논 문
60-11-10

PRT 시스템의 운행 시뮬레이터 개발

Development of the Operation Simulator for the PRT System

정 락 교† · 김 백 현* · 황 현 철*
(Rag-Gyo Jeong · Beak-Hyun Kim · Hyeon-Chyeol Hwang)

Abstract - A time based software simulator for the PRT system operation is presented. The purpose of the simulator is to estimate the passenger transportation performance of the PRT system. In this paper, it is presented how the system is modeled in the simulator to estimate passenger transportation performance and the running algorithm of the modeled subsystem. An application sample is also presented to find the system's design parameter to satisfy the transportation needs.

Key Words : PRT, Simulation, Asynchronous control

1. 서 론

PRT(Personal Rapid Transit) 시스템은 현재 세계 여러 나라에서 개발 중에 있고 영국 ULtra사가 히드로공항 터미널5-주차장간 연계노선과 네덜란드 2getthere사가 아랍에미리트연합 마스다르시에서 Pilot 노선이 개발되어 초기운영 중에 있으며, 국내에서도 실용화를 전제로 개발 중에 있다. 본 논문에서는 개발된 PRT 운행제어 시뮬레이터와 그 응용방안을 기술한다. 본 시뮬레이터는 PRT 시스템의 실용화 시에 수행되는 시스템의 경제성 평가와 기본 설계 시에 필요한 역사규모, 위치, 소요차량대수를 산정하는 툴(tool)의 기능을 수행할 목적으로 개발되었다. 시뮬레이터는 기본적으로 차량의 움직임을 모의하며, 차량의 가이드웨이 상에서 주행 및 역사 내의 승강장에서의 움직임이 모의된다. 또한 역사 내에서는 승객의 승차나 하차도 모의 된다. 역사 내에 공차가 있으면 승객이 탑승하고 공차가 없으면 승객은 대기한다. 반대로 공차는 있는데 승객이 없으면 빈 차량이 승강장에서 대기하고 있다. 승객은 시스템의 수송수요에 의해서 불규칙적으로 역사에 출현한다. 고객만족 측면에서는 승객이 역사에 출현하면 언제나 대기시간 없이 바로 차량에 탑승하여 목적지로 갈 수 있으면 가장 바람직하다. 그러나 이것은 가능하지 않을 수도 있고 혹은 역사의 규모를 늘리거나, 혹은 차량의 수를 늘려서 가능하여 질수도 있다. 물론 초기 투자비용은 증대될 것이다. 여기서 각 역사별 수송수요, 이용 가능한 차량대수, 역사규모 등이 투자비용과 맞물려 서로의 함수가 되어 있다. 또한 동일한 인프라와 차량을 갖고도 어떻게 차량을 운용할 것인지에 따라 결과는 크게 달라진다. 예를 들면 역사에 승객이 대기하고 있을 때 어디에 있는 공차를 배차할 것인지? 공차 수요를 미리 예측하여 배차할 수는 없는지? 이러한 운영알고리즘 최적화에 따라서 수송수요 처리 능력은 크게 달라진다. 본 시뮬레이터는 어떠한 시스템이 최적화 시스템인지? 어떠한 운영 알고리즘이 적절한지? 검토를 위한 툴(tool)을 제공한다. 본 시뮬레이터에서는 역사의 규모, 차량 대수, 공차관리 방법 등을 변경해 가면서 시스템의 수송능력 평가가 가능하다. PRT 시스템에서 수송능력은 단순히 어느 가이드웨이를 차량이 시간당 몇 대 통과하느냐의 문제가 아니다. PRT 시스템에서는 정해진 노선이 없으며 차량은 필요시 최적경로를 찾아서 출발지 역사에서 목적지 역사로 무정차로 주행한다. 따라서, PRT 시스템에서의 수송능력이란 승객이 역사에 출현하여 탑승할 때까지의 대기시간을 기준으로 하는 것이 타당하다고 판단된다. 물론 차량의 평균주행속도도 수송수요에 영향을 미치지만 주행속도가 늦어지면 공차나 승객탑승차량의 역사 도착시간도 늦어져서 결국은 승객의 대기시간의 증가로 나타난다. 본문에서는 시뮬레이터를 이용한 최적화 시스템 설계의 간략한 예가 소개된다. 시뮬레이터에서 차량의 주행제어는 비동기식 방법으로 이루어진다. 비동기식제어 방법에서 차량은 항상 선행차량(leader car)을 안전거리를 확보하며 따라간다. 이 안전거리 확보를 위한 주행제어 알고리즘 및 최적경로 탐색 알고리즘은 개발된 알고리즘을 적용하였다[1]. 본문에서는 시스템의 전체 구성과 각 구성시스템의 모의과정을 소개하고 그 응용예가 소개된다. 전체 시스템은 수송수요에 영향을 미치는 구성시스템 및 주행제어를 위해 필요한 구성시스템이 모의 되었으며, 여기에는 승객, 중앙제어시스템, 역사, 노선, 차량 시스템이 포함된다. 노선은 세분

† 교신저자, 정회원 : 한국철도기술연구원 책임연구원
E-mail : rgjeong@krri.re.kr
* 정 회 원 : 한국철도기술연구원 선임연구원
접수일자 : 2011년 8월 1일
최종완료 : 2011년 10월 25일

화되어 트랙으로 모의된다.

2. 본 론

2.1 시뮬레이터의 구성

시뮬레이터는 그림 1과 같이 구성되며 그림 2의 순서로 수행된다. 역사운전(Station_Operation)에서는 중앙제어컴퓨터 지시에 의한 공차의 배차, 역사 내 차량의 이동, 승객의 탑승, 하차 등이 수행된다. 승객운전(Passenger_Operation)에서는 승객 수송수요에 의해서 승객의 역사 출현이 수행된다. 트랙운전(Track_Operation)이란 각 트랙 내의 혹은 트랙 사이 차량의 리더-펄로우어(leader-follower) 관계의 설정을 의미한다. 본 시뮬레이터에서 차량의 주행제어는 비동기식 방법에 의해 제어되며 비동기식 제어 방식에서는 기본적으로 차량이 리더 차량을 쫓아가므로(follow) 차량의 주행제어를 위해서는 차량 상호간의 리더-펄로우어 설정이 중요하다. 차량운전(Car_Operation)에서는 모든 트랙 내 차량의 주행을 수행한다. 모든 차량은 리더 차량과의 안전거리를 유지하고 주어진 최대 기동파라메타 내에서 가능한 한 최고의 속도로 주행하도록 제어된다[1]. 시뮬레이터에서는 또한 모든 차량이 중앙제어컴퓨터에서 제시되는 기동 파라메타 즉 저크(jerk)에 의해 운전되며 차량과 중앙제어컴퓨터는 실시간으로 정보를 교신하며 이 정보교신 및 차량 운전 반응은 레이턴시(Latency) 시간 내에서 이루어진다.

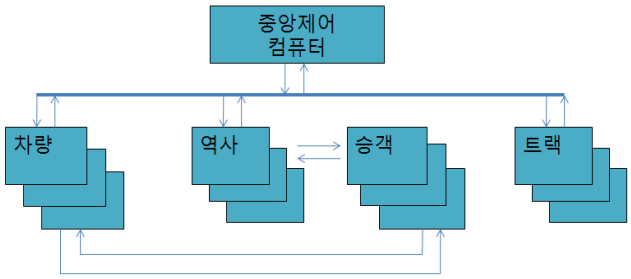

그림 1 시뮬레이터 구성 및 구성 시스템간의 관계
Fig. 1 Relationship between simulator and configuration system

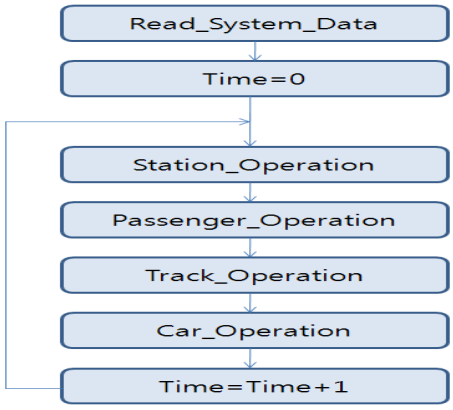

그림 2 시뮬레이션 수행 흐름도
Fig. 2 Flow-chart for Performing simulation

2.2 구성 시스템별 운전 알고리즘

2.2.1 역사 운전(Station_Operation)

역사의 승강장(berth)은 기본적으로 직렬 일렬 승강장 형태로 모의되며 출구 및 입구가 별도로 구분되는 타입(Single-platform on a siding with two entrances)으로 모의 하였다. 그림 3은 이와 같은 형태의 승강장 개념을 보여주는 데 앞의(그림의 좌측)의 2개 승강장에서는 탑승, 뒤의(그림의 우측) 3개의 승강장에서는 하차만 가능하다. 즉 승차 승강장, 하차 승강장이 구별되어 있다. 승객의 탑승, 하차, 승강장 내에서의 차량의 이동은 역사 제어 시스템에 의해 수행되며 각각의 수행시간은 각각의 타이머로 사용자에 의해 설정된다. 즉 사용자는 탑승 타이머, 하차타이머, 차량 이동 타이머를 설정할 수 있다. 따라서 다른 타입의 승강장 형태의 시뮬레이션은 이 타이머에 의해 간접적 방법으로 역사 승객처리 능력이 모의 될 수 있다. 사용자는 또한 탑승용 승강장의 수나 하차용 승강장의 수를 설정함으로써 역사 규모를 선정한다. 차량의 상태는 차량 속성인 모드(mode)로 표현된다. 역사 내 차량의 모드는 표 1과 같다. 차량이 역사에 도착하면 일단 역사 진입 대기 차량 리스트에 등록된다(active=0, mode='i'). 역사 승강장을 모두 차량이 점유하고 있어서 더 이상 차량 진입이 불가할 경우에 대비하여 역사 진입대기 차량이 머무를 수 있는 대기 장소(input queue)가 있으며 맨 뒤 승강장이 비게 되면 즉시 순서대로 진입한다. 이 후 차량이 승객을 태우고 역사를 출발하는 과정은 그림 4와 같이 진행된다. 프로시저 'Car_enter_the _berth' 에서는 맨 마지막 승강장의 점유를 검토하고 비워져 있으면 맨 마지막 승강장으로 진입한다. 물론 맨 마지막 승강장까지 차량이 이동하는 시간은 Move_timer를 설정하여 제어한다. 다음 프로시저 'Move_car_in_station'에서는 역사 내의 각 승강장을 검토하여 바로 앞의 승강장이 비워있으면 차량은 앞의 승강장으로 이동하며 알고리즘은 그림 5와 같다. 물론 하차 승강장에서 승차 승강장으로 이동할 때는 승객이 하차하였는가를 검토하여 승객이 하차하지 않았으면 승객 하차 후에 이동한다. 즉 하차 승강장에서 승차 승강장으로 이동할 때는 차량 내 승객이 없어야 한다. 또한 승객이 탑승 혹은 하차 중이면 이동하지 않는다. 차량 이동이 종료된 후 차량의 모드는 'w'(wait, 대기)로 바뀐다. 다음 프로시저 'Unboarding'에서는 하차 승강장내 차량을 검토하여 모드가 'w'이고 차량 내 승객이 있는 경우 승객을 하차시키며 알고리즘은 그림 6과 같다. 이때 모드는 'u'로 바뀌며 승객 하차에 필요한 시간은 하차_타이머(Unboard_timer)에 설정된다. 승객 하차가 완료되면 모드는 다시 'w'로 바뀐다. 다음 프로시저 'Boarding'에서는 승강장에 모드가 'w'이고 승객이 탑승하지 않은 차량이 있는가를 검토한다. 이러한 차량이 존재하고 또한 역사 대기승객이 있는 경우 승객을 차량에 탑승시킨다. 이때 모드는 'b'로 바뀌고 탑승에 필요한 시간 승차_타이머(Boarding_timer)가 설정된다. 승객 탑승 알고리즘은 하차 알고리즘과 같은 개념이어서 생략하였다. 승객 탑승이 완료되면, 즉 승차_타이머가 0이 되면 차량의 모드는 다시 'w'로 바뀐다. 다음 프로시저 'Dispatch_car'에서는 맨 앞의 승강장의 차량을 검토하여 모드가 'w'이고 승객이 탑

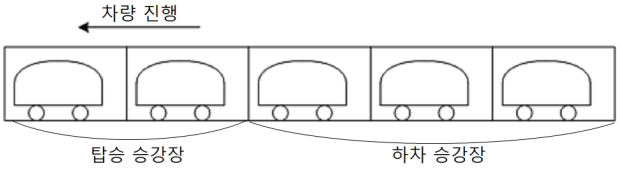

그림 3 직렬 일렬 승강장 개념도
Fig. 3 Concept of the serial platform

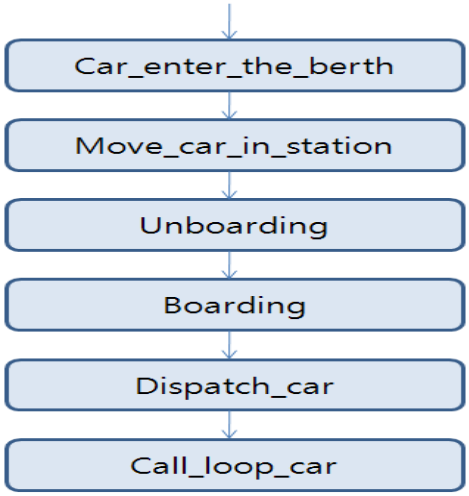

그림 4 역사 운영 흐름도
Fig. 4 Flow-char for operating station

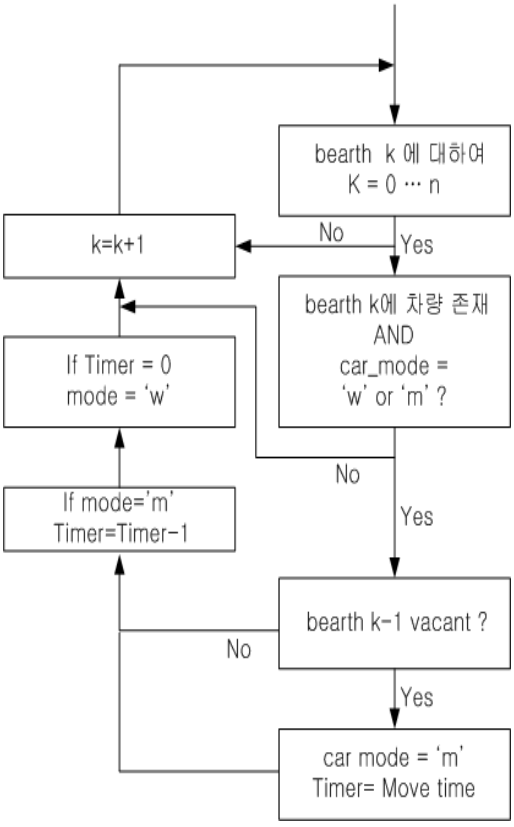

그림 5 차량의 승강장 내 이동 알고리즘
Fig. 5 Algorithm for moving vehicle in the platform

승하여 있으면 차량을 출발시킨다. 차량의 출발은 차량 속성 'active'를 '1'로 설정하고 차량의 위치를 맨 처음 트랙의 위치에서 '0'으로 설정함으로서 이루어진다. 차량 속성 'active'가 '1'이 되면 중앙제어컴퓨터에서 차량주행제어를 하게 된다. 다음 프로시저 'Call_loop_car'에서는 역사에 공차는 없는데 대기승객이 있을 경우 중앙제어컴퓨터에 공차의 배차를 요구한다. 루프카(Loop_car)란 일정한 목적지 없이 일정한 루프를 돌고 있는 공차를 의미한다. 그림 7은 역사 모델을 보여준다. 그림 7에서 44번 승객이 대기하고 있고 그림 8에서 다음 순간 대기하고 있는 차량 58번에 탑승한 모습을 보여 준다.

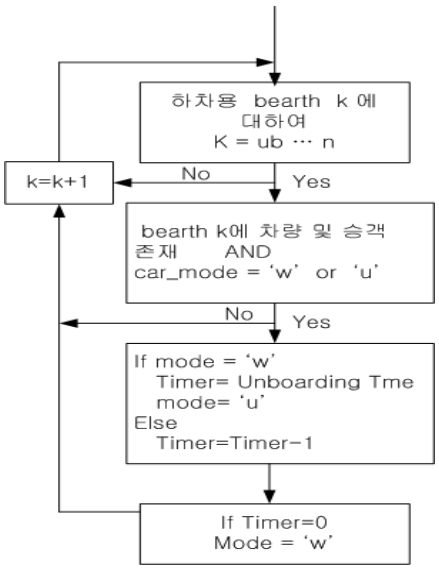

그림 6 승객 하차 알고리즘
Fig. 6 Algorithm for unloading passenger

표 1 역사 내 차량 모드
Table 1 Station in vehicle mode

차량 역사 내 위치(Active=0), 역사제어장치 제어	
mode	상 태
w	승차 대기
m	승강장 내 이동 중, 이동 타이머에 의해 제어
b	승차(boarding) 중, 승차 타이머에 의해 제어
u	하차(unboarding) 중, 하차 타이머에 의해 제어
i	승강장 진입 대기

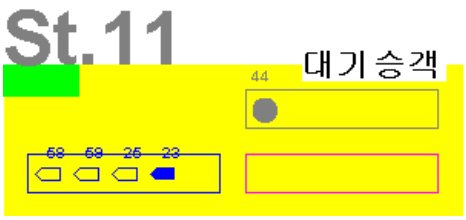

그림 7 역사 모델
Fig. 7 Station model

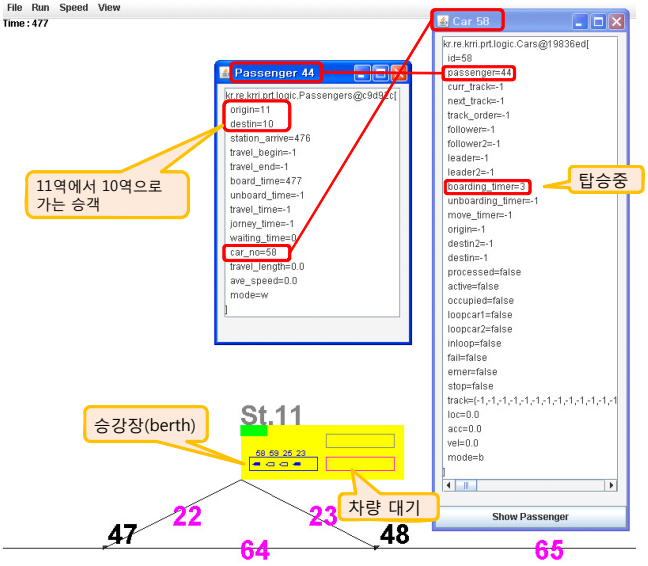

그림 8 역사 모델 및 승객 차량 속성
Fig. 8 Station model and passenger vehicle property

2.2.2 승객 관리(Passenger_Operation)

승객은 주어진 수송수요에 의해서 불규칙하게 각 역사에 출현하여 승객 고유의 아이디(ID)를 부여받고 승객 대기 리스트에 등록된다. 탑승 승강장에 대기 차량이 있을 경우 가장 앞선 대기 차량에 탑승하고 승객대기리스트에서 삭제된다. 승객이 대기 차량에 탑승하면 차량에 목적지 역사를 입력시키고 차량은 목적지 역사로 가는 최적의 경로를 탐색하여 주행한다. 차량이 목적지 역사에 도착하여 하차 승강장에서 정지하면 하차함으로써 여행이 종료된다.

2.2.3 트랙 관리(Track_Operation)

시스템 노선에서 역사, 분기지점, 병합지점 혹은 제한속도가 변하는 지점을 노드(node)로 설정하고 노드와 노드사이를 트랙으로 명명하였다. 각 트랙은 자기 트랙내에서 주행하는 차량의 정보를 중앙제어컴퓨터에 실시간으로 제공한다. 중앙제어컴퓨터는 각 트랙내 차량 및 위치를 통보받고 차량과 차량 사이의 리더-펄로우어 관계를 설정하고 차량의 주행모드를 설정한다. 차량의 주행모드는 특별한 언급이 없는 한 정상주행모드인 'n'으로 설정된다. 그림 9는 그림 2의 'Track_operation'에서 수행되는 상세내용을 보여 준다. 프로시저 'Check_leader_follower_in_the_same_track'은 동일 트랙 내의 모든 차량에 대하여 그 위치에 따라 리더(leader), 펄로우어(follower) 관계를 설정한다. 다음 프로시저 'Check_leader_follower_in_the_different_track'에서는 트랙 내 선두차량과 그 차량의 경로 상 다음 트랙의 후미 차량과의 리더-펄로우어 관계를 설정한다. 또한 현재 트랙이 분기 트랙인 경우 트랙 내 선두 차량과 트랙 내 2번째 선두 차량의 다음 주행 트랙이 서로 다를 경우 2번째 선두 차량은 리더 차량 외에 또 다른 리더 차량이 필요하며, 또 다른 리더 차량을 리더2(leader2) 차량이라 부른다. 즉 현재 트랙의 타입이 분기 트랙이면 선두차량과 2번째 선두차량의 다음 경로에 따라서 리더2 차량이 설정된다. 이 경우 차량의 모드는 'd'로 설정된다.

다음 프로시저인 'make_new_merge_order'에서는 병합구간의 차량들에 대한 리더2-펄로우어 관계를 설정한다. 병합구간에서는 자기 트랙 내 선행차량이 리더 차량이 된다. 그러나 경우에 따라서 본 트랙에 병합하는 다른 트랙을 주행하는 차량을 리더2 차량으로 설정할 필요가 있으며 이 경우 리더2 차량의 설정과 함께 차량의 주행모드는 'm'으로 설정된다. 다음 프로시저인 'Check_approaching_car'는 트랙의 끝 노드가 역사인 경우에 한 한다. 즉 현재 트랙이 마지막 주행 트랙이므로 트랙 끝에서는 속도가 0에 접근해야 한다. 따라서 이러한 트랙인 경우 차량의 위치에 따라 언제부터 일정한 감속도로 감속하여 정 위치에 정차시킬 것인가를 판단하고 차량이 이 범위에 위치할 경우 주행모드를 'a'로 바꾼다.

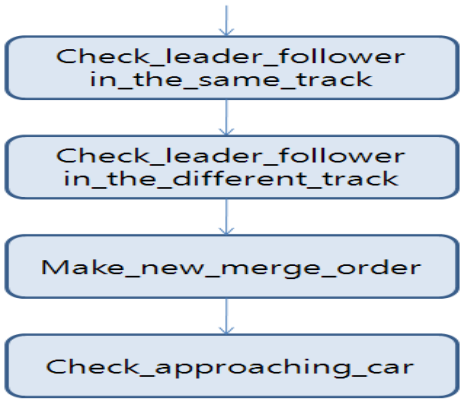

그림 9 트랙 관리 흐름도
Fig. 9 Flow chart for Track management

2.2.4 차량 운전(Car Operation)

차량의 속성인 'active'가 '0'이면 차량이 역사 내에 존재함을 의미하며, 'active'가 '1'이면 역사 밖의 트랙에 존재함을 나타낸다. 중앙제어컴퓨터는 모든 차량 중에서 'active'=1인 차량에 한해 주행제어를 한다. 차량은 현재 차량이 위치하고 있는 트랙의 타입에 따라 주행 방법이 변하며, 표 2는 트랙에 따른 차량의 주행모드를 보여준다. 이 주행 모드는 트랙 관리 프로시저에서 설정된다. 본 시뮬레이터에서 적용한 비동기식 제어 알고리즘의 경우, 리더 차량이 없는 경우도 있다. 앞 차량과의 거리가 너무 길거나 실제로 맨 선두차량인 경우가 그렇다. 이 경우 리더차량의 번호를 -1로 부여함으로써 리더차량이 없음을 중앙제어컴퓨터에 알린다. 차량의 주행제어 파라메타가 구해지면 그 차량의 boolean 속성 'processed'가 'false'에서 'true'로 바뀐다. 따라서 차량의 주행 파라메타를 구하기에 앞서 리더 차량의 속성 'processed'=TRUE가 전제조건이 된다. 그림 10은 '차량운전' 프로시저의 논리 흐름도이다.

표 2 가이드웨이 상에서 차량의 운전모드
Table 2 Guideway for operation vehicle mode

차량 역사 밖 위치(Active=1), 중앙제어컴퓨터 제어	
mode	상 태
a	역사 진입(approach) 주행 모드, 역사와 차량 사이에 다른 차량이 없을 경우
n	정상 주행 모드, 선행 차량과 안전거리를 확보에만 유의하면서 주행
m	병합(merge) 구간 주행 모드,
d	분기(diverge) 구간 주행 모드

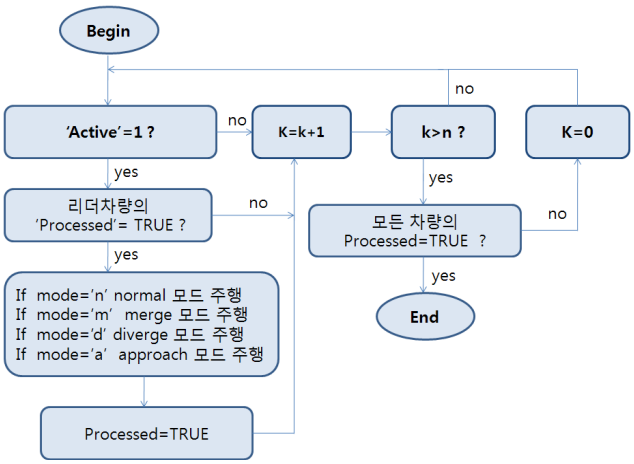

그림 10 차량 운전의 논리 흐름도
Fig. 10 Logical Flow chart for operating vehicle

2.2.5 공차관리 (Vacant vehicle management)

시뮬레이터에서 공차관리는 2가지 방법으로 수행된다. 하나는 공차 계획배차이며 또 하나는 루프(loop) 차량의 공급이다.

(1) 공차 계획배차

주어진 수송수요를 분석하면 통계적으로 공차가 요구되는 역사와 공차가 발생하는 역사로 구분하여 공차의 수를 계산한다. 이렇게 계산된 공차수요에 따라 계획적으로 시간에 맞춰서 배차하는 방법이다. 표 3은 수송수요를 차량수요로 변환한 표이며 이를 근거로 역사 도착 차량수와 역사 출발 차량 수를 계산하여 각 역사별 잉여차량 수를 계산하였다. 표에서 보는 바와 2, 3번역사는 공차가 발생하며, 1, 4번역사는 공차가 요구된다. 공차배차계획이란 공차가 발생하는 역사에서 공차가 요구되는 역사로 공차를 배차하는 것이다. 표 3에서는 2번역사에서 10대가 발생하므로 1번역사에 10대를 공급하고 3번역사는 60대가 발생하므로 1번역사에 30대 4번역사에 30 대 공차를 공급한다. 따라서 공차 배차 계획은 표 4와 같이 수립된다. 일반적으로 N_{ij} 를 역 i에서 역 j로 가는 시간당 차량의 수요라고 할 때 시간당 역 i 출발 차량 수 D_i 와 시간당 역 i 도착 차량 수 A_i는 식 (1) 및 (2)와 같다. 만약 $A_i > D_i$ 이면 잉여 공차가 생겨 다른 역사에 공차를 공급할 수 있고 시간당 역 i의 공차 공급 가능대수, S_i 는 식(3)과 같다. $D_i > A_i$이면 공차를 공급받아야 한다. 역 i의 필요 공차 수를 U_i라 하면 U_i는 식 (4)와 같다. 공급 역사 수를 N_S 필요 역사 수를 N_U 로 하고 역 i에서 역j로 가는 시간당 차량의 수를 E_{ij} 라 하면 S_i, D_i는 식(5) ,(6)으로 표현된다. 여기서 T_{ij} 를 공차가 역 i에서 역j로 가는 데 소요되는 시간으로 정의한다면 $E_{ij} \cdot T_{ij}$는 역 i에서 역 j로 가는 공차 주행시간의 합이다. 공차관리의 목적은 모든 공차에 대하여 이러한 합을 모두 합한 값, 즉 식 (7)의 F_E가 최소가 되는 조합을 찾아서 공차 배차 계획을 작성하는 것이다. 실제는 T_{ij}에 주행시간 대신 주행거리를 적용하였고 i를 선정한 후 j는 T_{ij}가 최소가 되는 최적경로를 탐색하여 i 및 j 조합을 선정하였다. 이는 실제 주행 시간이 경로의 트래픽에 따라 변하므로 사전에 공차배차계획을 수립하기 어렵기 때문이다.

$$D_i = \sum_{j \neq i} N_{ij} \tag{1}$$

$$A_i = \sum_{j \neq i} N_{ji} \tag{2}$$

$$S_i = A_i - D_i \tag{3}$$

$$U_i = D_i - A_i \tag{4}$$

$$S_i = \sum_{j=1}^{N_U} E_{ij} \qquad (i = 1, \ldots N_S) \tag{5}$$

$$U_i = \sum_{i=1}^{N_S} E_{ij} \qquad (j = 1, \ldots N_U) \tag{6}$$

$$F_E = \sum_{i=1}^{N_S} \sum_{j=1}^{N_U} E_{ij}\, T_{ij} \tag{7}$$

표 3 차량 배차 및 발생되는 공차 수요
Table 3 Allocation and demanded for vacant vehicle

출발지 역사	도착지 역사	차량 수요	출발 차량수	도착 차량수	잉여 차량수
1	2	40	150	110	-40
	3	50			
	4	60			
2	1	30	120	130	10
	3	40			
	4	50			
3	1	20	90	150	60
	2	30			
	4	40			
4	1	60	180	150	-30
	2	60			
	3	60			
계		540	540	540	0

표 4 표 3에 대응한 공차 배차 계획
Table 4 Plan vacant vehicle for table 3

출발지 역사	도착지 역사	공차배차 수
2	1	10
3	1	30
3	4	30

(2) 루프 차량의 배차

공차 수요를 만족시키기 위해 공차 계획배차 만으로는 공차 공급에 한계가 있다. 그것은 각 역사에 들어오는 차량 중 승객 수요를 뺀 차량 수만큼 만 공차로 공급할 수 있다는 것이다. 따라서 전체 역사에 배치된 차량 수가 절대적으로 적다든가 출퇴근 시간대처럼 공차가 많이 필요한 경우에는 공차 수요를 만족시킬 수 없다. 이에 대비한 방법이 소위 루프(loop) 차량의 공급이다. 이는 역사 외에 차량기지와 같은 곳에 다수의 공차를 보유하고 있다가 어느 지역에 공차수요가 증가하면 그 지역에 루프 차량을 공급하는 방법이다. 루프 차량이란 목적지 없이 일정한 루프를 계속 돌고 있는 차량을 의미한다. 어느 역사에서 대기승객이 증가하기 시작하면 이 루프 차량의 배차를 중앙제어컴퓨터에 요구하며, 중앙제어컴퓨터는 해당 역사에 가장 가깝게 위치한 루프 차량에게 역사로 진입하라는 지시를 한다.

2.3 사용자 입력

그림 2의 'Read_System_Data' 는 사용자 입력을 수행하는 프로시저이다. 사용자 입력은 그림 11과 같은 텍스트 파일에 해당 데이터를 지정한 위치에 입력함으로써 수행된다.

```
1 // case 상암, 2016
2 // 버스노선 폐지
3// Acc   Dcc   Jerk    Max_fail_dcc   Min_emer_dcc  Car_length begin_parameter
4// 2.5    -2.5   1.25    -4.0            -5.0            2.5
5// Time     boarding unboarding move begin_simulation_data
6// 3900     10          10           5
7// Node     X      Y    berth_no   icar_no  boarding_berth begin_station_data
8// 0        1500   830    5         5         2
9// 1        1700   1130   5         5         2
10// 2       1400   1130   5         5         2
11// ----------이 하 생 략 ------------------------
12// -999
13// NodeType      X      Y      begin_node_data
14// 12        n       0      200
15// 13        d       640    200
16// 14        m       760    200
17// 15        m       1200   200
18// 16        d       1340   200
19// ---------- 이 하 생 략 ----------------------
20// -999
21// from      to      limit    begin_track_data
22// 56        0       30
23// 0         57      60
24// 32        1       30
25// 1         33      60
26// 34        2       60
27// 2         35      30
28// ---- 이 하 생 략 -------
29// -999
30// from      to      num/hour     begin_od_data
31// 1         2           125
32// 1         3           158
33// 2         1           | 10
34// 2         3           240
35// 2         4           298
36// --- 이 하 생 략 -------
37// -999
```

그림 11 사용자 입력 파일(서울 상암DMC 사례)
Fig. 11 User input file(Case study Sangam DMC)

1, 2 번째 줄은 주석으로 결과물 출력 시 상단에 그대로 출력되며 프로그램 수행에는 영향을 주지 않는다. 4 번째 줄에는 시스템 차량의 재원을 입력한다. 여기에는 최대 가속도, 최대 감속도, 최대 저크, 최대 사고감속도, 최대 비상감속도, 차량길이가 포함된다. 6번째 줄에는 시뮬레이션시간, 승차타이머 값, 하차타이머 값 및 승강장 간 차량이동시간을 나타내는 이동타이머 값을 입력한다. 8번째 줄부터는 각 역사별로 역사의 좌표 값, 승강장 수, 초기 배차되는 차량 수, 승차용 승강장 수를 넣는다. 14줄 째부터는 역사 외 각 노드의 타입 및 위치, 22번째 줄부터는 각 트랙의 시작 노드, 끝 노드 및 트랙의 제한속도를 입력하며 31번째 줄부터는 수송수요 데이터를 입력한다.

2.4 응용 사례

서울 상암DMC 지역을 가상의 시험노선으로 선정하고 시뮬레이션을 수행하여 시험노선의 수송능력을 분석하였다. 분석의 목적은 수송수요를 만족하는 역사규모 및 소요차량 수를 도출하는 것이다. 수송 수요를 만족한다는 판단의 척도는 모든 역사에서 승객의 평균대기시간이 1분 미만일 경우로 가정하였다. 차량의 재원 및 시뮬레이션 조건 부분은 2.3항의 사용자 입력 부분과 같고 노선 데이터 및 수송수요는 상암 지역을 가정하였다. 그림 12는 시험노선 및 역사위치를 나타내고 표 5는 시간당 수송 수요를 나타낸다. 맨 처음 각 역사에 승강장 3개와 초기 차량을 3대 배치하고 시뮬레이션을 수행하였다. 그림 13은 시뮬레이션 분석결과이며 각 역사의 승강장 수, 총 차량 대수 및 각 역사별 승객 평균대기시간을 보여준다. 분석 결과 총 차량대수는 36대이며 6번역사에서 승객 평균대기시간이 622초이므로 각 역사별로 승강장 수와 초기 배치차량 수를 5대로 증가하였다. 분석결과는 그림 14와 같으며 총 차량대수는 60대이고 최대승객평균대기시간은 4번역사에서 192초이다. 역시 기준 값 60초보다 크므로 10대의 루프 차량을 추가 배치하여 다시 시뮬레이션을 수행하였다. 결과는 그림 15와 같으며 총 차량대수가 70대이고 평균승객대기시간은 9번역사에서 27초로 분석되었다. 표 5는 각 케이스별 시뮬레이션 결과의 요약이다. 물론 이 결과는 최적화된 결과가 아닐 수 있다.

표 5 차량소요대수 및 역사규모 분석결과
Table 5 Result for number of vehicles and station scale analysis

구분	총 차량 소요대수	최대승객 평균대기시간	역사규모
각 역사 3대 차량 배치	36	622	승강장 3개
각 역사 5대 차량 배치	60	192	승강장 5개
루프 차량 10대 추가 배치	70	27	승강장 5개

그림 12 시험노선(서울 상암DMC 지역)

Fig. 12 Test Line(Sangam DMC)

표 6 PRT 수송 수요 OD(서울 상암DMC 지역)

Table 6 PRT transportation demand OD(Sangam DMC)

PRT 수송수요 (2021년), 통행량/hr														
역사	101	102	103	104	105	106	107	108	109	110	111	100	계
101	-	7	8	137	1	0	0	0	0	0	1	4	159
102	0	-	13	16	23	8	0		0	1	3	3	68
103	1	0	-	9	16	6	3	4	3	3	4	1	50
104	107	0	0	-	14	6	6	4	3	46	28	3	217
105	0	2	0	0	-	3	6	8	5	0	23	64	111
106	0	1	0	0	-	-	10	13	7	0	0	5	36
107	0	1	1	1	0	-	-	10	16	0	0	3	32
108	8	2	8	4	2	0	-	-	4	0	0	0	28
109	30	11	31	20	9	1	0	-	-	0	0	0	101
110	1	0	0	1	0	-	11	22	2	-	0	0	38
111	1	0	0	0	-	-	4	2	2	53	-	20	82
100	65	13	1	1	4	2	0	0	-	12	1	-	99
계	213	37	63	188	69	25	41	62	41	116	61	103	1019

```
// case 상암_2021
//
   * 최대가속도= 2.5 [m/s2]
   * 최대감속도= -2.5 [m/s2]
   * 평균 탑승 인원= 1

   * 역사 별 차량 및 승강장 수
// ================================================
ID      승강장 수   차량 수
 0          3          3
 1          3          3
 2          3          3
 3          3          3
 4          3          3
 5          3          3
 6          3          3
 7          3          3
 8          3          3
 9          3          3
10          3          3
11          3          3
  * 총 차량 수= 36

   *역사 Data
<================================================>
                                    평균       평균       평균       평균
      승객       승객       승객     대기       대기       대기       대기
역    입장       출발       도착     승객       시간       차량       시간
 0       96         74         71     10.0      395.9        0.0        0.0
 1      159        157        164      2.0       56.2        0.0        0.0
 2       66         40         28     13.0      793.5        0.0        0.0
 3       48         44         47      1.0       97.4        0.0        0.0
 4      217        172        169     24.0      408.7        0.0        0.0
 5      110         85         51     13.0      449.2        0.0        0.0
 6       35         21         18      6.0      621.9        0.0        0.0
 7       28         27         32      0.0       51.9        0.0        0.0
 8       28         27         48      0.0       61.9        0.0        0.0
 9      101         83         33      8.0      313.3        0.0        0.0
10       37         35         79      0.0       42.5        0.0        0.0
11       81         49         44     14.0      591.9        0.0        0.0
```

그림 13 시뮬레이션 결과(상암DMC, 차량 36량)

Fig. 13 Simulation result(Sangam DMC, Vehicle : 36)

```
// case 상암_2021
//
   * 최대가속도= 2.5 [m/s2]
   * 최대감속도= -2.5 [m/s2]
   * 평균 탑승 인원= 1

   * 역사 별 차량 및 승강장 수
// ================================================
ID      승강장 수   차량 수
 0          5          5
 1          5          5
 2          5          5
 3          5          5
 4          5          5
 5          5          5
 6          5          5
 7          5          5
 8          5          5
 9          5          5
10          5          5
11          5          5
  * 총 차량 수= 60

   *역사 Data
<================================================>
                                    평균       평균       평균       평균
      승객       승객       승객     대기       대기       대기       대기
역    입장       출발       도착     승객       시간       차량       시간
 0       96         96         95      0.0       14.2        0.0        0.0
 1      159        155        192      0.0       22.2        0.0        0.0
 2       66         66         34      0.0       12.3        0.0        0.0
 3       48         48         57      0.0        1.7        0.0        0.0
 4      217        200        179     11.0      191.5        0.0        0.0
 5      110        108         64      1.0       42.7        0.0        0.0
 6       35         33         22      0.0        3.8        0.0        0.0
 7       28         28         37      0.0        1.0        0.0        0.0
 8       28         28         58      0.0        1.1        0.0        0.0
 9      101         99         37      1.0       62.5        0.0        0.0
10       37         35        103      0.0       27.3        0.0        0.0
11       81         75         55      2.0       92.4        0.0        0.0
```

그림 14 시뮬레이션 결과(상암DMC, 차량 60량)

Fig. 14 Simulation result(Sangam DMC, Vehicle : 60)

```
// case 상암_2021
//
   * 최대가속도= 2.5 [m/s2]
   * 최대감속도= -2.5 [m/s2]
   * 평균 탑승 인원= 1

   * 역사 별 차량 및 승강장 수
// ================================================
ID      승강장 수   차량 수
 0          5          5
 1          5          5
 2          5          5
 3          5          5
 4          5          5
 5          5          5
 6          5          5
 7          5          5
 8          5          5
 9          5          5
10          5          5
11          5          5
  * 총 차량 수= 70

   *역사 Data
<================================================>
                                    평균       평균       평균       평균
      승객       승객       승객     대기       대기       대기       대기
역    입장       출발       도착     승객       시간       차량       시간
 0       96         96         97      0.0        3.0        0.0        0.0
 1      159        158        202      0.0       10.2        0.0        0.0
 2       66         66         34      0.0        4.2        0.0        0.0
 3       48         48         57      0.0        1.7        0.0        0.0
 4      217        216        181      0.0        6.1        0.0        0.0
 5      110        109         66      0.0       16.3        0.0        0.0
 6       35         35         23      0.0        1.5        0.0        0.0
 7       28         28         37      0.0        1.0        1.0        0.0
 8       28         28         58      0.0        1.1        0.0        0.0
 9      101        101         38      0.0       26.9        0.0        0.0
10       37         37        110      0.0        4.0        0.0        0.0
11       81         80         58      0.0       14.7        0.0        0.0
```

그림 15 시뮬레이션 결과(상암DMC, 차량 70량)

Fig. 15 Simulation result(Sangam DMC, Vehicle : 70)

3. 결 론

PRT 운행제어 시뮬레이터의 각 구성시스템 별로 운영 알고리즘이 제시되었다. 또한 구성시스템의 규모를 변경하면서 수송능력 향상을 이루는 방안이 응용 예에서 간략히 제시되었다. PRT 시스템은 신 교통 시스템이므로 운영 경험도 실적도 없다. 본 시뮬레이터는 응용 예에서 보이듯이 시스템의 기본 설계 시 요구되는 주요 설계요소 즉 역사 위치 및 규모, 노선 선정, 소요차량대수 판단을 위한 분석 툴로 이용될 수 있다. 시스템의 하드웨어 및 인프라 못지 않게 공차관리 등의 소프트웨어적 운행 알고리즘도 수송처리 능력을 크게 좌우한다. 본문에서는 기본적인 공차관리, 루프 차량의 관리 등에 대해 논의하였지만 최적화는 논의되지 않았다. 공차관리의 최적화 알고리즘은 전체 시스템 수송능력을 크게 좌우하므로 향후 지속적으로 개발되어야 될 것으로 사료된다.

참 고 문 헌

[1] 정상기, 정락교, 김백현, “비동기식 PRT 차량의 주행제어 알고리즘”, 대한전기학회 논문집 제60권 제1호, 2011.1.

[2] Jack H Irving, Harry Bernstein, C. L. Olson, Jon Buyan “Fundamentals of Personal Rapid Transit” Lexington Books, 1997.

[3] Markus Theodor Szillat ″A low level PRT Microsimulation″ Dissertation, University of Bristol, 2001.

저 자 소 개

정 락 교 (鄭 樂 敎)

1991년 2월 : 인하대학교 전기공학과(공학사), 1999년 8월 : 인하대학교 전기공학과(공학석사), 2005년 2월 : 인하대학교 전기공학과(공학박사), 1990년 12월～1994년 12월 : 한진중공업 사원, 1995년 1월～현재 : 한국철도기술연구원 수요응답형교통연구단 단장(책임연구원)

Tel : 031-460-5725

Fax : 031-460-5036

E-mail : rgjeong@krri.re.kr

김 백 현 (金 伯 鉉)

1994년 2월 : 인하대학교 전자공학과(공학사), 1996년 2월 : 인하대학교 전자공학과(공학석사), 2003년 2월 : 인하대학교 전자공학과(공학박사), 2003년 3월～현재 : 한국철도기술연구원 수요응답형교통연구단 선임연구원

Tel : 031-460-5443

Fax : 031-460-5036

E-mail : bhkim@krri.re.kr

황 현 철 (黃 鉉 喆)

1997년 2월 : 인하대학교 전자공학과(공학사), 1999년 2월 : 인하대학교 전자공학과(공학석사), 2006년 2월 : 인하대학교 전자공학과(공학박사), 2006년 8월～현재 : 한국철도기술연구원 무가선트램연구단 선임연구원

Tel : 031-460-5747

Fax : 031-460-5036

E-mail : hchwang@krri.re.kr