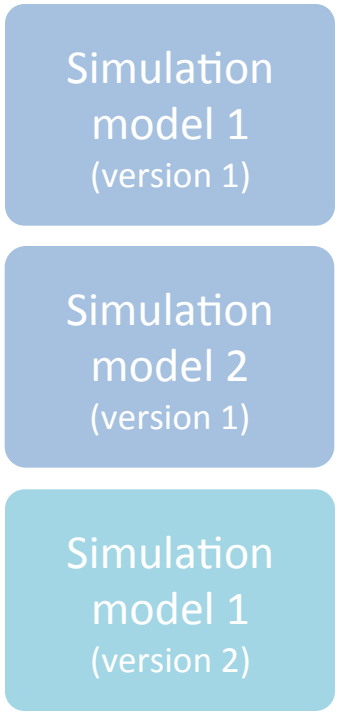
Version Conflicts

Plugins



High chances for conflicts

Embedded scripts



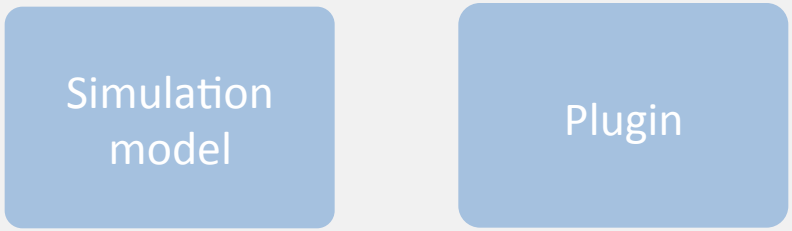
No chances for conflicts

Embedded Script Advantages 4/4

Portability

Plugins

Same OS



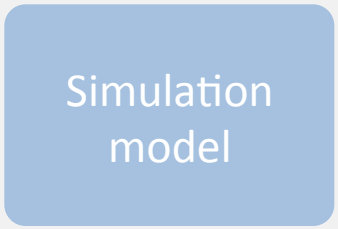
2 files

Different OS

Many files
Compilation required
OS-specific problems

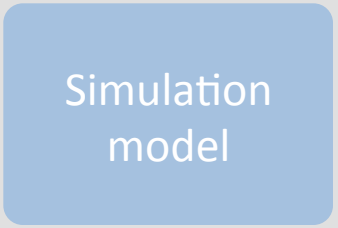
Embedded scripts

Same OS



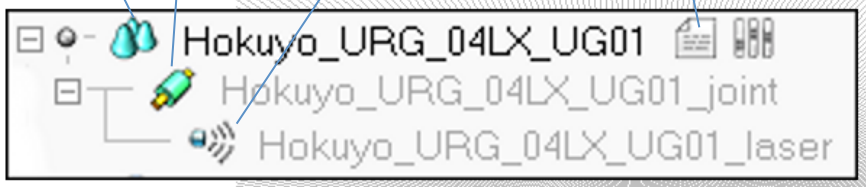
1 file

Different OS



1 file
No compilation required
No OS-specific problems

Embedded Scripts – Simple Example



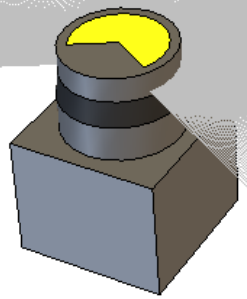
Shape

Joint

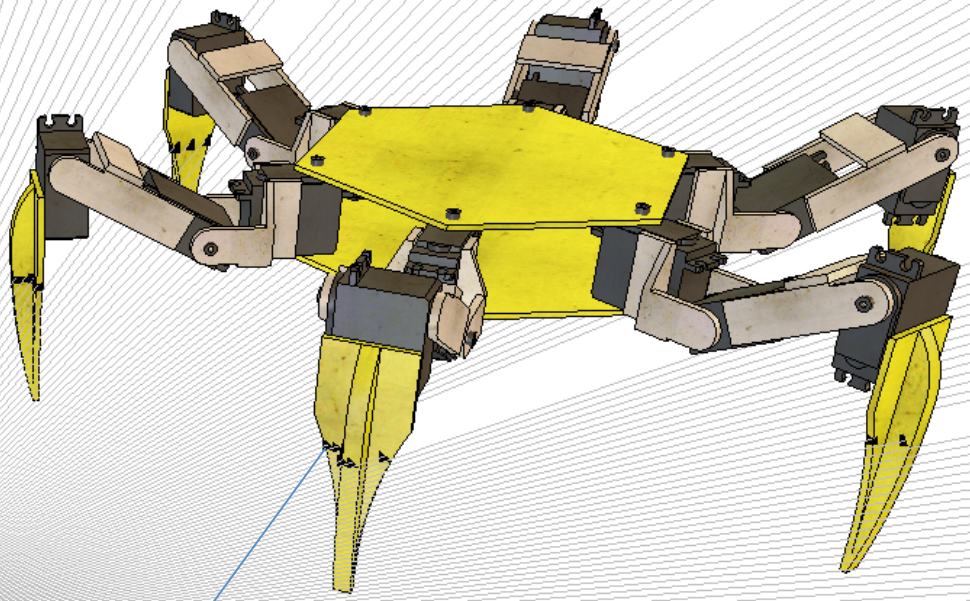
Proximity sensor

Embedded script

Hokuyo model hierarchial representation



Hokuyo simulation model

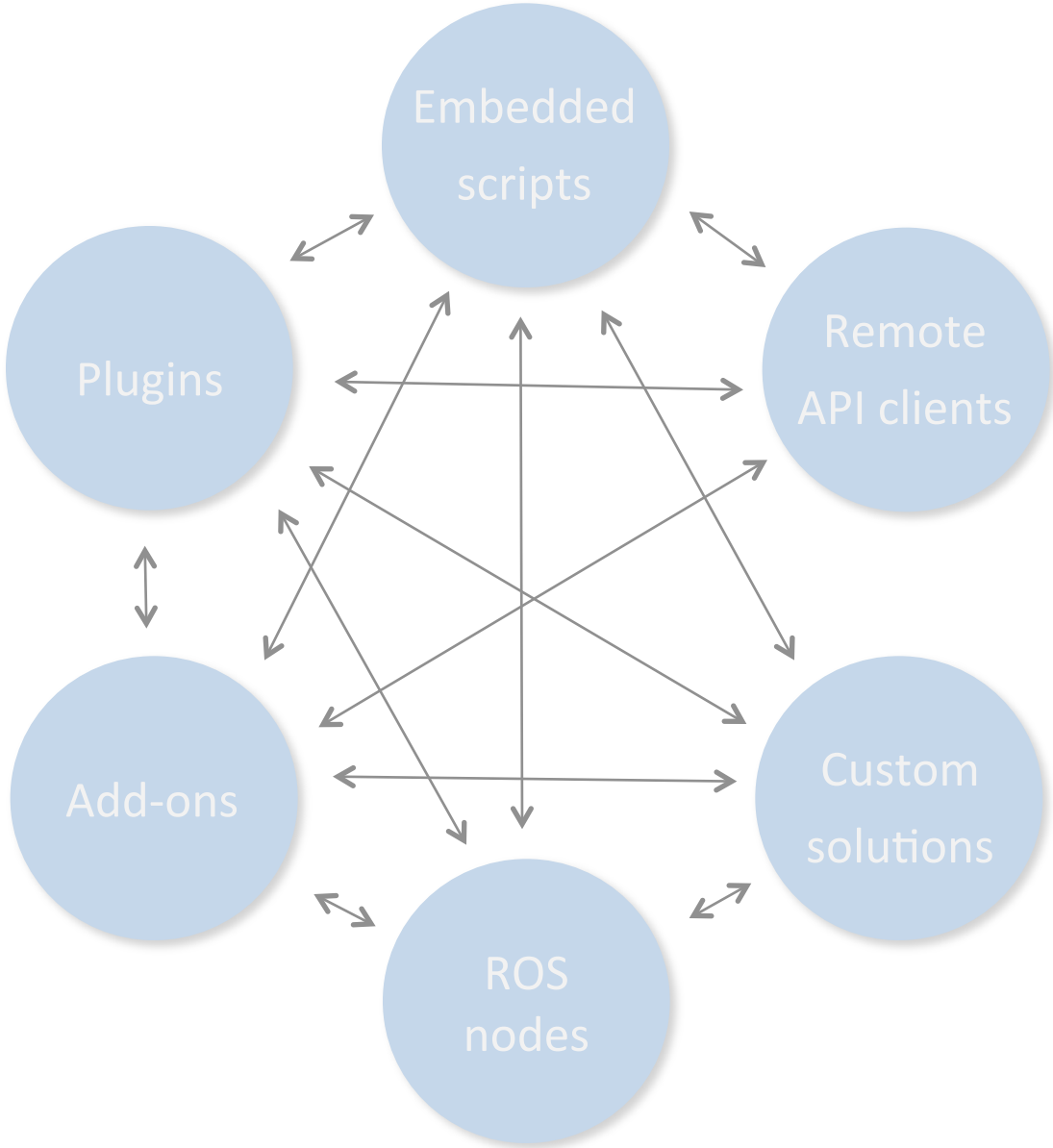


Detected points

Hexapod simulation model

Distance sampling rays

Collaborative Control Mechanisms



Example of Collaborative Mechanism 1 / 3

Plugin

```
void SCRIPT_DO_SOME_MAGIC_CALLBACK(SLuaCallBack* p)
{
  ... ( gets called when a script calls „simxExt_doSomeMagic“ )
}


// Initialization phase of plugin: register new script commands:
#define SCRIPT_DO_SOME_MAGIC "simExt_doSomeMagic"
int inArgs[]={2,sim_lua_arg_int,sim_lua_arg_float|sim_lua_arg_table};

simRegisterCustomLuaFunction(SCRIPT_DO_SOME_MAGIC,strConCat("number value1,
        number value2=",SCRIPT_DO_SOME_MAGIC,"(number index,table inVals)"),
        inArgs,SCRIPT_DO_SOME_MAGIC_CALLBACK);
```

Registers and handles the custom script API function „simExt_doSomeMagic“

Calls the custom API function „simExt_doSomeMagic“

Embedded script

```
returnData1,returnData2=simExt_doSomeMagic(arg1,arg2)
```

Example of Collaborative Mechanism 2 / 3

Enable image streaming to ROS

```
-- Following in the script initialization phase (executed just once):

-- Retrieve the handle of the vision sensor we wish to stream:
visionSensorHandle=simGetObjectHandle('Vision_sensor')

-- Now enable topic publishing and streaming of the vision sensor's data:
topicName=simExtROS_enablePublisher('visionSensorData',1,
        simros_strmcmd_get_vision_sensor_image,visionSensorHandle,0,'')

-- Retrieve the handle of the passive vision sensor. We will use
-- the passive vision sensor to visualize received data:
passiveSensorHandle=simGetObjectHandle('PassiveVision_sensor')

-- Now enable a topic subscription:
simExtROS_enableSubscriber(topicName,1,
        simros_strmcmd_set_vision_sensor_image,passiveSensorHandle,0,'')
```

Embedded script

Enable image streaming from ROS

Example of Collaborative Mechanism 3 / 3

Remote API client

Appends coordinate information to the signal „objCreation“

```
int pos[3]={1.0f,2.0f,3.0f};
simxAppendStringSignal("objCreation",(simxChar*)pos,3*4,simx_opmode_oneshot);
```

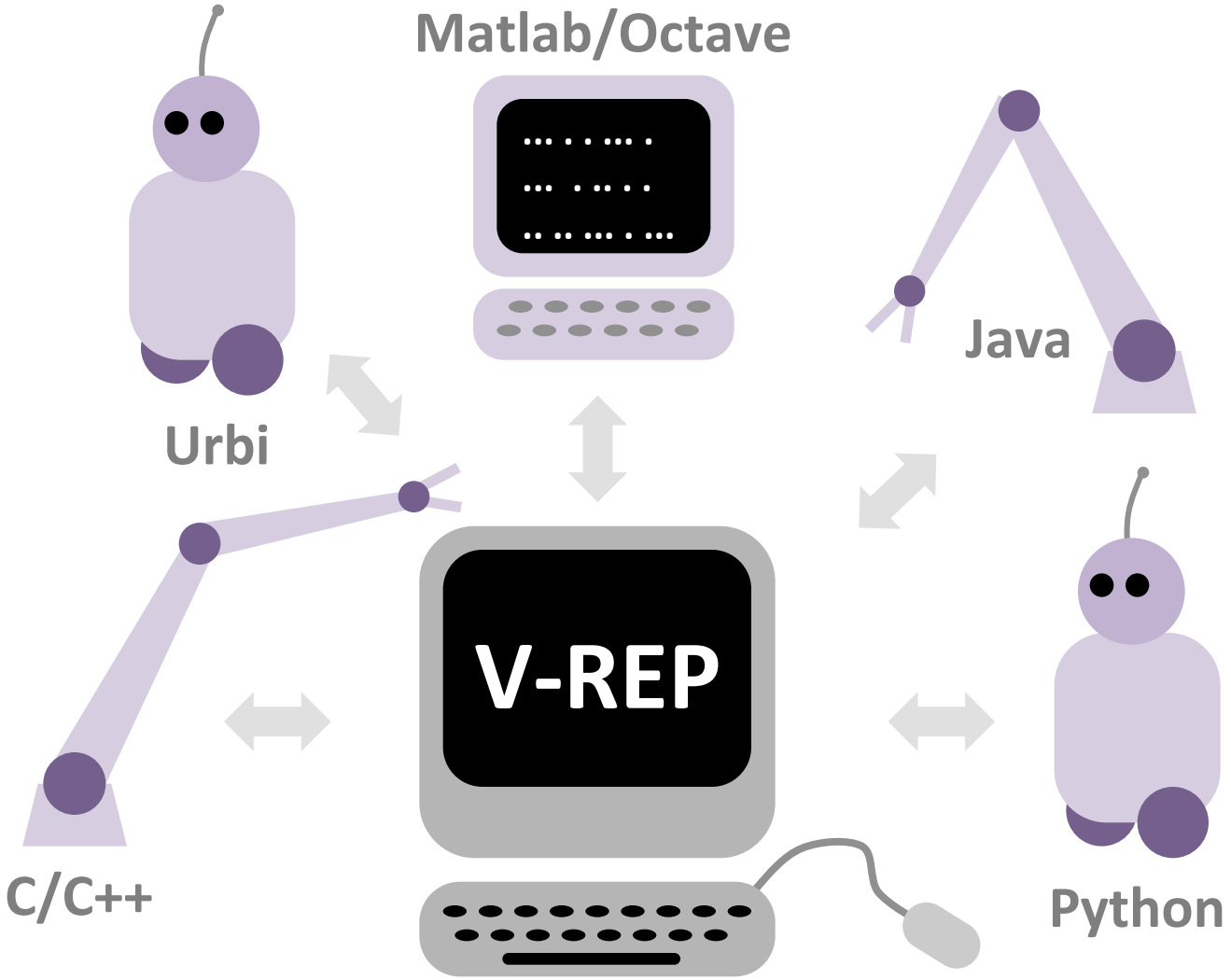
Creates cylinders at the coordinates indicated in the signal „objCreation“

Embedded script

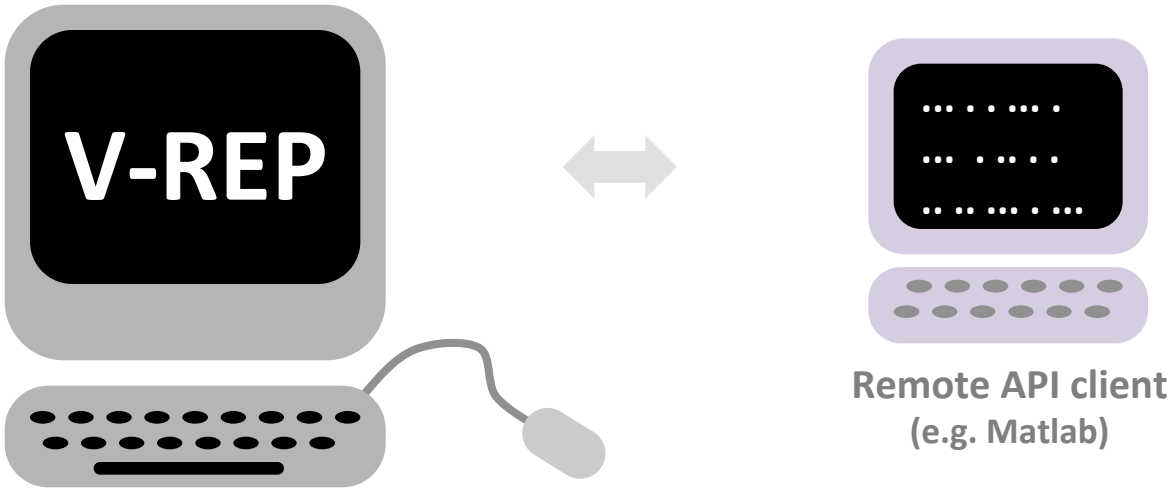
```
signalData=simGetStringSignal("objCreation")
simClearStringSignal("objCreation")
if signalData then
  data=simUnpackFloats(signalData)
  for i=0,(#data-1)/3,1 do
    handle=simCreatePureShape(2,22,{0.1,0.1,0.1},0.1)
    simSetObjectPosition(handle,-1,{data[3*i+1],data[3*i+2],data[3*i+3]})
  end
end
```

Remote API

Runs on any hardware, lightweight, several languages

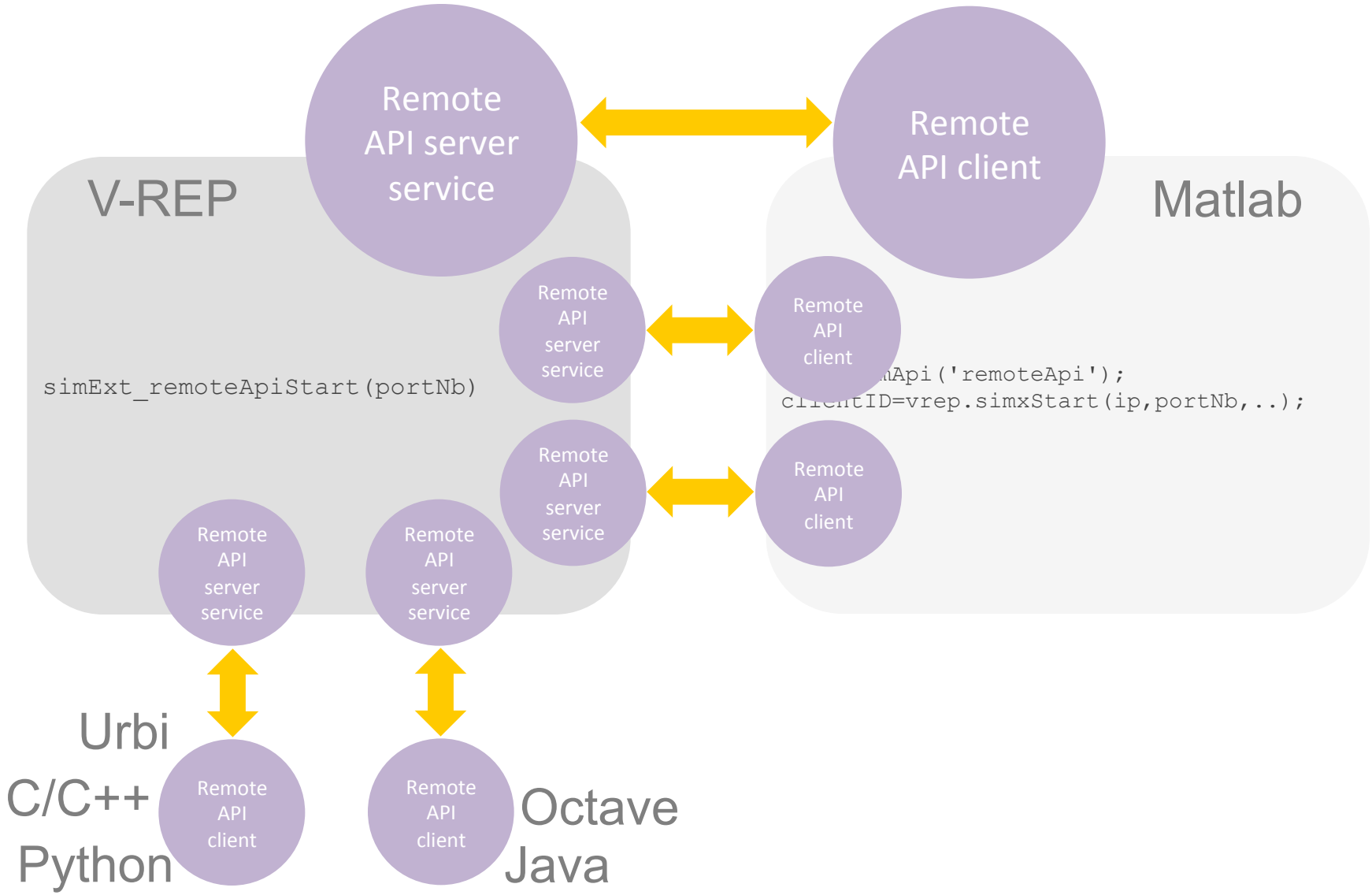


Remote API



Server ⟷ Client

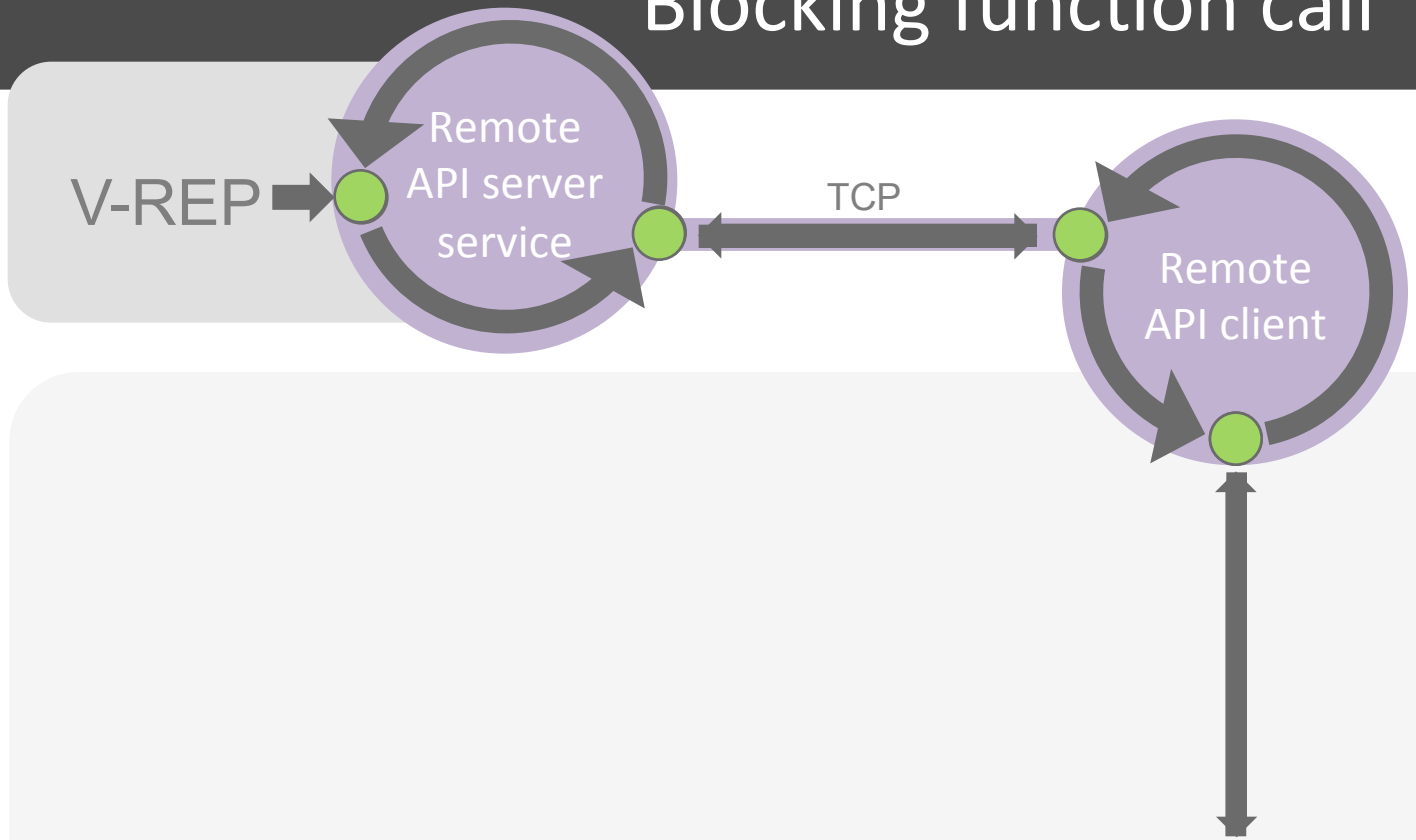
Remote API


Remote API server service
Remote API client
V-REP
Matlab
Remote API server service
Remote API client
simExt_remoteApiStart(portNb)
nApi('remoteApi');
clientID=vrep.simxStart(ip,portNb,..);
Remote API server service
Remote API client
Remote API server service
Remote API server service
Urbi
C/C++
Python
Remote API client
Remote API client
Octave
Java

Remote API

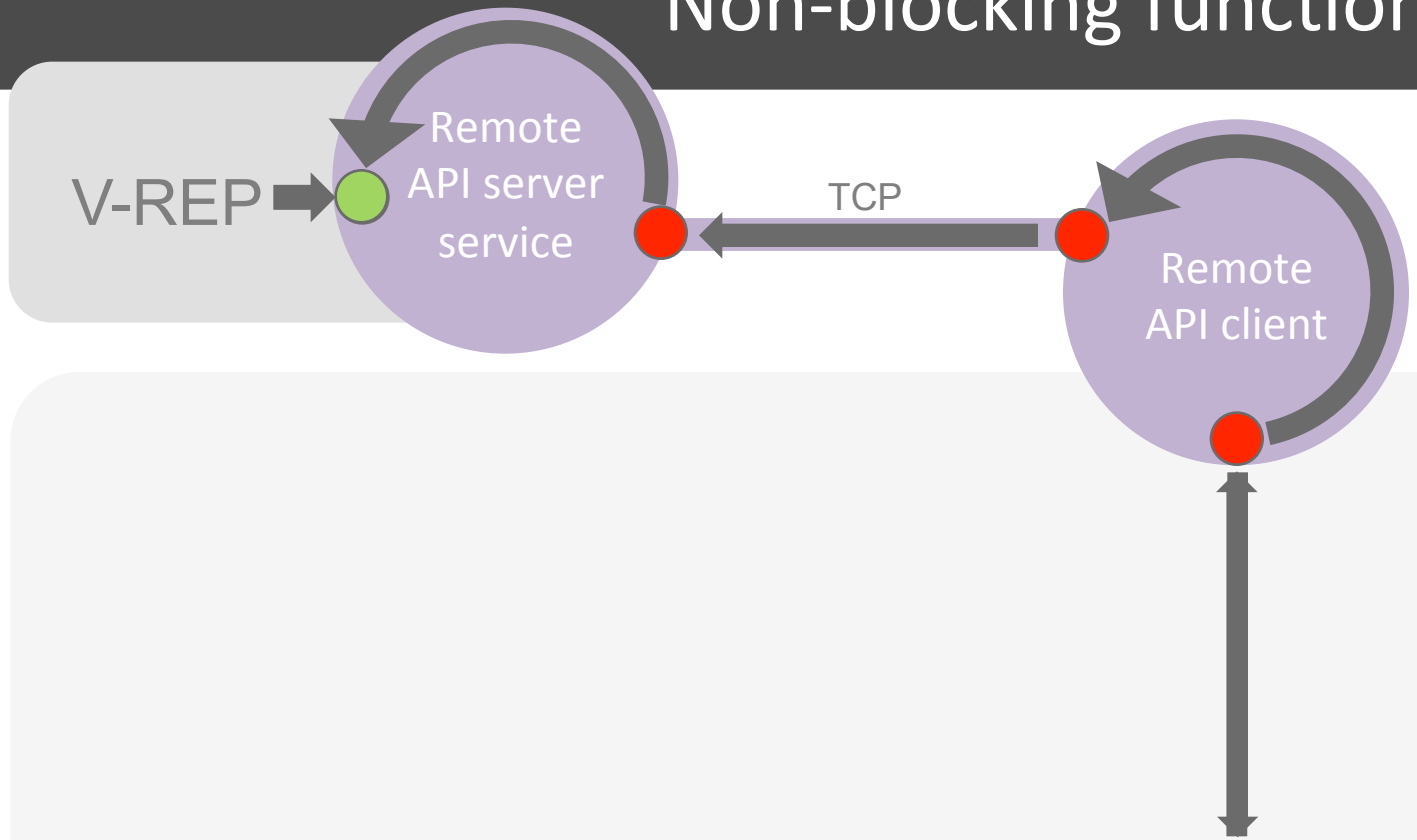
On client side:

Blocking function call



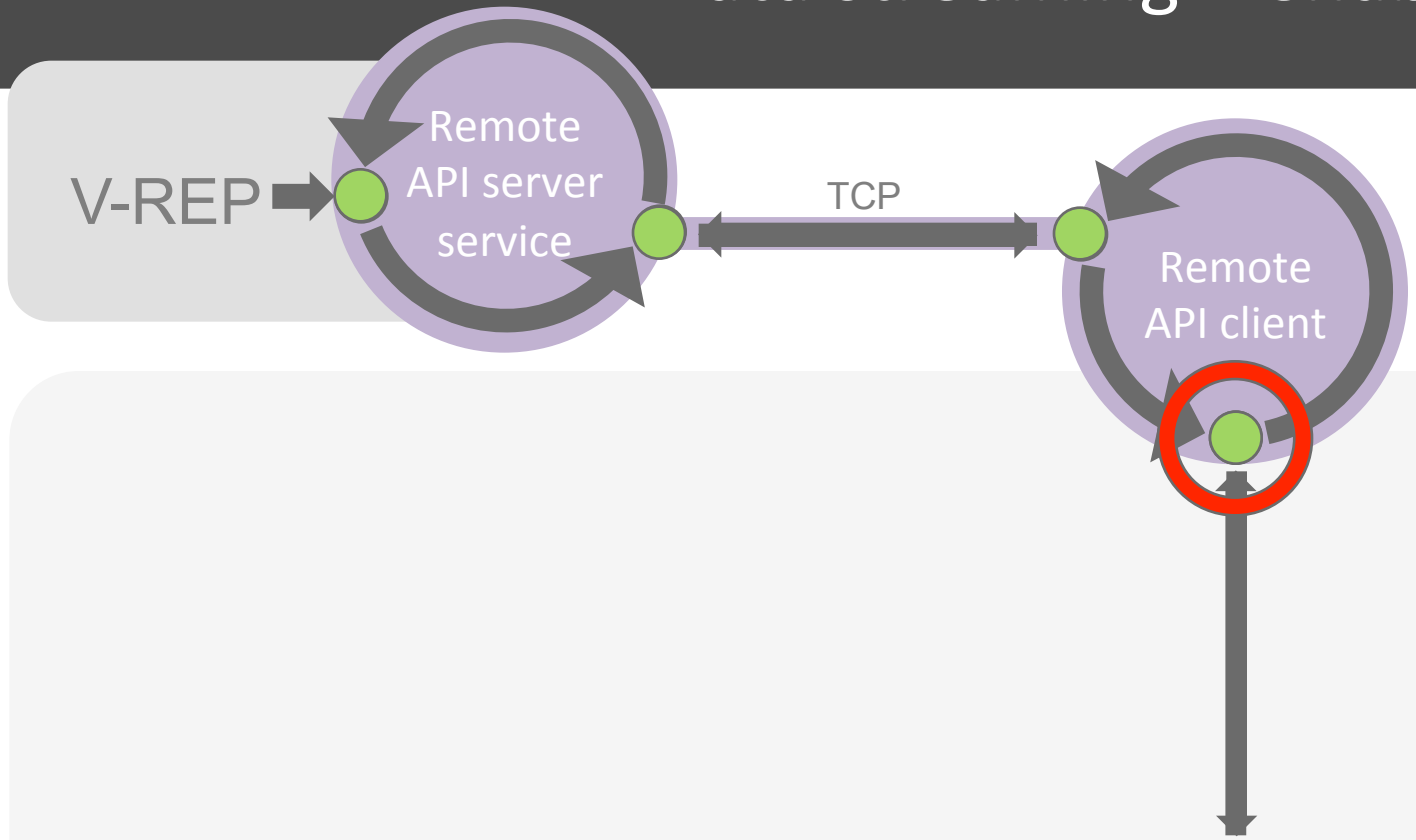

```
[result,position]=vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_oneshot_wait);
```

Non-blocking function call



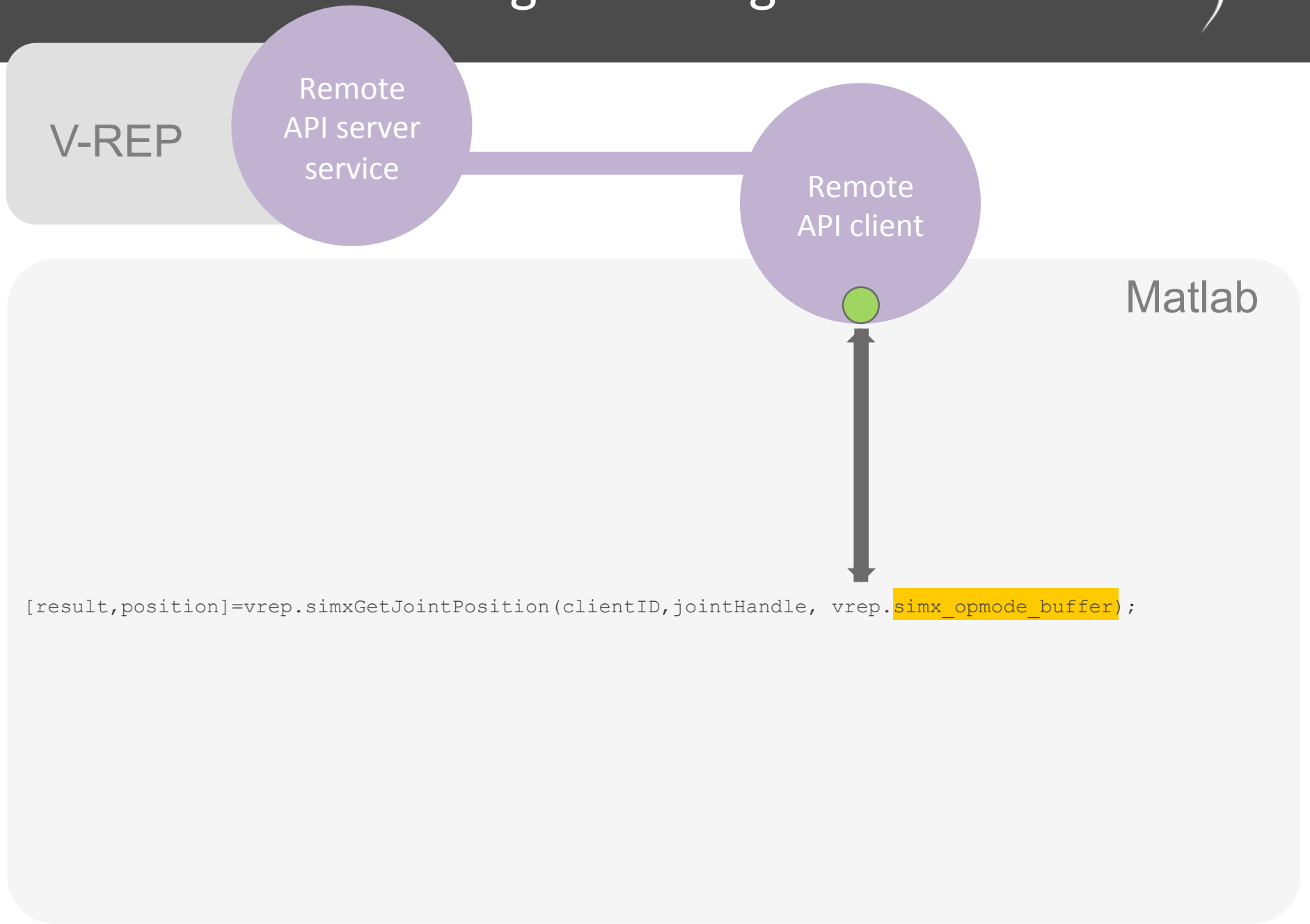

```
vrep.simxSetJointPosition(clientID,jointHandle,jointPosition, vrep.simx_opmode_oneshot);
```

Data streaming – enabling it

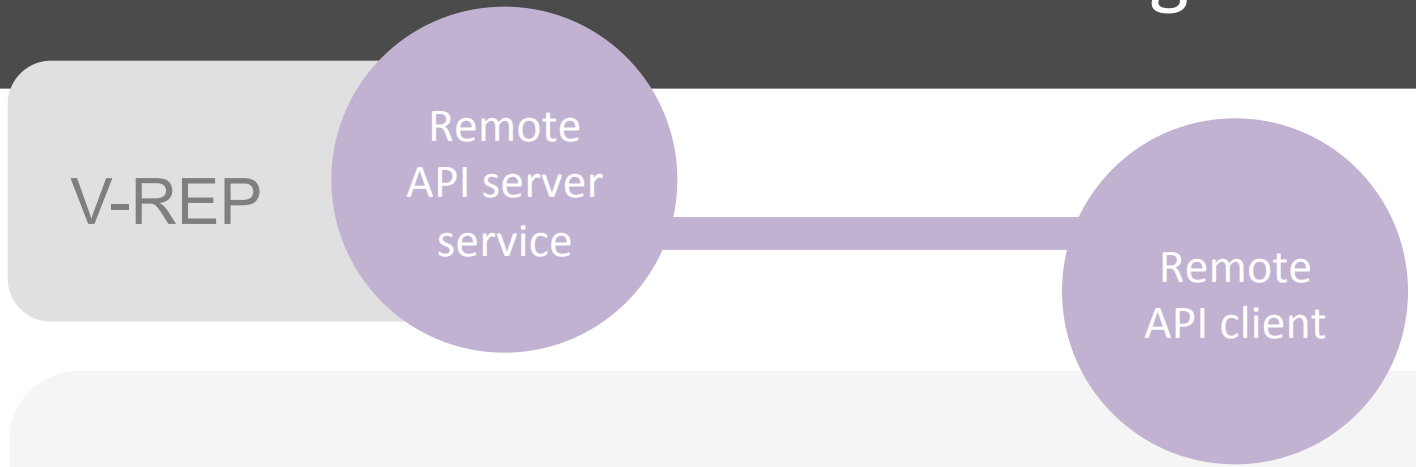



```
vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_streaming);
```

Data streaming – reading streamed data



Data streaming – overview



Matlab

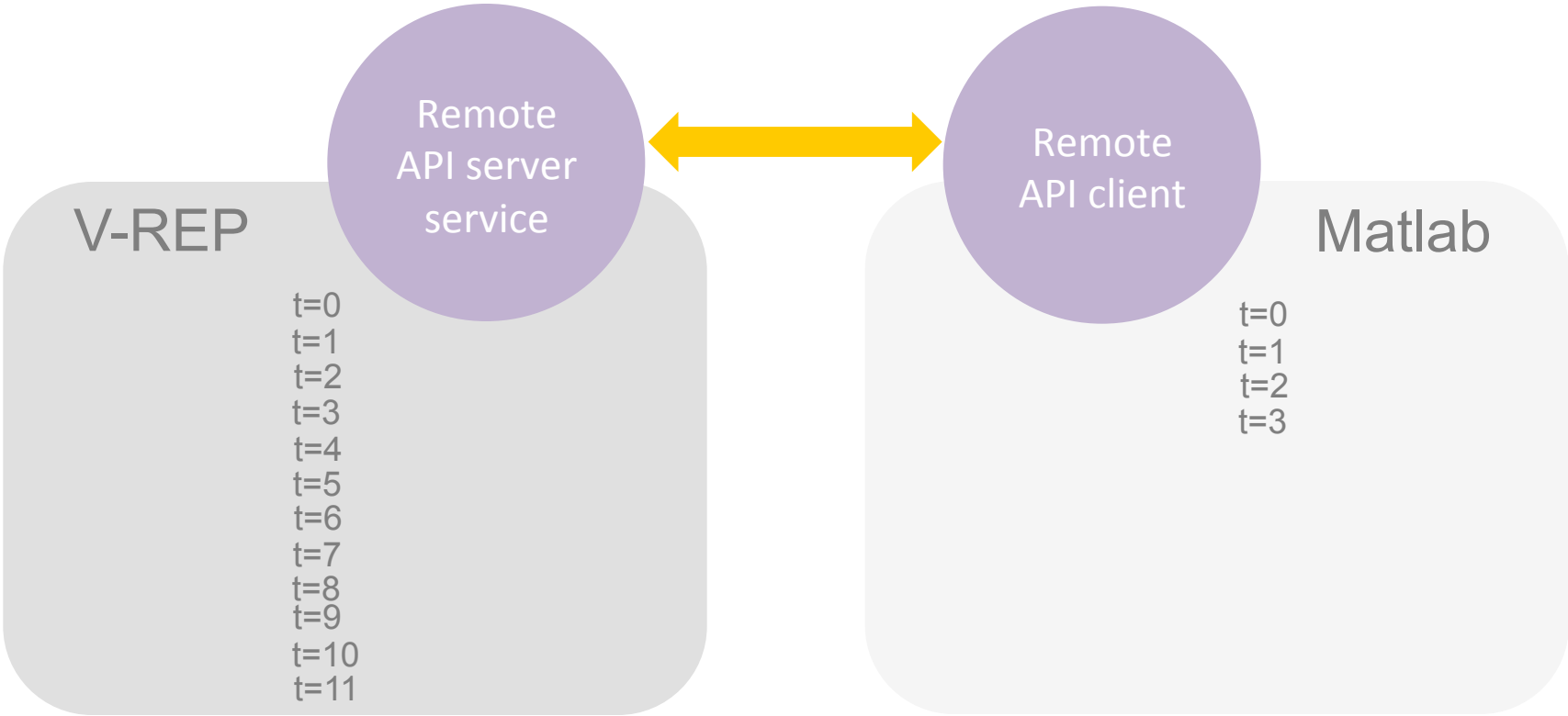
WRONG:

```
vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_streaming);
[result,position]=vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_buffer);
```

CORRECT:

```
vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_streaming);
while (vrep.simxGetConnectionId(clientID)~=-1)
    [result,position]=vrep.simxGetJointPosition(clientID,jointHandle, vrep.simx_opmode_buffer);
    if (result==vrep.simx_return_ok)
        % position is valid here!
    end
end
```

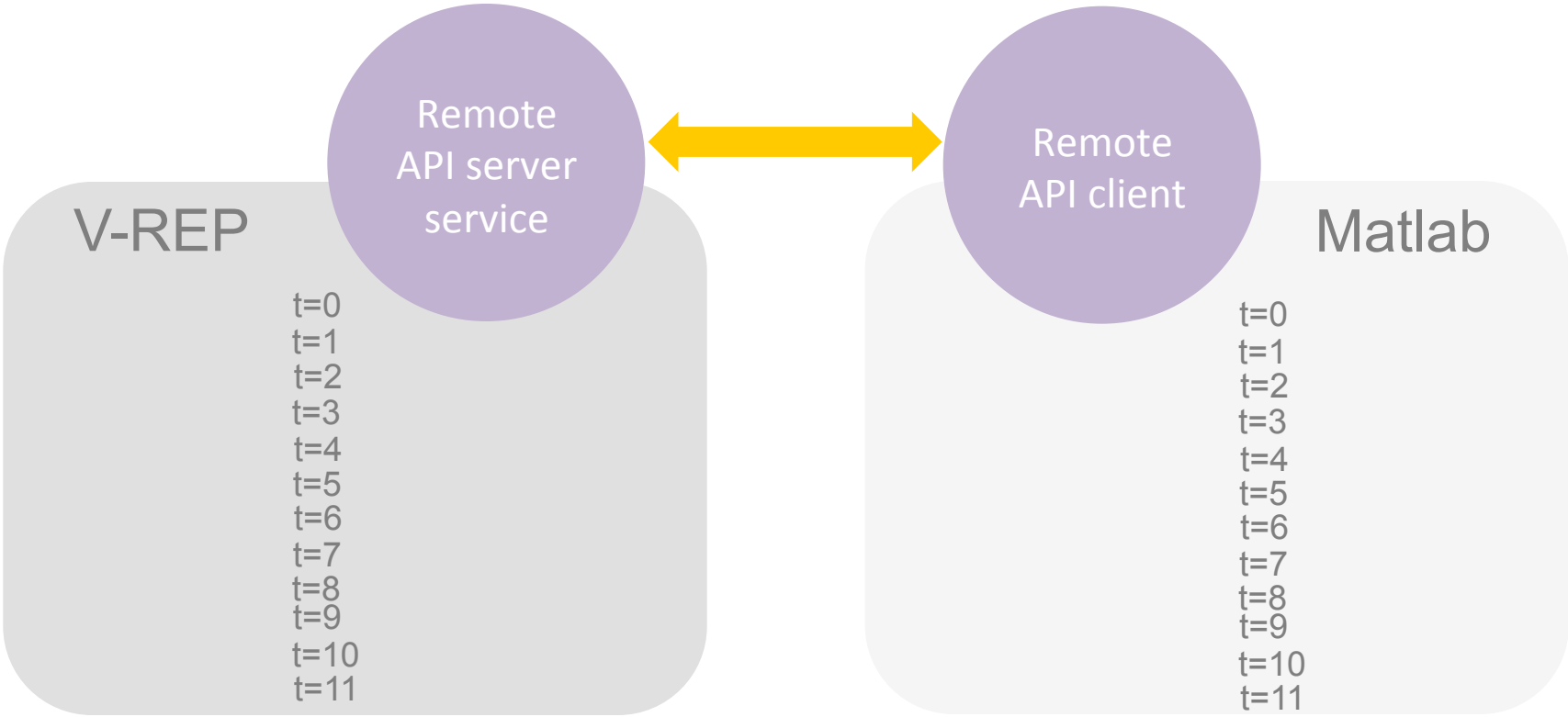
Remote API – real-time vs non real-time



Non real-time

Real-time
(and asynchronous)

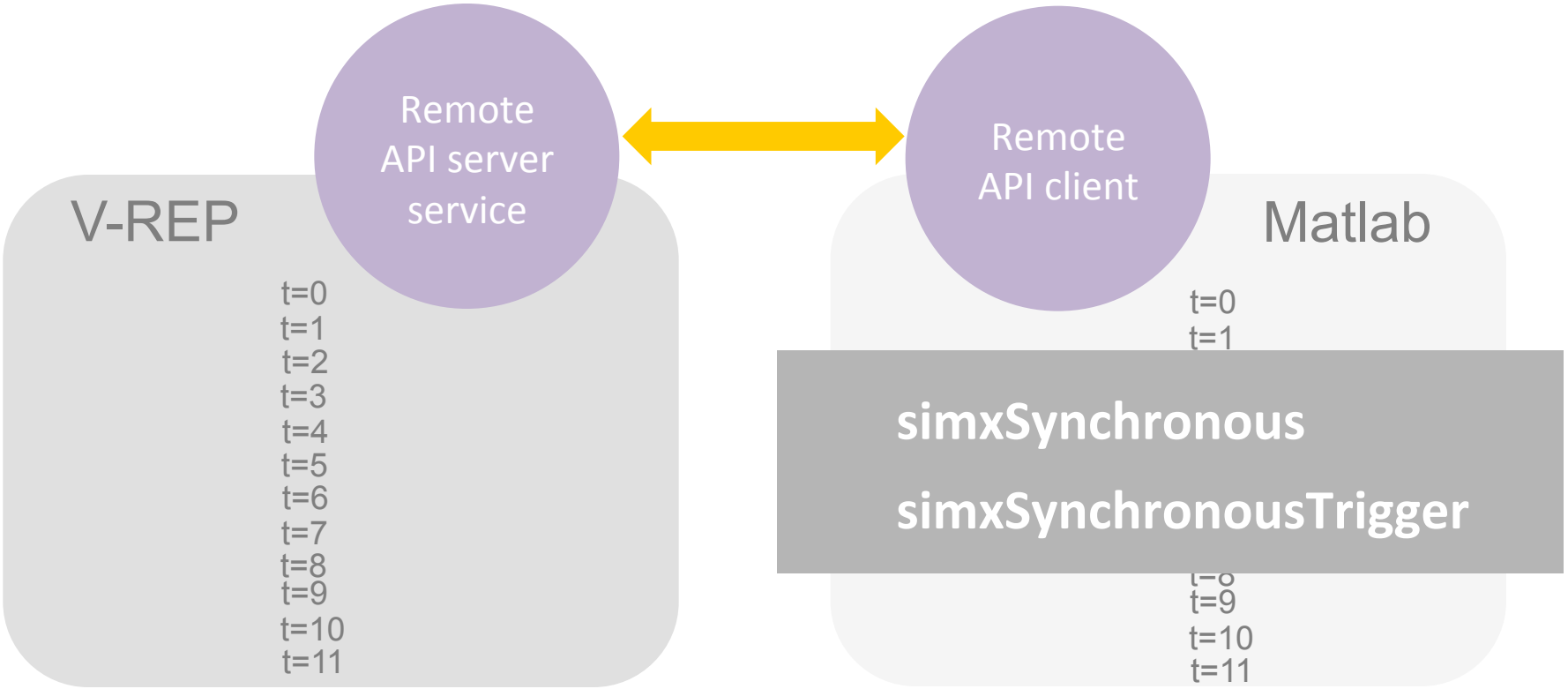
Remote API – real-time vs real-time



Real-time

Real-time
(and asynchronous)

Remote API - synchronized operation



Non real-time

Non real-time
(and synchronous)

Matlab Remote API – what is needed

- remApi.m

- remoteApiProto.m

- remoteApi.dll (or remoteApi.dylib, or remoteApi.so)

Questions?