소프트웨어연구부문
Software
Research Laboratory

SUMO's Dynamic Interface
- TraCI and Libsumo -

이명철
(스마트데이터연구실)

ETRI 한국전자통신연구원
www.etri.re.kr

2020.07.09(목)

목차

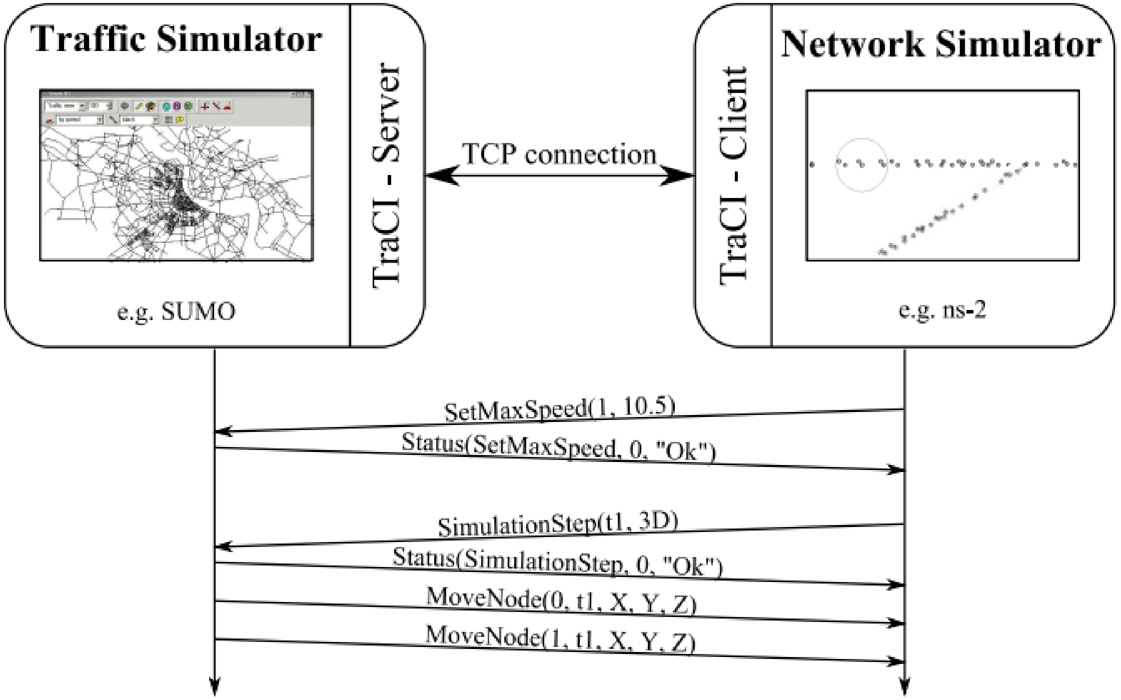
- SUMO’s Dynamic Interface
  - TraCI
  - Libsumo
- SALT’s Dynamic Interface
  - Considerations

TraCI

- TraCI (Traffic Control Interface)
  - SUMO 시뮬레이션 실행 중, 시뮬레이터 데이터를 수집 및 변경할 수 있는 인터페이스
    - 기존: 정적 입력 파일로 시뮬레이터에 제공
  - 교통 환경 변화에 따라 교통 흐름에 끼치는 효과를 더 잘 이해 가능
    - 예) Adaptive traffic light system
  - 대상 시뮬레이션 데이터
    - Edge, lane, vehicle, vehicletype, route, person, junction,
    - Induction loop detector, lane area detector, multi-entry-exit detector, calibrator
    - Trafficlight
    - Poi, polygon
    - Gui, simulation
    - Busstop, chargingstation, overheadwire, parkingarea

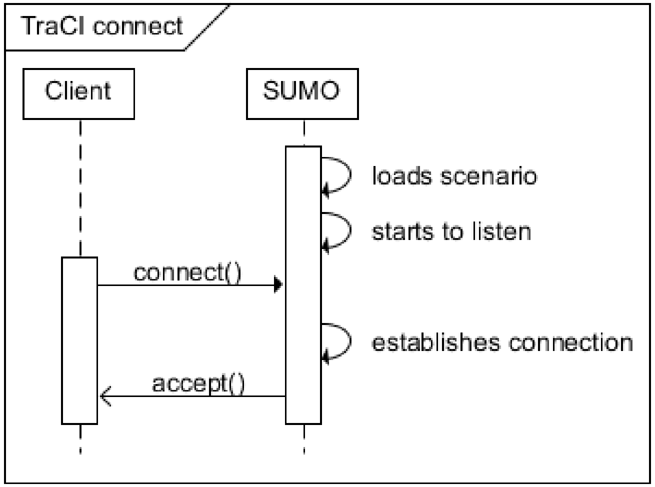
TraCI

- Uses a TCP-based client/server architecture
  - SUMO acts as a server
  - External script (the “controller”) is the client
    - E.g. a Python-Script
    - Receives information about the simulation state from the server and then sends instructions back

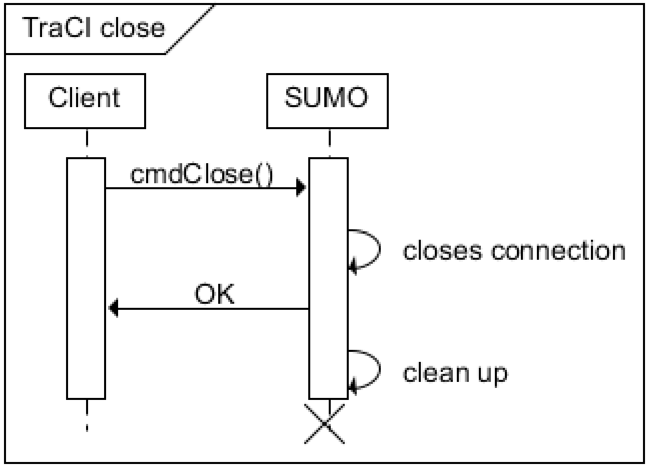


Using TraCI

- Start up SUMO as TraCI server
  - With cmd-line option: --remote-port <INT>
    - <INT>: the port SUMO will listen on for incoming connections
  - SUMO only prepares the simulation and waits for all external applications to connect and take over the control
  - When using SUMO-GUI as a server
    - Simulation must be started before TraCI commands are processed
      - by using the *play* button
      - or by setting the option **--start**



- Shutdown SUMO
  - Client issues “close” command to SUMO
  - SUMO detects whether all route files have been exhausted and all vehicles have left the simulation



Protocol Spec.

- TCP connection is used between SUMO and TraCI client for the exchange of commands/data
- TCP message format
  - Container for a set of commands, responses, data types

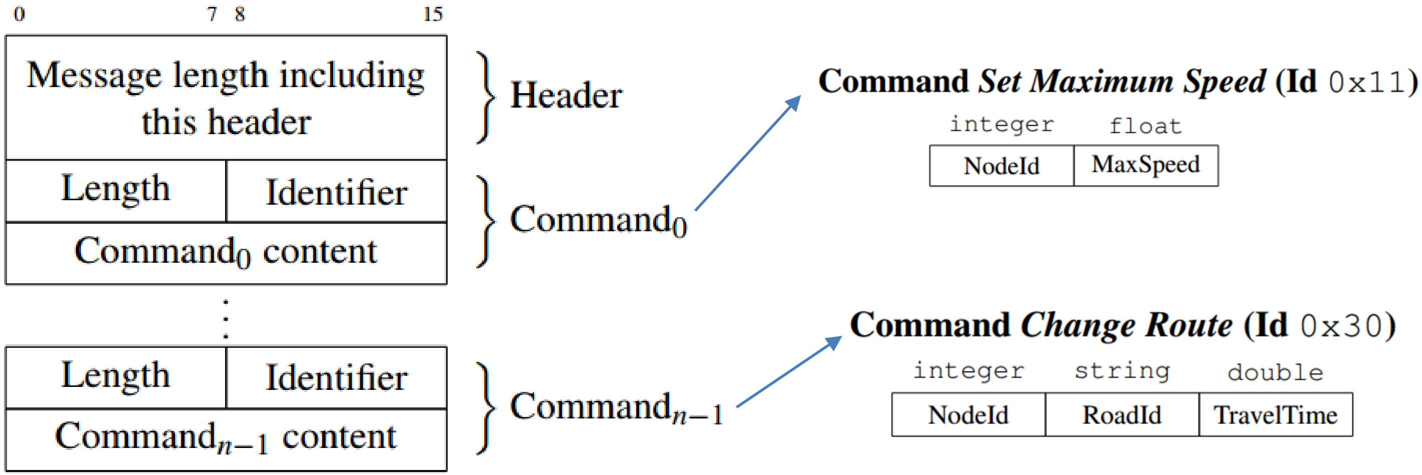

0
7 8
15
Message length including this header
Header
Length
Identifier
Command_0 content
Command_0
⋮
Length
Identifier
Command_{n-1} content
Command_{n-1}
Command *Set Maximum Speed* (Id 0x11)
integer
float
NodeId
MaxSpeed
Command *Change Route* (Id 0x30)
integer
string
double
NodeId
RoadId
TravelTime

Figure 2. TraCI message format: TraCI messages are composed of a small header and a variable number of commands.

Commands

- Value Retrieval
  - Induction Loop Value Retrieval
  - Lane Area Detector Value Retrieval
  - Multi-Entry-Exit Detectors Value Retrieval : multi-entry/multi-exit detectors
  - Traffic Lights Value Retrieval
  - Lane Value Retrieval
  - Vehicle Value Retrieval
  - Person Value Retrieval
  - Vehicle Type Value Retrieval
  - Route Value Retrieval
  - PoI Value Retrieval : points-of-interest
  - Polygon Value Retrieval
  - Junction Value Retrieval
  - Edge Value Retrieval
  - Simulation Value Retrieval
  - GUI Value Retrieval : simulation visualization

- State Changing
  - Change Lane State
  - Change Traffic Lights State
  - Change Vehicle State
  - Change Person State
  - Change Vehicle Type State
  - Change Route State
  - Change PoI State change a point-of-interest's state (or add/remove one)
  - Change Polygon State change a polygon's state (or add/remove one)
  - Change Edge State
  - Change Simulation State
  - Change GUI State : change simulation visualization

- Control-related Commands
  - Perform a simulation step, close the connection, reload the simulation

- Subscriptions
  - TraCI/Object Variable Subscription
  - TraCI/Object Context Subscription

TraCI - Language Bindings

- TraCI APIs
  - Python (Tested daily and supports all TraCI commands)
    - tools/traci/*.py
  - C++: TraCIAPI (API coverage is almost complete)
    - src/utils/traci/TraCIAPI.h
    - src/utils/traci/TraCIAPI.cpp
  - Java: TraaS (API coverage is almost complete)
    - tools/contributed/traas
  - .NET: TraCI.NET (almost complete API coverage)
  - Matlab: TraCI4Matlab (Not all TraCI commands implemented)
    - *tools/contributed/traci4matlab*
  - SOAP: TraaS Webservice (API lags behind the Python client)
    - /tools/contributed/traas/src/main/java/de/tudresden/ws/
- TraCI Server
  - src/traci-server
    - .cpp, .h

Main APIs

- 시뮬레이션을 코드에서 스텝 별로 진행
  - traci.simulationStep()

- 교통 신호 정보 검색 및 변경
  - traci.trafficlights.setPhase(tlsID, index)
    - setPhase(string, integer) -> None
    - Switches to the phase with the given index in the list of all phases for the current program
  - traci.trafficlight.getPhase(tlsID)
    - getPhase(string) -> integer
    - Returns the index of the current phase within the list of all phases of the current program

Main APIs

- 도로 정보 검색 및 변경
  - 현재 특정 도로 위에 있는 차량들의 ID 검색
  - traci.edge.getLastStepVehicleIDs(edgeID)
    - getLastStepVehicleIDs(string) -> list(string)
    - Returns the ids of the vehicles for the last time step on the given edge.
  - traci.lane.getLastStepVehicleIDs(laneID)
    - getLastStepVehicleIDs(string) -> list(string)
    - Returns the ids of the vehicles for the last time step on the given lane.
- 교차로 정보 검색
  - 특정 교차로의 x, y 좌표 검색
  - traci.junction.getPosition(junctionID)
    - getPosition(string) -> (double, double)
    - Returns the coordinates of the center of the junction.

Main APIs

- 차량 정보 검색
  - 현재 특정 차량의 위치, 레인 인덱스, 속도, 대기시간 검색
  - traci.vehicle.getPosition(v)
    - getPosition(string) -> (double, double)
    - Returns the position of the named vehicle within the last step
  - traci.vehicle.getLaneIndex(v)
    - getLaneID(string) -> string
    - Returns the id of the lane the named vehicle was at within the last step
  - traci.vehicle.getSpeed(v)
    - getSpeed(string) -> double
    - Returns the speed in **m/s** of the named vehicle within the last step
  - traci.vehicle.getWaitingTime(v)
    - getWaitingTime() -> double
    - Return the waiting time of a vehicle, which is defined as the time (in seconds) spent with a speed below **0.1m/s**

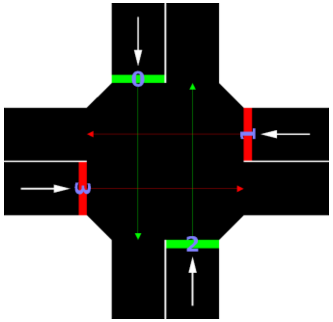
Using TraCI

- Example

```
sumoBinary = "/path/to/sumo-gui“
sumoCmd = [sumoBinary, "-c", "yourConfiguration.sumocfg"]

import traci
traci.start(sumoCmd)

step = 0
while step < 1000:
   traci.simulationStep()
   if traci.inductionloop.getLastStepVehicleNumber("0") > 0:
      traci.trafficlight.setRedYellowGreenState("0", "GrGr")
   step += 1

traci.close()
```

검지기 ID

신호등 ID



leftmost letter "G" encodes the green light for link 0, followed by red for link 1, green for link 2 and red for link 3.

Performance

- TraCI slows down the simulation speed
  - number of TraCI function calls per simulation step
  - types of TraCI functions being called
  - computation within the TraCI script
  - client language
- Bologna scenario (9000 vehicles, 5000 simulation steps)
  - without TraCI : 8s
  - plain position retrieval : 90s
  - retrieval using subscriptions : 42s
  - retrieval using embedded python : 46s
  - retrieval using subscriptions and embedded python : 34s
- C++ client is faster
  - plain position retrieval : 80s
  - retrieval using subscriptions : 28s

Replaced with Libsumo

```
while traci.simulation.getMinExpectedNumber() > 0:
  for veh_id in traci.vehicle.getIDList():
    position = traci.vehicle.getPosition(veh_id)
    traci.simulationStep()
```

plain position retrieval

```
while traci.simulation.getMinExpectedNumber() > 0:
  for veh_id in traci.simulation.getDepartedIDList():
    traci.vehicle.subscribe(veh_id, [traci.constants.VAR_POSITION])
    positions = traci.vehicle.getAllSubscriptionResults()
    traci.simulationStep()
```

retrieval using subscriptions

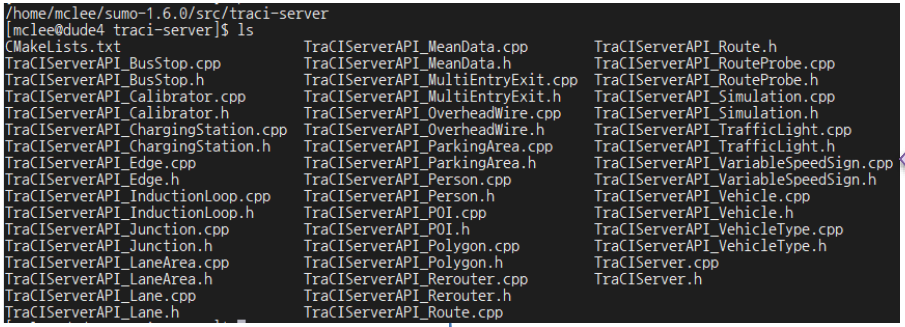
Libsumo

- TraCi
  - Main way to interact with a running simulation
  - Gives the complete flexibility of doing cross-platform, cross-language, and networked interaction with SUMO acting as a server
  - One major drawback
    - Communication overhead due to the protocol and the socket communication
- Libsumo
  - Used to embed SUMO as a library into client process
  - Avoids overhead of socket communication
  - C++, Java and Python bindings are included
    - Implemented C++ interface based on static functions and a few simple wrapper classes for results
    - Function signatures similar to TraCI
    - Pre-built language bindings for Java and Python (using SWIG)
  - Support for other programming languages via SWIG

TraCI vs. Libsumo

- TraCI

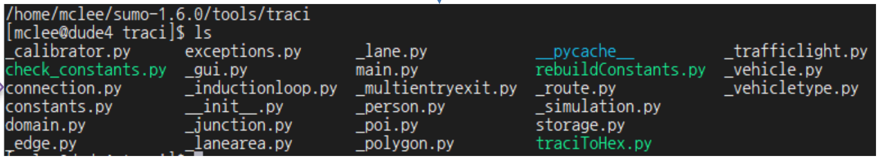
<TraCI Server in C++>



```
/home/mclee/sumo-1.6.0/src/traci-server
[mclee@dude4 traci-server]$ ls
CMakeLists.txt                      TraCIServerAPI_MeanData.cpp        TraCIServerAPI_Route.h
TraCIServerAPI_BusStop.cpp          TraCIServerAPI_MeanData.h          TraCIServerAPI_RouteProbe.cpp
TraCIServerAPI_BusStop.h            TraCIServerAPI_MultiEntryExit.cpp  TraCIServerAPI_RouteProbe.h
TraCIServerAPI_Calibrator.cpp       TraCIServerAPI_MultiEntryExit.h    TraCIServerAPI_Simulation.cpp
TraCIServerAPI_Calibrator.h         TraCIServerAPI_OverheadWire.cpp    TraCIServerAPI_Simulation.h
TraCIServerAPI_ChargingStation.cpp  TraCIServerAPI_OverheadWire.h      TraCIServerAPI_TrafficLight.cpp
TraCIServerAPI_ChargingStation.h    TraCIServerAPI_ParkingArea.cpp     TraCIServerAPI_TrafficLight.h
TraCIServerAPI_Edge.cpp             TraCIServerAPI_ParkingArea.h       TraCIServerAPI_VariableSpeedSign.cpp
TraCIServerAPI_Edge.h               TraCIServerAPI_Person.cpp          TraCIServerAPI_VariableSpeedSign.h
TraCIServerAPI_InductionLoop.cpp    TraCIServerAPI_Person.h            TraCIServerAPI_Vehicle.cpp
TraCIServerAPI_InductionLoop.h      TraCIServerAPI_POI.cpp             TraCIServerAPI_Vehicle.h
TraCIServerAPI_Junction.cpp         TraCIServerAPI_POI.h               TraCIServerAPI_VehicleType.cpp
TraCIServerAPI_Junction.h           TraCIServerAPI_Polygon.cpp         TraCIServerAPI_VehicleType.h
TraCIServerAPI_LaneArea.cpp         TraCIServerAPI_Polygon.h           TraCIServer.cpp
TraCIServerAPI_LaneArea.h           TraCIServerAPI_Rerouter.cpp        TraCIServer.h
TraCIServerAPI_Lane.cpp             TraCIServerAPI_Rerouter.h
TraCIServerAPI_Lane.h               TraCIServerAPI_Route.cpp
```

TCP

Application



```
/home/mclee/sumo-1.6.0/tools/traci
[mclee@dude4 traci]$ ls
_calibrator.py      exceptions.py       _lane.py             __pycache__          _trafficlight.py
check_constants.py  _gui.py             main.py              rebuildConstants.py  _vehicle.py
connection.py       _inductionloop.py   _multientryexit.py   _route.py            _vehicletype.py
constants.py        __init__.py         _person.py           _simulation.py
domain.py           _junction.py        _poi.py              storage.py
_edge.py            _lanearea.py        _polygon.py          traciToHex.py
```

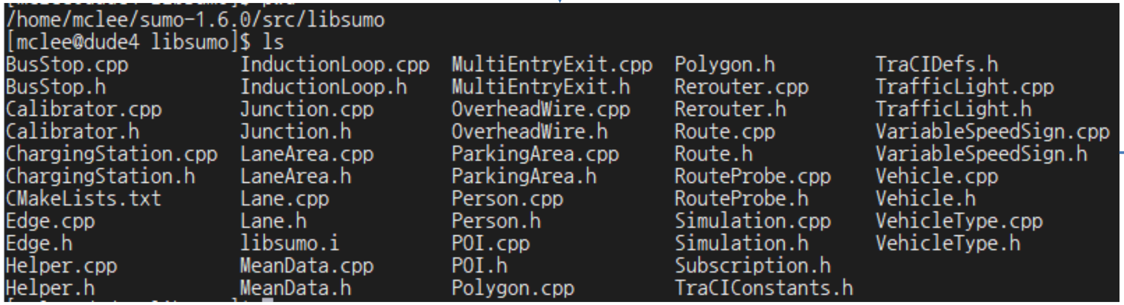
<TraCI Client Library in Python>

SUMO

Embedded in Server

- Libsumo

Application



```
/home/mclee/sumo-1.6.0/src/libsumo
[mclee@dude4 libsumo]$ ls
BusStop.cpp          InductionLoop.cpp  MultiEntryExit.cpp  Polygon.h         TraCIDefs.h
BusStop.h            InductionLoop.h    MultiEntryExit.h    Rerouter.cpp      TrafficLight.cpp
Calibrator.cpp       Junction.cpp       OverheadWire.cpp    Rerouter.h        TrafficLight.h
Calibrator.h         Junction.h         OverheadWire.h      Route.cpp         VariableSpeedSign.cpp
ChargingStation.cpp  LaneArea.cpp       ParkingArea.cpp     Route.h           VariableSpeedSign.h
ChargingStation.h    LaneArea.h         ParkingArea.h       RouteProbe.cpp    Vehicle.cpp
CMakeLists.txt       Lane.cpp           Person.cpp          RouteProbe.h      Vehicle.h
Edge.cpp             Lane.h             Person.h            Simulation.cpp    VehicleType.cpp
Edge.h               libsumo.i          POI.cpp             Simulation.h      VehicleType.h
Helper.cpp           MeanData.cpp       POI.h               Subscription.h
Helper.h             MeanData.h         Polygon.cpp         TraCIConstants.h
```

SUMO

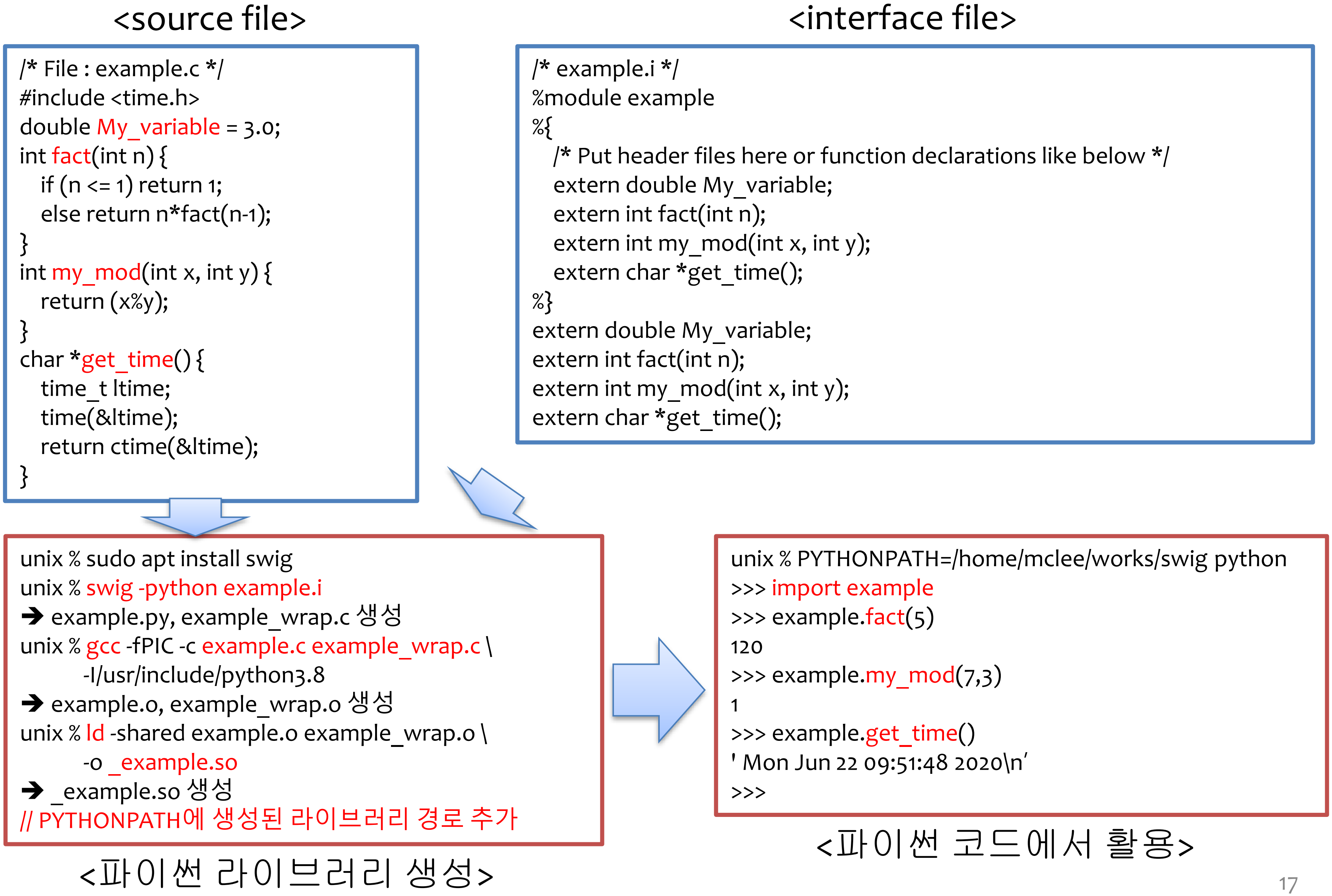
Embedded in Client

<Client Library in C++ → Python Interface>

SWIG

- SWIG (http://www.swig.org/)
  - A software development tool that connects programs written in C and C++ with a variety of high-level programming languages
  - Used with different types of target languages including
    - Common scripting languages such as Javascript, Perl, PHP, Python, Tcl and Ruby
    - Non-scripting languages such as C#, D, Go language, Java including Android, Lua, OCaml, Octave, Scilab and R
    - Several interpreted and compiled Scheme implementations (Guile, MzScheme/Racket)
  - Typically used to parse C/C++ interfaces and generate the 'glue code' required for the above target languages to call into the C/C++ code
  - Free software and the code that SWIG generates is compatible with both commercial and non-commercial projects

SWIG



<source file>

```
/* File : example.c */
#include <time.h>
double My_variable = 3.0;
int fact(int n) {
   if (n <= 1) return 1;
   else return n*fact(n-1);
}
int my_mod(int x, int y) {
   return (x%y);
}
char *get_time() {
   time_t ltime;
   time(&ltime);
   return ctime(&ltime);
}
```

<interface file>

```
/* example.i */
%module example
%{
   /* Put header files here or function declarations like below */
   extern double My_variable;
   extern int fact(int n);
   extern int my_mod(int x, int y);
   extern char *get_time();
%}
extern double My_variable;
extern int fact(int n);
extern int my_mod(int x, int y);
extern char *get_time();
```

```
unix % sudo apt install swig
unix % swig -python example.i
➔ example.py, example_wrap.c 생성
unix % gcc -fPIC -c example.c example_wrap.c \
       -I/usr/include/python3.8
➔ example.o, example_wrap.o 생성
unix % ld -shared example.o example_wrap.o \
       -o _example.so
➔ _example.so 생성
// PYTHONPATH에 생성된 라이브러리 경로 추가
```

<파이썬 라이브러리 생성>

```
unix % PYTHONPATH=/home/mclee/works/swig python
>>> import example
>>> example.fact(5)
120
>>> example.my_mod(7,3)
1
>>> example.get_time()
' Mon Jun 22 09:51:48 2020\n'
>>>
```

<파이썬 코드에서 활용>

Libsumo

- Using Libsumo

```
import libsumo
libsumo.start(["sumo", "-c", "test.sumocfg"])
libsumo.simulationStep()
libsumo.xxx() // Use TraCI APIs
```

```
import libsumo as libsumo
import os
import traci.constants as tc

def test(sumo_cfg_file):
    sumo_home = os.environ['SUMO_HOME ' ]
    sumo_bin = sumo_home + r'/bin/sumo '
    sumo_cmd_libsumo = [sumo_bin, '-c', sumo_cfg_file, '--
step-length', '1’, ‘--no-step-log', "True"]

    libsumo.start(sumo_cmd_libsumo)
    set_subscribe()

    sim_time =0
    while sim_time <=5400:
        libsumo.simulationStep()
        sim_time += 1
        libsumo.close()
    print("simulation end!!!")
```

```
def set_subscribe():
    inductionloop_det = libsumo.inductionloop.getIDList()
    print("inductionloop id list:\n", inductionloop_det)

    lanearea_det = libsumo.lanearea.getIDList()
    for e1det in inductionloop_det:
        libsumo.inductionloop.subscribe(e1det, (tc.LAST_STEP_VEHICLE_NUMBER,))
        for e2det in lanearea_det:
            libsumo.lanearea.subscribe(e2det, (tc.JAM_LENGTH_METERS,
tc.JAM_LENGTH_VEHICLE,))
            if __name__=="__main__":
                for i in range(1000):
                    if i ==0:
                        sumo_cfg_file_path = r"C:\test1.sumocfg"
                        print("test1.sumocfg")
                    else:
                        sumo_cfg_file_path = r"C:\test2.sumocfg"
                        print("test2.sumocfg")
        test(sumo_cfg_file_path)
```

Libsumo

- Limitations (do not work (or work differently))
  - Cannot run with SUMO-GUI
  - Subscriptions that require additional arguments (except for vehicle.getLeader)
  - Stricter type checking
    - TraCI client sometimes accepts any iterable object
      - where Libsumo wants a list
    - TraCI client may accept any object
      - where Libsumo needs a boolean value
  - Using traci.init or traci.connect is not possible
    - Always need to use libsumo.start
  - TraCI generates every TraCIException message on stderr
    - Libsumo does not generate this message
  - stepListener interface is not supported

SALT의 동적 인터페이스 - 고려 사항

- TraCI는 방대한 시뮬레이션 데이터에 대한 API 를 제공
  - 도로(Edge, Lane), 교차로(Junction), 신호(Traffic Lights), 차량(Vehicle Type, Vehicle), 검지기(Induction Loop, Lane Area Detector, Multi-Entry-Exit Detectors), 경로(Route), 시설물(PoI), 사람(Person), Polygon 정보 검색 및 상태 변경
  - 시뮬레이션(Simulation) 및 GUI (GUI) 정보 검색 및 상태 변경
  - 시뮬레이션 제어(Control)
  - 일반 파라미터(Generic Parameters)
  - 가입(Object Context, Object Variable)

- 모든 API를 다 제공할 필요가 있는가?
  - SALT 는 제공할 수 없는 API가 존재
  - SALT 에서 제공 가능한 API만 일단 제공

- 어떤 API를 신호 최적화에서 추가로 필요로 할 것인가?
  - 신호 최적화에서 제안이 필요

SALT의 동적 인터페이스 - 고려 사항

- Client/Server 구조 or Embedded Library 구조?
  - Client/Server 구조
    - Completeness → SUMO 기준일 뿐
    - Flexibility → SALT를 원격 제어하면, 좀 더 다양한 환경에서 사용 가능
  - Embedded Library 구조
    - No communication overhead
    - Non-UI simulation만 가능?
      - Embedded 이면서 GUI를 띄울 수 있을 것 같다
      - GUI는 새로 만들어야 하나?
        - » 아니면 Web-based GUI or Windows-based GUI를 활용할 수 있나?

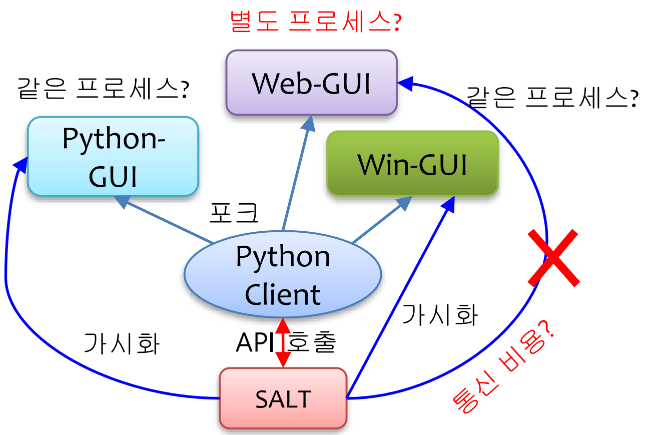
- GUI or Non-UI simulation?
  - GUI: 동적 제어 결과 바로 눈으로 확인 가능
    - Web-based GUI 보유
    - Windows-based GUI 보유 (SALT-Visualizer)
    - Python GUI : 새로 개발 필요?
  - Non-UI: 동적 제어 결과를 시뮬레이션이 끝나고 나중에 확인해야 함
    - 기존 Web/Windows-based GUI 활용 가능

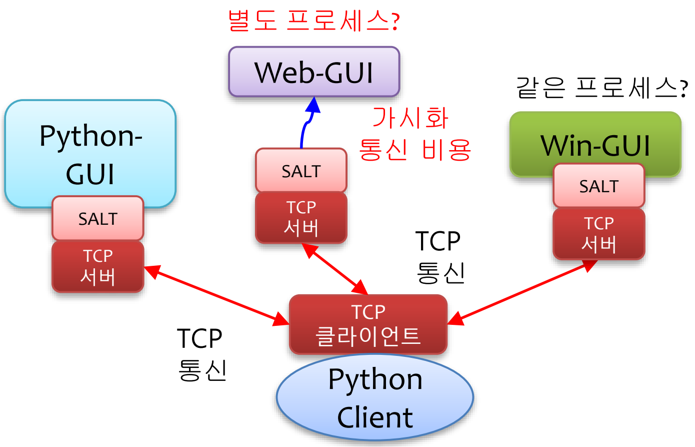
GUI-based Dynamic Simulation

구분	장단점	Web-GUI	Win-GUI	Python-GUI (PyQT)
(2) Python Client에 SALT 임베딩	장점	• R/Python 분석 + Jupyter Notebook 처럼 시뮬레이션 • 클라우드 서버에서 활용 가능	• Python에서 GUI 포크 • SALT-Visualizer 활용 (sumo-gui 유사) • SALT와 GUI가 동일 프로세스 • 가시화 데이터 API 호출로 전달	• R/Python 분석처럼 시뮬레이션 • 클라우드 서버에서 활용 가능 • SALT와 GUI가 동일 프로세스 • 가시화 데이터 API 호출로 전달
	단점	• SALT와 GUI가 별도 프로세스? → SALT/GUI 간에 가시화 데이터 통신 비용 발생 • 기존 GUI 코드 수정 필요	• 기존 GUI 코드 수정 필요 • 클라우드 서버에서 활용 불가	• 새로 GUI 개발 필요 • 클라우드 서버 구동시 원격 디스플레이 활용 필요
Client Server 구조 (3)	장점	• GUI는 지금처럼 별도의 원격 웹 서버 앱으로 동작 • 기존 GUI 코드 활용 가능 • 클라우드 서버에서 활용 가능	• SALT-Visualizer 활용 (sumo-gui 유사) • SALT는 GUI와 동일 프로세스 • SALT/GUI 간에 가시화 데이터 API 호출로 전달 • 기존 GUI 코드 활용 가능	• Sumo-gui 처럼 원격 서버로 동작 • SALT는 GUI와 동일 프로세스로 동작 • SALT/GUI 간에 가시화 데이터 API 호출로 전달 • 클라우드 서버에서 활용 가능
	단점	• SALT는 GUI와 별도의 프로세스? → SALT/GUI 간에 가시화 데이터 통신 비용 발생 • Client/SALT 간에 TCP 통신 비용 발생 • 기존 GUI 코드 수정 필요	• 기존 GUI 코드 수정 필요 • Client/SALT 간에 TCP 통신 비용 • 클라우드 서버에서 활용 불가	• Client/SALT 간에 TCP 통신 비용 • 새로 GUI 개발 필요 • 클라우드 서버 구동시 원격 디스플레이 활용 필요

(1)

(2)





참고

- TraCI: an interface for coupling road traffic and network simulators (CNS, 2008)
  - https://dl.acm.org/doi/10.1145/1400713.1400740
- TraCI
  - https://sumo.dlr.de/docs/TraCI.html
  - https://sumo.dlr.de/docs/TraCI/Interfacing_TraCI_from_Python.html
  - https://sumo.dlr.de/docs/TraCI/C++TraCIAPI.html
- Libsumo
  - https://sumo.dlr.de/docs/Libsumo.html

감사합니다