SALT 신호 변경 테스트

신호 스케줄 변경 테스트 (C++)

SALT 신호 변경 테스트 (Schedule 변경)

- Schedule 변경 테스트
  - 특정 시간, 특정 사거리 신호 Schedule을 다음 Schedule로 변경하는 테스트
    - 평일 오전 6시 - 오전 7시 59분 (3시간) 동안 강동구 시뮬레이션 수행 중, 특정 시간 (=시작 후 30분)에 길동사거리 신호 스케줄을 다음 스케쥴로 변경하는 테스트
    - Class SimulationController 에 다음 public function 추가
      - TrafficSignal* getTrafficSignalByNodeID(const string& _nodeID);
        - » 특정 교차로 (예: 목표 교차로) NodeID 가 주어졌을 때, 해당 TrafficSignal 객체 반환
      - void changeTLSchedule(SALTTime triggertime, const string& _nodeID);
        - » 신호 변경 시간 (e.g. SALT TimeStep), 특정 교차로 (예: 목표 교차로) NodeID 가 주어졌을 때, 해당 시간을 기준으로 TODPlan의 다음 Schedule로 변경
  - 테스트 결과
    - 다음 스케줄이 존재하는 경우, 변경 됨을 확인함.

SALT 신호 변경 테스트 (Schedule 변경)

```cpp
// ------------------------------------------------------------------------
// @ do simulation loop
std::cout << "[Simulation Start]" << std::endl;
auto timeStart = Clock::now();
int NUMBER_PRINT = 20; // simulation progress states will be printed (NUMBER_PRINT) times // 20 -> each 5%

// repeat to call SimulationController->doSimulationStep(currentStep)
while(!SC->checkEnd()) {
    // get current step + update status of vehicles, traffic signal and event
    SALTTime currentStep = SC->getCurrentStep();
    string targetTLNode = "cluster_572701114_572701116_572701117_572712746_572712755";
    TrafficSignal * targetTL = SC->getTrafficSignalByNodeID(targetTLNode);
    SALTTime curNextScheduleTime = targetTL->getNextScheduleTime();
    auto curScheduleID = targetTL->getCurTrafficSignalScheduleID();
    SALTTime targetTriggerTime = 1800;

    if (currentStep % 1200 == 0) {
        cout<<"Next Schedule Time: " << curNextScheduleTime << ", [ScheduleID: " << curScheduleID << "]" << endl;
    }

    if(currentStep == targetTriggerTime){
        cout<<"[Trigger Point]"<< endl;
        cout<<"TL Change Time: " << targetTriggerTime << ", Original Next Schedule Time: " << curNextScheduleTime << "

        SC->changeTLSchedule(targetTriggerTime, targetTLNode);

        auto afterNextTime = targetTL->getNextScheduleTime();
        auto afterScheduleID = targetTL->getCurTrafficSignalScheduleID();
        cout<<"Change Next Schedule Time: " << afterNextTime << ", [ScheduleID: " << afterScheduleID << "]" << endl;
    }
    SC->doSimulationStep(currentStep); // doSimulationStep will increase SC::currentStep

    // (optional) print
    SC->printStep(currentStep,NUMBER_PRINT);

}
```

```
[Simulation Start]
Next Schedule Time: 3600, [ScheduleID: 1]
4% done
9% done
14% done
Next Schedule Time: 3600, [ScheduleID: 1]
19% done
[Trigger Point]
TL Change Time: 1800, Original Next Schedule Time: 3600, [ScheduleID: 1]
Before : 3600, [ScheduleID: 1]
[ScheduleID: 13]
[ScheduleID: 13]
After : 99999999, [ScheduleID: 13]
Change Next Schedule Time: 99999999, [ScheduleID: 13]
29% done
Next Schedule Time: 99999999, [ScheduleID: 13]
34% done
39% done
44% done
49% done
Next Schedule Time: 99999999, [ScheduleID: 13]
54% done
59% done
64% done
Next Schedule Time: 99999999, [ScheduleID: 13]
69% done
74% done
79% done
Next Schedule Time: 99999999, [ScheduleID: 13]
84% done
89% done
94% done
99% done
[Simulation End]
```

SALT 신호 변경 (Schedule) 을 위한 이슈

- 현재 tss.xml의 경우, TODPlan 테이블 기준으로 생성됨.
  - 평일의 경우, tss.xml에는 ScheduleID 1, 2, 3, 5, 6 만 포함됨.
  - 신호최적화 수행 시, 모든 ScheduleID에 대해 변경 수행하려면, 현재 tss.xml에 TIME PLAN1에 포함된 모든 Schedule이 포함되어야 함.

PLAN TABLE

TIME PLAN 1

수준	주기 (초)	ID	연동 (초)	시간분할(초) RING A. B	감응제어 압버튼(×)	검지기(○)	검지기(×)
1	180	1	148	20,82,23,17,28,10 20,82,23,17,28,10			
2	180	2	20	30,72,23,17,28,10 30,72,23,17,28,10			
		3	170	25,77,23,17,28,10 25,77,23,17,28,10			
		4	135	25,77,23,17,28,10 25,77,23,17,28,10			
3	180	5	160	25,77,23,17,28,10 25,77,23,17,28,10			
		6	158	25,77,23,17,28,10 25,77,23,17,28,10			
		7	148	25,77,23,17,28,10 25,77,23,17,28,10			
4		8					
		9					
		10					
5		11					
		12					
		13					
6		14					
		15					
		16					

구분	TOD PLAN 1 시각	수준	ID	TOD PLAN 2 시각	수준	ID	TOD PLAN 3 시각	수준	ID	예약정보(월~일 : 0, 토:7,일:1,월~금:8) 시작	종료	요일/점멸
1	00:00	1	1	00:00	1	1	00:00	1	1			
2	07:00	2	2	07:00	2	2	09:30	2	3			
3	09:30	2	3	09:30	2	3	21:00	3	6			
4	17:00	3	5	17:00	3	5						
5	21:00	3	6	21:00	3	6						
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

주간신호계획

요일	TOD PLAN	TIME PLAN
일	3	1
월	1	1
화	1	1
수	1	1
목	1	1
금	1	1
토	2	1

특수일신호계획

요일	TOD PLAN	TIME PLAN
1.1	3	1
3.1	3	1
5.5	3	1
6.6	3	1
8.15	3	1
10.3	3	1
12.25	3	1

MEMO

LSU사용 방법

LSU사용은 북동남서순으로...
그외에는 LSU5번부터 사용...
버스전용차선5번(북),6번(남)사용

SALT 신호 변경 (Schedule) 을 위한 이슈

- TODPlan에서 고려가능한 Schedule 변경 외 새로운 Schedule 제안 시
  - Schedule 값에 해당하는 id/offset와 phase 정보가 주어져야 함.

```
<schedule offset="182" id="13">
    <phase state="rrrrGGGGGGrrrrGGGGGGr" duration="78"/>
    <phase state="rrrryyyyyyrrrryyyyyyr" duration="4"/>
    <phase state="rrrrrrrrrrGrrrrrrrrrG" duration="14"/>
    <phase state="rrrrrrrrrryrrrrrrrrry" duration="3"/>
    <phase state="GGGGGrrrrrrrrrrrrrrrr" duration="44"/>
    <phase state="yyyyyrrrrrrrrrrrrrrrr" duration="3"/>
    <phase state="rrrrrrrrrrrrrrrrrrrrr" duration="1"/>
    <phase state="rrrrrrrrrrrGGGGrrrrrr" duration="39"/>
    <phase state="rrrrrrrrrrryyygrrrrrr" duration="4"/>
</schedule>
```

신호 PHASE 변경 테스트 (C++)

SALT 신호 Phase 변경

- 신호 Phase 변경
  - [현재 SALT] 시간 기반 신호 운영 수행 → 매 스텝마다 신호 Schedule/Phase 변경할 시간인지 체크하여 변경 시간인 경우 신호 Schedule/Phase를 다음 상태로 변경 업데이트함.
  - [신호 최적화 과정에서] Adaptive 신호 적용을 위해서, 대상 교차로의 신호는 TODPlan에서 벗어나 학습에 따라 Phase 변경/유지할 필요가 있음.

```
<schedule offset="182" id="13">
    <phase state="rrrrGGGGGGrrrrGGGGGGr" duration="78"/>
    <phase state="rrrryyyyyyrrrryyyyyyr" duration="4"/>
    <phase state="rrrrrrrrrrGrrrrrrrrrG" duration="14"/>
    <phase state="rrrrrrrrrryrrrrrrrrry" duration="3"/>
    <phase state="GGGGGrrrrrrrrrrrrrrrr" duration="44"/>
    <phase state="yyyyyrrrrrrrrrrrrrrrr" duration="3"/>
    <phase state="rrrrrrrrrrrrrrrrrrrrr" duration="1"/>
    <phase state="rrrrrrrrrrGGGGrrrrrr" duration="39"/>
    <phase state="rrrrrrrrrrryyyyrrrrrr" duration="4"/>
</schedule>
```

시간 (duration 값) 기반 변경

학습 과정에서 액션 결정 기반 변경

→ **기존의 duration 값 무의미**

→ **예: Action 결정 시기**에 **특정 Schedule**의 **특정 Phase로 변경** 요청 후, **다음 Action 결정 시기까지** 해당 신호 Phase를 **유지**.

신호최적화에서 필요한 기본 기능

- 학습 대상 교차로의 신호 정보 요청
- 특정 Schedule의 특정 Phase로 변경 요청

SALT 변경

- 학습 대상 교차로의 신호를 다른 교차로 신호와 별도 운영할 수 있도록 변경
- SetPhase 기능 추가

SALT 신호 Phase 변경

- SALT 신호 Phase 변경 테스트
  - 특정 시간 마다, 특정 사거리 신호에 포함된 모든 스케줄에 존재하는 Phase를 확인하여, TriggeringTime 마다 순차적으로 변경하는 테스트
    - 평일 오전 6시 - 오전 7시 59분 (3시간) 동안 강동구 시뮬레이션 수행 중, 길동사거리의 신호의 모든 스케줄 정보를 기반으로 Phase 진행 순서를 생성하고, 특정 시간 (= 매 30초) 마다 신호 Phase를 다음 Phase로 변경하는 테스트
    - 학습 모드를 위한 함수를 추가하여 SimulationController 클래스 수정
      - → 추후 동적 인터페이스 제공을 위한 별도 클래스로 구현 예정
    - 주요 추가 함수
      - TrafficSignal* getLearningTrafficSignalByNodeID(const string& _nodeID);
        - » 학습 모드용으로 분리 운영하는 교차로를 지정하고, 노드ID 기반 신호 정보 반환하는 함수
      - void changeTLPhase(SALTTime triggertime, const string& _nodeID, const string& _scheduleID, int _phaseIndex);
        - » 동적으로 신호 Phase를 변경하기 위한 함수
        - » 변수인자 → 변경 적용 시간, 타겟 교차로의 노드ID, 변경할 Phase가 포함된 ScheduleID, 변경할 Phase의 Index 정보
  - 테스트 결과
    - 다음 스케줄이 존재하는 경우, 변경 됨을 확인함.

setPhase(self, tlsID, index)

setPhase(string, integer) -> None

Switches to the phase with the given index in the list of all phases for the current program.

setPhaseDuration(self, tlsID, phaseDuration)

setPhaseDuration(string, double) -> None

Set the remaining phase duration of the current phase in seconds. This value has no effect on subsquent repetitions of this phase.

setPhaseName(self, tlsID, name)

setPhase(string, string) -> None

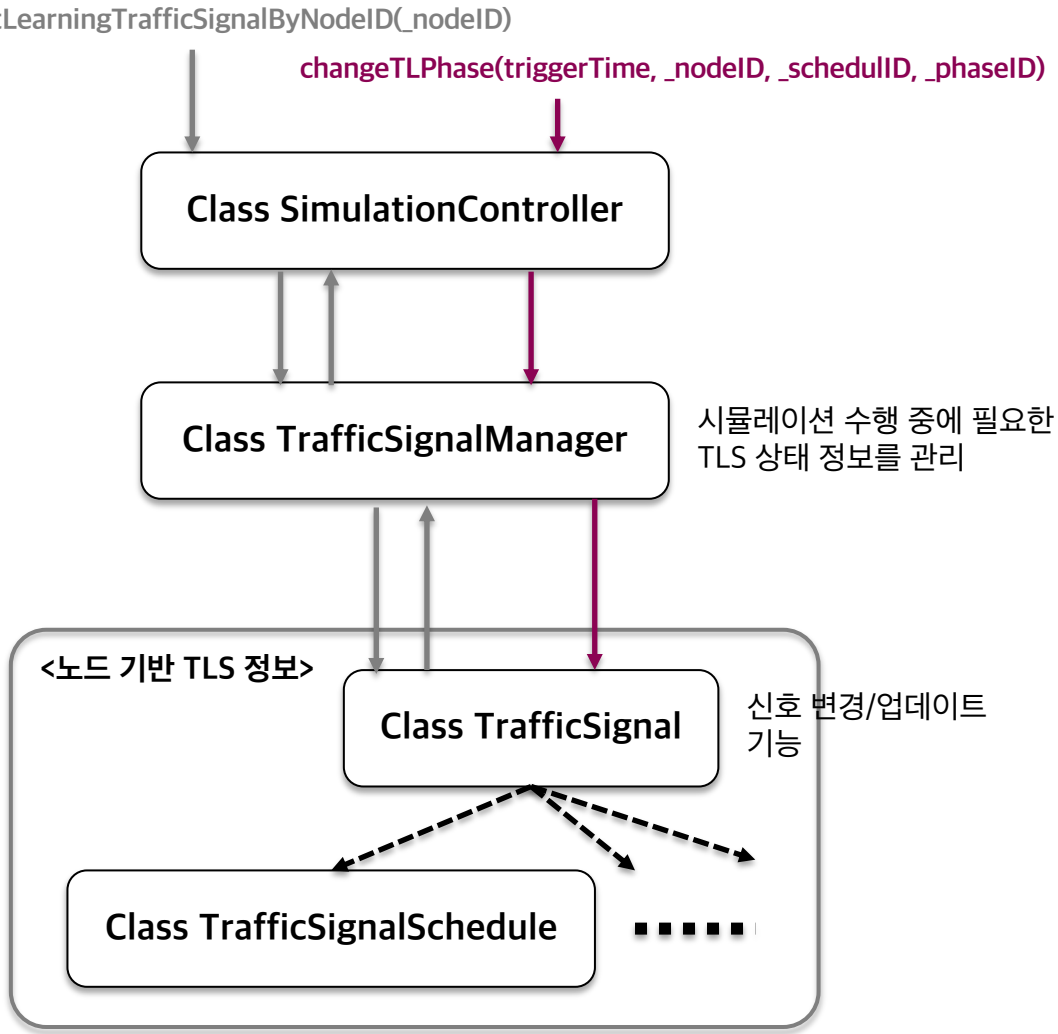
Sets the name of the current phase within the current program

SALT 신호 Phase 변경

- SALT 시뮬레이터 수행을 위한 시나리오, 입력데이터
- 신호 변경 대상 교차로 Node ID

신호 Phase 변경 테스트 (main)

1. 대상 교차로의 신호 정보 획득
2. 해당 교차로 신호 스케줄을 이용하여 Phase 변경 순서 결정
3. SALT 수행 루프 수행 중에 30초 간격 마다 Phase 변경 순서에 따라 Phase 변경 함수 호출



SALT 신호 Phase 변경

```
//@LibSALT test code for changing phase every 30
string targetTLNode = "cluster_5507814[illegible]";
TrafficSignal * targetTL = SC->getLearningTrafficSignalByNodeID(targetTLNode);
std::map<string,TrafficSignalSchedule*> tsm = targetTL->getLearningTrafficSignalScheduleMap();
SALTTime targetTriggerDuration = 30;
std::map<int,pair<std::string,int>> orderlist;
int turn = 0;
for(auto ts : tsm){
    for(int i = 0; i < ts.second->getMyPhaseNum(); i++){
        cout<<"in FOR"<<endl;
        cout<<"phase duration: "<<ts.second->convertIndex2Duration(i)<<endl;
      if(ts.second->convertIndex2Duration(i) > 5) {
          cout<<"in Second FOR"<<endl;
        orderlist[turn]=make_pair(ts.first, i);
        turn++;
      }
    }
}

while(!SC->checkEnd()) {
    // get current step + update status of vehicles, traffic signal and event
    SALTTime currentStep = SC->getCurrentStep();

     if (currentStep % 1200 == 0) {
        cout<<"Current Schedule ID : " << targetTL->getCurrentLearningTrafficSignalScheduleID() << ", [Phase Sta
     }

    if((currentStep % targetTriggerDuration)==0){
        int ordering = (currentStep / targetTriggerDuration) % orderlist.size();
        auto curTSScheduleID = orderlist[ordering].first;
        auto curTSPhaseIndex = orderlist[ordering].second;

        cout<<"Ordering: " << ordering << ", current ts schedule id: " << curTSScheduleID << ", current ts phase i

        cout<<"[Trigger Point]"<< endl;
        cout<<"TL Change Time: " << currentStep << ", Current Schedule ID : " << targetTL->getCurrentLearningTraff

        SC->changeTLPhase(currentStep, targetTLNode, curTSScheduleID, curTSPhaseIndex);

        cout<<"[After Changing] Current Schedule ID : " << targetTL->getCurrentLearningTrafficSignalScheduleID() <
    }
    SC->doSimulationStep(currentStep); // doSimulationStep will increase SC::currentStep

    // (optional) print
    SC->printStep(currentStep,NUMBER_PRINT);

}
```

```
[Simulation Start]
Ordering: 0, current ts schedule id: 1, current ts phase index: 0
[Trigger Point]
TL Change Time: 0, Current Schedule ID : 1, [Phase State : rrrrGGGrrrrrrrGGGr]
Before : 3600, [ScheduleID: 1, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrGGGr]
After Changing TL Phase:3600[ScheduleID: 1, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrGGGr]
[After Changing] Current Schedule ID : 1, [Phase State : rrrrGGGrrrrrrrGGGr]
Ordering: 1, current ts schedule id: 1, current ts phase index: 3
[Trigger Point]
TL Change Time: 30, Current Schedule ID : 1, [Phase State : rrrrGGGrrrrrrrGGGr]
Before : 3600, [ScheduleID: 1, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrGGGr]
After Changing TL Phase:3600[ScheduleID: 1, PhaseIndex: 3, CurrentPhaseState: rrrrrGGGGrrrrrrrrG]
[After Changing] Current Schedule ID : 1, [Phase State : rrrrrGGGGrrrrrrrrG]
Ordering: 2, current ts schedule id: 1, current ts phase index: 5
[Trigger Point]
TL Change Time: 60, Current Schedule ID : 1, [Phase State : rrrrrGGGGrrrrrrrrG]
Before : 3600, [ScheduleID: 1, PhaseIndex: 3, CurrentPhaseState: rrrrrGGGGrrrrrrrrG]
After Changing TL Phase:3600[ScheduleID: 1, PhaseIndex: 5, CurrentPhaseState: GGGrrrrrrrGGGrrrrrrr]
[After Changing] Current Schedule ID : 1, [Phase State : GGGrrrrrrrGGGrrrrrrr]
Ordering: 3, current ts schedule id: 1, current ts phase index: 8
[Trigger Point]
TL Change Time: 90, Current Schedule ID : 1, [Phase State : GGGrrrrrrrGGGrrrrrrr]
Before : 3600, [ScheduleID: 1, PhaseIndex: 5, CurrentPhaseState: GGGrrrrrrrGGGrrrrrrr]
After Changing TL Phase:3600[ScheduleID: 1, PhaseIndex: 8, CurrentPhaseState: rrrGGrrrrrrrrGGGrrr]
[After Changing] Current Schedule ID : 1, [Phase State : rrrGGrrrrrrrrGGGrrr]
Ordering: 4, current ts schedule id: 13, current ts phase index: 0
[Trigger Point]
TL Change Time: 120, Current Schedule ID : 1, [Phase State : rrrGGrrrrrrrrGGGrrr]
Before : 3600, [ScheduleID: 1, PhaseIndex: 8, CurrentPhaseState: rrrGGrrrrrrrrGGGrrr]
After Changing TL Phase:3600[ScheduleID: 13, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrrGGGr]
[After Changing] Current Schedule ID : 13, [Phase State : rrrrGGGrrrrrrrrGGGr]
Ordering: 5, current ts schedule id: 13, current ts phase index: 3
[Trigger Point]
TL Change Time: 150, Current Schedule ID : 13, [Phase State : rrrrGGGrrrrrrrrGGGr]
Before : 3600, [ScheduleID: 13, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrrGGGr]
After Changing TL Phase:3600[ScheduleID: 13, PhaseIndex: 3, CurrentPhaseState: rrrrrGGGGrrrrrrrrG]
[After Changing] Current Schedule ID : 13, [Phase State : rrrrrGGGGrrrrrrrrG]
Ordering: 6, current ts schedule id: 13, current ts phase index: 5
[Trigger Point]
TL Change Time: 180, Current Schedule ID : 13, [Phase State : rrrrrGGGGrrrrrrrrG]
Before : 3600, [ScheduleID: 13, PhaseIndex: 3, CurrentPhaseState: rrrrrGGGGrrrrrrrrG]
After Changing TL Phase:3600[ScheduleID: 13, PhaseIndex: 5, CurrentPhaseState: GGGrrrrrrrGGGrrrrrrr]
[After Changing] Current Schedule ID : 13, [Phase State : GGGrrrrrrrGGGrrrrrrr]
Ordering: 7, current ts schedule id: 13, current ts phase index: 8
[Trigger Point]
TL Change Time: 210, Current Schedule ID : 13, [Phase State : GGGrrrrrrrGGGrrrrrrr]
Before : 3600, [ScheduleID: 13, PhaseIndex: 5, CurrentPhaseState: GGGrrrrrrrGGGrrrrrrr]
After Changing TL Phase:3600[ScheduleID: 13, PhaseIndex: 8, CurrentPhaseState: rrrGGrrrrrrrrGGGrrr]
[After Changing] Current Schedule ID : 13, [Phase State : rrrGGrrrrrrrrGGGrrr]
```

이슈 정리

- 신호 최적화에서 모든 스케줄의 Phase에 대해서 학습 수행 시
  - TODPlan을 제약조건 사용 시, 시간에 따라 TODPlan 내 Phase들로만 학습 진행
    - *→ 학습 필요가 있는가….*
    - 안) 현재 SALT Schedule 변경 로직에 따라 Schedule 변경하고, 학습과정에서 현재 신호 스케줄 정보를 확인 후 해당 ScheduleID에 포함된 Phase를 변경하면서 학습 진행
  - 그렇지 않은 경우, Tss.xml에 교차로에서 가능한 모든 스케줄 정보를 포함하여 학습 수행
    - 안) 학습 관련 정보를 시나리오 다운로드 단계에 반영해서 학습 대상 교차로 정보만 확장

신호 PHASE 변경 테스트 (PYTHON)

libsalt::TrafficLightSignal

- Class TrafficSignalManager 내 libsalt용 get/set/change 함수 호출을 위한 클래스

```
class TrafficLightSignal {
public:
    static SALT::TrafficSignalManager* getTLSManager();
    static std::vector<std::string> getTLSIDList();
    static int getTLSIDCount();
    static TLSLogic* getTLSByNodeID(std::string _tlsID);
    static std::vector<std::string> getTLSScheduleIDsByNodeID(std::string _tlsID);
    static std::string getCurrentTLSScheduleIDByNodeID(std::string _tlsID);
    static TLSSchedule* getCurrentTLSScheduleByNodeID(std::string _tlsID);
    static std::vector<std::pair<int,std::string>> getTLSPhaseByNodeID(std::string _tlsID, std::string _scheduleID, int _phaseIndex);
    static std::string getCurrentTLSPhaseStateByNodeID(std::string _tlsID);
    static int getCurrentTLSPhaseIndexByNodeID(std::string _tlsID);
    static int getLastTLSPhaseSwitchingTimeByNodeID(std::string _tlsID);
//        static SALT::SALTTime getLastTLSScheduleSwitchingTimeByNodeID(std::string _tlsID);


        static SALT::Result setTLSByNodeID(std::string _tlsID, TLSLogic* _logic);
//        static void setTLSScheduleByNodeID(std::string _tlsID, TLSSchedule* _schedule);
//        static void setTLSPhaseStateByNodeID(std::string _tlsID, int _phaseIndex, std::string _phaseState);
//        static void setTLSPhaseByNodeID(std::string _tlsID, int _phaseIndex, std::pair<SALT::SALTTime,std::string> _phasePair);
//        static void setTLSPhaseDurationByNodeID(std::string _tlsID, int _phaseIndex, SALT::SALTTime _phaseDuration);
        static void changeTLSPhase(int triggertime, std::string _nodeID, std::string _scheduleID, int _phaseIndex);

private:
    static std::map<std::string, int> _transformTODPlan(TLSLogic* _logic);


};
```

libsalt::TLSLogic

- class TrafficSignal의 SWIG Wrapper 생성용 클래스 정의

```
class TLSLogic {
public:
    TLSLogic(std::string _nodeID, std::map<std::string,TLSSchedule*> _schedules, std::map<SALT::SALTTime,std::string> _todplan)
    : myTLSID(_nodeID), myScheduleMap(_schedules), myTODPlan(_todplan){}
    virtual ~TLSLogic() {}

    std::string myTLSID;

    // Time of Day Plan
    std::map<std::string, TLSSchedule*> myScheduleMap; // (schedule name, schedule itself)
    std::map<int, std::string> myTODPlan; //ToD Plan (schedule starting time, schedule name)

    std::string getTLSID() {  return myTLSID;  }
    std::map<std::string, TLSSchedule*> getScheduleMap() {
        return myScheduleMap;
    }
    std::map<int, std::string> getTODPlan() {
        return myTODPlan;
    }

};
```

```
#ifdef SWIG
%{
    namespace swig {
        template <>  struct traits<libsalt::TLSSchedule> {
        typedef pointer_category category;
        static const char* type_name() { return"TLSSchedule"; }
        };
    }

%}
%template(TLSSchedulesMap) std::map<std::string,libsalt::TLSSchedule *>; // *NOPAD*
%template(TLSTODMap) std::map<int, std::string>;
#endif
```

ETRI
Electronics and Telecommunications Research Institute

libsalt::TLSSchedule

- Class TrafficSignalSchedule 의 SWIG Wrapper 생성용 클래스 정의

```
class TLSSchedule {
public:
    TLSSchedule(std::string _scheduleID, int _offset, std::vector<std::pair<int,std::string>> _phasevector)
    : myID(_scheduleID), myOffset(_offset), myPhaseVector(_phasevector) {}
    virtual ~TLSSchedule() {}
    std::string myID;
    int myOffset; // ????
    // phase := pair<duration,state>
    std::vector<std::pair<int,std::string>> myPhaseVector;

    std::vector<std::pair<int,std::string>> getPhaseVector() {  return myPhaseVector; }
    std::string getSchedulID() {  return myID;  }
    int getOffset() {  return myOffset;  }
};
```

```
#ifdef SWIG
%template(TLSPhaseVector) std::vector<std::pair<int,std::string>>;
#endif
```

libsalt.i

- Class TrafficLightSignal 함수에 대한 python 함수 정의

```
def getTLS(nodeid):
    return trafficsignal.getTLSByNodeID(nodeid)

def getTLSScheduleIDs(nodeid):
    return trafficsignal.getTLSScheduleIDsByNodeID(nodeid)

def getCurrentTLSScheduleID(nodeid):
    return trafficsignal.getCurrentTLSScheduleIDByNodeID(nodeid)

def getCurrentTLSSchedule(nodeid):
    return trafficsignal.getCurrentTLSScheduleByNodeID(nodeid)

def getTLSPhase(nodeid):
    return trafficsignal.getTLSPhaseByNodeID(nodeid)

def getCurrentTLSPhaseState(nodeid):
    return trafficsignal.getCurrentTLSPhaseStateByNodeID(nodeid)

def getCurrentTLSPhaseIndex(nodeid):
    return trafficsignal.getCurrentTLSPhaseIndexByNodeID(nodeid)

def getLastTLSPhaseSwitchingTime(nodeid):
    return trafficsignal.getLastTLSPhaseSwitchingTimeByNodeID(nodeid)
```

SALT 신호 Phase 변경 테스트 (python)

- 신호 Phase 변경 테스트
  - C++ 코드 기반 테스트와 동일한 진행
  - 특정 시간 마다, 특정 사거리 신호에 포함된 모든 스케줄에 존재하는 Phase를 확인하여, TriggeringTime 마다 순차적으로 변경하는 테스트
    - 평일 오전 6시 - 오전 7시 59분 (3시간) 동안 강동구 시뮬레이션 수행 중, 길동사거리의 신호의 모든 스케줄 정보를 기반으로 Phase 진행 순서를 생성하고, 특정 시간 (= 매 30초) 마다 신호 Phase를 다음 Phase로 변경하는 테스트

  - 테스트 결과
    - 다음 스케줄이 존재하는 경우, 변경 됨을 확인함.

```
[hwonsong@handcourage-ex2 libsalt % sh s.sh
/Users/hwonsong/CLionProjects/uniq/data/2020-gd-tlchange-test.json
scenarioFile: /Users/hwonsong/CLionProjects/uniq/data/2020-gd-tlchange-test.json
[SALT Simulator 2.0]
Loading Road Network ... done
Loading Vehicle Demand ... done
Loading Traffic Signal System ... done
[Simulation Output (Periodic)] >> output/2020TLChangeTest/2020TLChangeTest-PeriodicOutput.csv
[Progress status] >> output/2020TLChangeTest/progress.txt
[Simulation Info]
Input
        -Simulation Name: 2020TLChangeTest
        -Road Network: /Users/hwonsong/CLionProjects/uniq/data/2020TLChangeTest/node.xml, /Users/hwonsong/CLionProjects/uniq/data/2020TLChang
eTest/edge.xml, /Users/hwonsong/CLionProjects/uniq/data/2020TLChangeTest/connection.xml
        -Vehicle Demand: /Users/hwonsong/CLionProjects/uniq/data/routes/2019/gd_trips_odpair_weighted_selection_mon_final_20191223_r_02.rou.x
ml
        -Traffic Light System: /Users/hwonsong/CLionProjects/uniq/data/2020TLChangeTest/tss.xml
        -Time Range: 0~7140
Output
        -Output Per Period: output/2020TLChangeTest/2020TLChangeTest-PeriodicOutput.csv

[Simulation Start]
cluster_572701114_572701116_572701117_572712746_572712755
2
['1', '13']
{0: ['1', 0], 1: ['1', 3], 2: ['1', 5], 3: ['1', 8], 4: ['13', 0], 5: ['13', 3], 6: ['13', 5], 7: ['13', 8]}
Ordering: 0, current ts schedule id: 1, , current ts phase index: 0
[Trigger Point]
TL Change Time: 0, Current Schedule ID : 1  [Phase State : rrrrGGGrrrrrrrGGGr]
Before : 3600, [ScheduleID: 1, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrGGGr]
After Changing TL Phase:3600[ScheduleID: 1, PhaseIndex: 0, CurrentPhaseState: rrrrGGGrrrrrrrGGGr]
[After Changing] Current Schedule ID : 1 [Phase State : rrrrGGGrrrrrrrGGGr
Ordering: 1, current ts schedule id: 1, , current ts phase index: 3
```

SALT 신호 Phase 변경 테스트 (python)

Python 테스트 시 SALT 시나리오 파일 및 데이터

- Git: traffic-simulator/data/2020-gd-tlchange-test.json
- Git: traffic-simulator/data/2020TLChangeTest/

Python 테스트 코드 예

- Git: traffic-simulator/test/libsalt/test_tls_phase_change.py 참고

```
libsalt.start(salt_cfg)

step = 0

targetTLNode = "cluster_572701114_572701116_572701117_572712746_572712755"
tl = libsalt.getTLS(targetTLNode)
print(tl.myTLSID)
print(tl.getScheduleMap().size())
tsm = tl.getScheduleMap()
print(tsm.keys())
triggerduration = 30
orderdic = {}
turn = 0
for k in tsm.keys():
    for p in enumerate(tsm[k].getPhaseVector()):
        if p[1][0] > 5:
            pair = []
            pair.append(k)
            pair.append(p[0])
            orderdic[turn] = pair
            turn+=1

print(orderdic)

while step <= 5400:
    curstep = libsalt.getCurrentStep()

    if (curstep % triggerduration)==0:
        ordering = (curstep / triggerduration) % len(orderdic);
        curscheduleid = orderdic[ordering][0];
        curphaseindex = orderdic[ordering][1];

        print("Ordering: {}, current ts schedule id: {}, , current ts phase index: {}".format(ordering, curscheduleid, curphaseindex))
        print("[Trigger Point]")
        print("TL Change Time: {}, Current Schedule ID : {}  [Phase State : {}]".format(curstep,
                                                                                        libsalt.getCurrentTLSScheduleID(targetTLNode),
                                                                                        libsalt.getCurrentTLSPhaseState(targetTLNode)))
        libsalt.trafficsignal.changeTLSPhase(curstep, targetTLNode, curscheduleid, curphaseindex)

        print("[After Changing] Current Schedule ID : {} [Phase State : {}".format(libsalt.getCurrentTLSScheduleID(targetTLNode),
                                                                                   libsalt.getCurrentTLSPhaseState(targetTLNode)))

    libsalt.simulationStep()
    #get_values()
    step += 1

libsalt.close()
print("simulation end!!!")
```

기타 이슈 및 이후 개발 일정

- TODPlan 내 스케줄 단위 변경
  - 특정 시간, 특정 사거리 신호 Schedule을 변경 → 추후 추가 개발
    - 1차 테스트와 달리, 최적화 학습 모드에서 스케줄 변경 함수로 변경/추가 구현
      - 고려사항
        - » Schedule 변경이 이루어지기 위한 Offset 계산
        - » 연동 신호 간 Offset 계산
      - chageTLSchedule() → 추후 개발

- 진행 일정
  - Simulation 결과에 대한 Value Retrieval Function 개발 완료 후 테스트 진행 (~8/28)
  - 동적 인터페이스 1차 릴리즈 완료 예정 (~9/10)
    - 1차 릴리즈 목표: Adaptive 신호 운영이 가능한 형태로 학습 진행 가능