반도체디스플레이기술학회지 제17권 제3호(2018년 9월)
Journal of the Semiconductor & Display Technology, Vol. 17, No. 3. September 2018.

드론 협업 지속을 위한 프레임워크 연구

김강주*·박용범*†
*†단국대학교 컴퓨터학과

Study on Framework for Continuing Drone Collaboration

Kang-Ju Kim* and Young B. Park*†
*†Dankook University, Dept. of Computer Eng

ABSTRACT

The drone has the restrictions on the controls, the battery and the surrounding environment in performing missions such as fire extinguishing. This restriction can improve the limitations that leave the leader can be monitored. The existing method of constructing the leader based on the GPS is highly dependent on the signal and is vulnerable to hardware defects. In this paper, we solve these problems with dynamic leaders decision. Drones can use their leader drones rather than remote controls. Information about the drones changes depending on the surrounding environment by replacing the leader with a dead battery or electing leader by the drones themselves without human intervention. This suggests that the leader monitors the community through a framework for continuing the drones collaboration and that the community can collaborate to overcome the limitations and continue the mission. The analysis of the proposed system through simulation experiments confirm that it has a better task performance. By using this system, it is possible to continue the mission and solve problems that are vulnerable to hardware defects.

Key Words : drone, collaboration, restrictions, mission, leader

1. 서 론

무인항공기인 드론이 상용됨에 따라 상업적, 군사적, 산업적인 방면으로 기술 개발이 이루어지고 있다[1]. 이로 인하여 지속적으로 드론을 이용한 기술들이 생겨 드론에 관심도가 증가하고 있다. 그동안 드론을 조종하는 조종사가 필요하였는데 드론과 컴퓨터가 통신하여 자율적으로 주행하는 드론이 개발되었다. 대표적인 예가 ArduPilot이다. ArduPilot은 드론의 하드웨어 부분 초기화 및 드론의 임무가 주어졌을 때 해당 임무를 업로드 할 수 있는 시스템이다[2]. 이 시스템은 개발을 하는 드론을 보다 편리성과 신뢰성을 고려하여 안정화하는 방법을 제안 및 구현하였다.

MAVLink는 ArduPilot으로 만들어진 펌웨어를 업로드하고 이 드론의 데이터를 Ground Control Station(GCS)이나 드론으로 보낼 수 있는 프로토콜이다[2, 3]. 드론에는 화재 진압 등 임무를 수행함에 있어 조종 장치, 배터리, 주변 환경에 대한 다음과 같은 제약이 있다[1, 5, 7, 16].

조종 장치 한 개로 여러 대의 드론을 조종하기에는 비효율적이다. 드론의 크기와 무게 때문에 배터리 용량에 대한 제약이 있기 때문에 전력을 어느 방향으로 움직일 때 적게 쓰는지에 대한 연구가 있다[5]. 주변 환경에 따라 드론 상태에 대한 정보가 변한다[7]. 이를 해결하기 위해 기존 연구된 논문의 문제는 다음과 같다.

Dietrich, Thomas, et al.[4]은 MAV(Micro Air Vehicles) 그룹과 군집 관리 및 MAV-MAV와 MAV-toGCS 통신을 위한 새로운 메시지와 데이터 구조를 정의하는 통신 라이브러리 MAVLink의 확장을 제안하였다. MAVLink에 내용을 추가함으로써 MAVLink를 사용하는 장비에서는 모두 사용할 수 있다고 주장하였다. 하지만 MAVLink 내용만 추가하여 실제 데이터 값에 대한 정보가 뜨지만 장비별로 일일이 정보를 입력해야 하고 표준화가 필요하다.

†E-mail: ybpark@dankook.ac.kr

Dietrich, Thomas, Silvia Krung, and Armin Zimmermann[5]은 에너지 관리 및 다중 UAV를 포함한 노드 교체 전략에 초점을 맞춘 모바일 로봇 시스템을 위한 새로운 시뮬레이션 프레임 워크를 제시하였다. 이 프레임 워크의 목표는 임무 일정 계획 전략 및 관리 프로세스를 평가하는 것이다. 시뮬레이션 설정의 비용, 효율성 및 안정성을 특성화한 수치적인 성능 결과가 된다. 이러한 결과를 다양한 조합 및 매개 변수 값과 비교하여 유지 관리 전략 권장 사항을 정의할 수 있다. 하지만 구현된 동작 모델 및 유지관리 전략을 기반으로 UAV동작을 시뮬레이션하고 미션 시나리오를 정의할 수 있어야한다.

Koubaa, Anis, and B. Quershi[6]은 클라우드 로봇 플랫폼을 통한 실시간 물체 추적에 대한 성능 평가 연구를 수행하였다. 이 논문에서 제안한 DroneTrack 시스템은 무인 항공기, 클라우드 및 사용자 간 원활한 연결을 통해 대상과 무인 항공기 사이의 통신 범위 제한에 관계없이 언제 어디서나 이동 목표를 추적하는데 안정적으로 사용될 수 있다. 하지만 관련된 프로세스의 최적화, 클라우드의 처리 지연 감소, 네트워크 지연 및 보다 빠른 속도 및 가속도에 대한 무인 항공기의 응답성을 필요로 하고 안정적으로 유지되려면 GPS위치 확인 파라미터의 정확성, 이동하는 물체의 새로운 위치의 업데이트 빈도, 사용자와 클라우드 간의 통신 품질이 필요로 한다.

Smigielski, Piotr, Mateusz Raczynski, and Lukasz Gosek[7]은 시뮬레이션이 항상 고급 로봇 시스템 개발에 중요하다고 하였다. 또한 중요한 핵심 솔루션 중 하나는 로봇 솔루션 비용이다. 비전 시스템과 네비게이션 알고리즘과 같은 시스템 요소의 초기 버전을 테스트하기 위해 시뮬레이션 환경을 활용하면 값비싼 사고를 피할 수 있다고 주장하였다. 이러한 어플리케이션에서 로봇은 장애물을 피하는 것뿐만 아니라 주어진 작업을 수행하고 정확하게 자리잡기 위해 목표 위치 계산 및 경로 계획이 필요하다.

Limbu, Narendra, et al.[8]은 GPS 기반의 지역화만을 사용하는 Leader-Follow 구성에서 협력적인 야외 기동을 위한 분산 컨트롤러의 개발 및 구현하였다. 하지만 GPS 기반으로 Leader-Follow 수행 알고리즘을 사용하여 GPS에 대한 의존성이 높다.

본 논문에서는 이러한 문제들을 제안한 드론 협업 지속을 위한 프레임워크로 해결할 수 있다고 예상한다. 멀리 떨어진 조종 장치보다 가까운 리더 드론을 활용하여 해결할 수 있다는 것을 보여주었다. 리더가 되는 드론은 성능이 다른 드론에 비해서 좋다고 가정하였다. 배터리 시간이 다된 리더를 교체하거나 사람이 개입하지 않고 드론 군집이 스스로 행동함으로써 주변 환경에 대한 제약을 줄일 수 있다. 드론이 군집을 이루어 각각의 리더를 통해 모니터링함으로써 협업으로 한계점을 극복하고 임무 수행을 지속할 수 있다고 제안한다. 2장에서는 리더 결정 알고리즘에 대해 설명한다. 3장에서는 협업 시스템 구성에 대해서 다룬다. 4장에서는 드론 협업 지속을 위한 프레임 워크를 실험 및 분석한다. 5장은 결론으로 끝을 맺는다.

2. 관련 연구

분산 프로세스에서의 대표적인 리더 결정 알고리즘에는 Bully Election Algorithm과 Token Ring Election Algorithm, Chang and Roberts Election Algorithm이 있다.

2.1 Bully Election Algorithm

Bully 선출 알고리즘은 동적으로 리더나 후보자를 선출하는 방법이다. 메시지 형태로 설명을 하면 Process 6이 역할이 상실되면 Process 3은 Process(n>3)에 리더를 선출한다고 알리는데 Process 6은 반응을 보이지 않는다. 하지만 Process 4와 Process 5는 Okay 사인을 보낸다. Process 4는 선출 메시지를 Process 5와 Process 6에 보낸다. Process 6은 상실되어 반응이 안 오고 Process 5만 반응이 온다. Process 5는 선출 메시지를 Process 6에 보낸다. Process 6이 반응이 없으면 Process 5가 자신이 리더라고 선언한다[20]. 표 1 Bully 선출 알고리즘은 메시지 전달을 전제로 항상 리더의 활동을 점검할 수 있다. 하지만 프로세스가 충돌하여 동일한 번호의 프로세스로 교체된 경우 안전 조건을 충족시키는 것이 보장되지 않는다. 충돌한 프로세스를 대체하는 프로세스는 다른 프로세스가 가장 높은 번호를 가진 것으로 결정한 것처럼 가장 높은 번호를 가진 프로세스를 결정할 수 있다. 이럴 때 두 개 프로세스가 동시에 리더가 된다. 메시지 전달 순서에 대한 보장이 없고 이러한 메시지의 수신자는 리더 프로세스가 서로 다른 결론을 내릴 수 있다. 이는 프로세스를 신뢰할 수 없다는 것을 의미한다. 이를 향상시키기 위한 Bully 알고리즘 연구가 진행되고 있다[9, 10].

Table 1. Bully Election Algorithm

Process	Connected Process	Leader
P1	P2, P3, P4, P5, P6	
P2	P1, P3, P4, P5, P6	
P3	P1, P2, P4, P5, P6	
P4	P1, P2, P3, P5, P6	
P5	P1, P2, P3, P4, P6	
P6	P1, P2, P3, P4, P5	O

2.2 Token Ring Election Algorithm

Token Ring 선출 알고리즘은 여러 프로세스들이 하나의 링에 이어져 형성되며, 데이터는 항상 한 방향으로만 흐른다. 메시지 형태로 설명하면 Process 6이 역할이 상실되면 Process 3은 Process 6이 반응을 보이지 않아 다음 노드인 Process 5에게 선출 메시지를 보낸다. Process 5를 지나면서 Process 3의 식별자를 선출 메시지에 추가한다. Process 1을 지나면서 Process 5의 식별자를 선출 메시지에 추가한다. 또한 Process 2을 지나면서 Process 1의 식별자를 선출 메시지에 추가한다. 마찬가지 Process 4를 지나면서 Process 2의 식별자를 추가하고 Process 3이 해당 메시지를 받으면서 식별자 중 가장 큰 수의 식별자를 고른다. Process 3은 Process 5가 리더라고 링 주변에 동료 메시지를 보낸다. Process들은 동료 메시지를 받게 된다. 이로서 리더가 변경된다[22]. 표 2 Token Ring 선출 알고리즘은 해당 프로세스 옆에 있는 프로세스로 메시지를 전달하여 다른 알고리즘에 비해 간단하다. 다른 프로세스와 더 이상 통신하지 않으면 정보를 다음 프로세스로 전달하여 트래픽을 전달하는 메시지에 유용하다. 하지만 메시지 전달에 많은 시간을 소비한다. 정보 메시지를 전달하고 받아야한다. 모든 정보를 얻는데 시간이 걸린다. 리더를 선출한 후에 리더 선출 메시지는 링의 모든 프로세스를 통과해야 한다. 리더 선출 메시지는 모든 프로세스로 전달되지않지만 하나의 다른 프로세스로 전달된다. Token Ring 선출 알고리즘은 링 주위를 돌아다니는 정보나 리더 선출 메시지가 손실되거나 놓칠 수 있다. 이를 보완하기 위한 연구가 진행되고 있다[11, 12].

Table 2. Token Ring Election Algorithm

Process	Connected Process	Leader
P1	P2, P5	
P2	P1, P4	
P3	P4, P5, P6	
P4	P2, P3	
P5	P1, P3, P6	
P6	P3, P5	O

2.3 Chang and Roberts Election Algorithm

Chang and Roberts 선출 알고리즘은 Ring형 알고리즘에서 파생한 알고리즘으로 각 프로세스가 UID(Unique Identification)을 가지고 각 프로세스가 시계 방향으로 통신하는 단방향 링으로 정렬되어 있다고 가정한다. 지도자가 없다고 알리는 과정으로 선출을 시작한다. UID를 포함하는 선출 메시지를 작성한다. 그런 다음 메시지를 시계방향 이웃 프로세스에 보낸다. 프로세스가 선출 메시지를 보내거나 전달할 때마다 프로세스 자체도 참여자로 표시된다. 프로세스가 선출 메시지를 받으면 메시지 UID와 자신 UID를 비교한다. 선출 메시지 UID가 더 크면 프로세스는 다시 시계방향으로 선출 메시지를 전달한다. 선출 메시지 UID가 더 작고 프로세스가 아직 참여자가 아닌 경우 프로세스는 메시지의 UID를 자신의 UID로 바꾸고 업데이트된 선출 메시지를 시계 방향으로 보낸다. 선출 메시지 UID가 더 작고 프로세스가 아직 참여자가 아닌 경우 프로세스는 메시지의 UID를 자신의 UID로 바꾸고 업데이트된 선출 메시지를 시계방향으로 보낸다. 선출 메시지 UID가 더 작고 프로세스가 이미 참여자일 때 프로세스는 선출 메시지를 무시한다. 들어오는 선출 메시지 UID가 프로세스 UID가 프로세스 UID와 같으면 프로세스가 리더 역할을 시작한다. 프로세스가 리더 역할을 시작하면 리더 프로세스는 자신을 비참여자로 표시하고 선출과 UID를 알리는 이웃에게 선출완료 메시지를 보낸다. 선출완료 메시지를 프로세스가 받으면 자신을 비참여자로 표시하고 선출완료 UID를 기록하며, 선출완료 메시지를 변경되지 않은 채 전달한다. 선출완료 메시지가 새로 선출된 리더에게 도착하면, 리더는 메시지를 삭제하고 선출이 끝난다. Chang and Roberts 선출 알고리즘은 안전성이 좋다[13]. 참여자와 비참여자를 사용하여 여러 프로세스가 거의 동시에 선출을 시작하면 한 명의 리더만 발표된다. 하이 레벨 알고리즘으로 표현하면 그림 1과 같다.

Algorithm 1 Chang and Roberts Algorithm

```
var state_p;
begin if p is initiator then
  begin state_p := cand ; send (tok, p) to Next_p;
   while state_p ≠ leader do
    begin receive (tok, q) ;
     if q = p then state_p := leader
     else if q < p then
       begin if state_p ≠ cand then state_p:=lost;
        send (tok, p) to Next_p
       end
    end
  end
 else while true do
  begin receive (tok, q) ; send (tok, q) to Next_p;
   if state_p = sleep then state_p:=lost
  end
end
```

Fig. 1. Chang and Roberts Algorithm.

2.4 Peterson Leader Election Algorithm

Peterson Leader 선출 알고리즘은 Ring형 알고리즘에서 파생한 알고리즘으로 Chang and Roberts 선출 알고리즘과 마찬가지로 각 프로세스가 UID(Unique Identification)을 가지고 각 프로세스가 시계 방향으로 통신하는 단방향 링으로 정렬되어 있다고 가정한다. 활성 모드에서 각 프로세스는 자신 프로세스 UID를 다음 프로세스로 보내고 이웃 프로세스로부터 첫번째 메시지를 수신할 때까지 대기한다. 그런 다음 수신된 메시지를 자신 프로세스 UID와 비교한다. UID가 일치하면 자신을 링의 리더로 알린다. 그렇지 않으면, 각 프로세스는 수신된 메시지를 링 내의 다음 이웃 프로세스로 전달하고 다음 메시지를 수신하기를 기다린다. 각 프로세스는 다음 메시지를 수신하면 처음 수신한 메시지와 새로운 수신된 메시지를 비교한다. 첫번째로 받은 메시지가 두 번째 메시지와 자신 프로세스 UID보다 큰 경우 자신 프로세스 UID를 첫번째로 받은 메시지로 업데이트한다. 그렇지 않으면 릴레이 모드로 들어간다. 릴레이 모드에서, 프로세스는 수신된 메시지에 대해 어떠한 추가 작업도 수행하지 않고 수신된 메시지를 링의 다음 이웃 프로세스로 전달한다. Peterson Leader 선출 알고리즘의 복잡성 연구가 진행되었다[14]. 이 알고리즘은 좀 더 복잡하지만 최악의 경우 메시지 복잡성이 낮다.

3. 협업 시스템 구성

3.1 협업시스템 소프트웨어 아키텍처

ArduPilot은 공개된 오픈소스로 일반적인 탐색에 사용되는 주변 센서, 하나이상의 마이크로 컨트롤러 또는 마이크로프로세스로 구성된 광범위한 내장형 하드웨어에서 실행된다. 시스템 구조는 그림 2와 같다. 드론은 현재 ArduPilot을 구성으로 사용하여 제어하는데 유용하다. 또한 OS가 있는 마이크로프로세스로 구성된 여러 하드웨어에서 커스터마이징이 가능하다. 그래서 ArduPilot으로 구성된 펌웨어를 채택하였다.

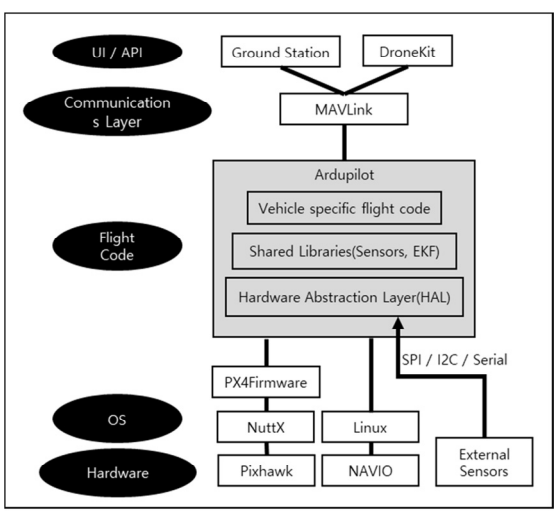

Fig. 2. Attitude control high-level diagram for each axis.

MAVLink는 ArduPilot이 업로드 된 드론의 정보를 전송하거나 미션을 업로드 할 수 있는 소형 비행체에 적합하도록 매우 가볍고, 헤더만 가지는 메시지 마샬링(mars halling)라이브러리이다[3]. MAVLink Packet 구조는 그림 3과 같다.

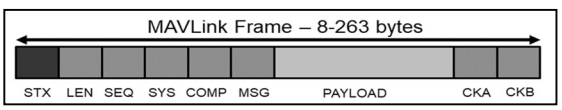

Fig. 3. MAVLink Packet Structure.

표 3은 MAVLink 패킷 안에 들어있는 내용이다. 데이터 부분이 따로 존재하고 보내는 System ID와 Component ID가 존재한다.

Table 3. MAVLink Packet Content

Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE(v0.9: 0x55)	Indicates the start of a new packet
1	Payload length	0 – 255	Indicates length of the following payload
2	Packet sequence	0 – 255	Each component counts up his send sequence
3	System ID	1 – 255	ID of the SENDING system
4	Component ID	0 – 255	ID of the SENDING component
5	Message ID	0 – 255	ID of the message
6 to (n+6)	Data	(0- 255) bytes	Data of the message, depends on M ID
(n+7) to (n+8)	Checksum(low byte, high byte)	ITU X.25/SAE AS-4 hash, excluding packet start sign, so bytes 1..(n+6)	

MAVLink는 멀티캐스트 디자인으로 이루어져 있으며 Topic모드(Publish-Subscribe)와 Point-to-Point 모드가 있다. Topic 모드는 위치, 자세 정보와 같이 모든 비행에 관련된 데이트스트림이 이용한다. Point-to-Point 모드는 타겟 ID와 타겟 컴포넌트를 사용하고 하위 프로토콜(Missions, Parameters, Commands)은 전송을 보장한다. 그림 4에서 보여주는 MAVLink 매개변수 구조체에 정의되어 있는 값을 이용하여 실제 드론이 어떻게 행동하고 어떤 오류가 있는지 알 수 있다. 다른 매개변수를 활용하면 어느 위치에 있는지도 파악할 수 있다.

```
MAVPACKED(
typedef struct __mavlink_sys_status_t {
 uint32_t onboard_control_sensors_present;
 /*Bitmap showing which onboard controllers and sensors are present.*/
 uint32_t onboard_control_sensors_enabled;
 /*Bitmap showing which onboard controllers and sensors are enabled*/
 uint32_t onboard_control_sensors_health;
 /*Bitmap showing which onboard controllers and sensors are operational or have an error*/
 uint16_t load;
 /*[d%] Maximum usage in percent of the mainloop time.*/
 uint16_t voltage_battery; /*[mV] Battery voltage*/
 int16_t current_battery; /*[cA] Battery current*/
 uint16_t drop_rate_comm;
 /*[c%] Communication drop rate, (UART, I2C, SPI, CAN), dropped packets on all links */
 uint16_t errors_comm;
 /*Communication errors (UART, I2C, SPI, CAN), dropped packets on all links */
 uint16_t errors_count1; /*< Autopilot-specific errors*/
 uint16_t errors_count2; /*< Autopilot-specific errors*/
 uint16_t errors_count3; /*< Autopilot-specific errors*/
 uint16_t errors_count4; /*< Autopilot-specific errors*/
 int8_t battery_remaining; /*< [%] Remaining battery energy*/
}) mavlink_sys_status_t;
```

Fig. 4. MSG_SYS_STATUS parameter defined.

3.2 드론 협업 지속을 위한 프레임 워크

제안한 드론 협업 지속을 위한 프레임 워크에서는 MAVLink를 추가하는 방법이 아닌 시스템 내에서 직접 그룹화를 진행하고 관리를 진행할 수 있도록 도와준다. 이로서 장비 별로 일일이 정보를 입력해야 하고 표준화가 필요한 점을 해결한다.

이 시스템을 사용함으로써 UAV동작을 시뮬레이션하고 미션 시나리오를 정의할 수 있다. 프로세스의 최적화, 클라우드의 처리 지연 감소, 네트워크 지연 등은 시스템 내에 존재하는 리더 결정 알고리즘으로 해결할 수 있다.

GPS위치 확인 파라미터의 정확성, 이동하는 물체의 새로운 위치의 업데이트 빈도, 사용자와 클라우드 간의 통신 품질도 실제 파라미터를 분석하여 사용함으로써 해결한다.

목표 위치 계산 및 경로 계획도 실제 하드웨어와 소프트웨어 통신 방법인 미션 임무가 존재하는 프로토콜을 사용하여 통신함으로써 해결한다. GPS뿐만 아니라 각종 파라미터를 사용하여 해결함으로써 GPS에 대한 의존성을 줄일 수 있다.

드론의 그룹을 이루는 것을 표현하면 그림 5와 같은 형태로 보여 질 수 있다. Leader 간 통신을 통해 각각의 군집이 임무를 제대로 수행하는지 여부를 확인할 수 있고 Leader가 한 대 존재하는 그룹 안의 상태는 Leader로서 상태점검을 통해 확인할 수 있다.

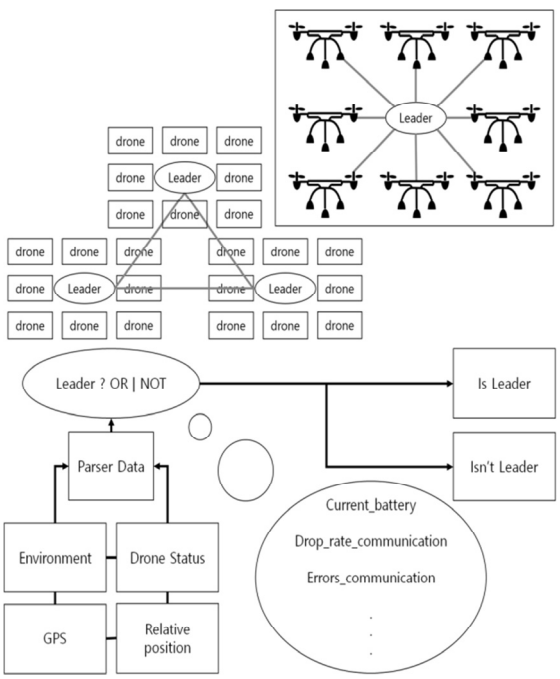

Fig. 5. Framework with leader decision algorithm.

3.3 리더 결정 알고리즘

앞에 설명한 드론 협업 지속을 위한 프레임워크에서 사용하는 리더 결정 알고리즘이다. 리더는 그룹화를 하고 그룹이 이루는 군집을 관리한다. MAVLink 프로토콜에 존재하는 데이터를 가지고 변화하는 드론의 데이터를 분석하여 협업을 지속하기 위해 리더를 바꾸어야 하는지 결정한다. 앞에서 운영체제에 존재하는 프로세스도 리더 결정 알고리즘을 사용하여 관리하는 것을 보여줬다. 각 팀 합의를 통해 결정하는 방법과 리더 단일로 의사를 결정하는 방법에 대한 연구도 있다[23]. 의사 결정 속도는 단일 리더 의사 결정이 좋지만 의사 결정 품질은 팀의 의견을 수렴하여 의사 결정하는 것이 우수하였다. 하지만 의사 결정 속도가 의사 결정권에 대한 개별 팀 구성원의 인식과 관련이 있고, 의사 결정 스타일은 더 나은 의사 결정과 관련이 없었다. 이에 따라 리더 결정 알고리즘이 드론 협업 지속을 위한 프레임 워크의 기반이 된다.

본 논문에서 제안한 모델은 MAVLink의 데이터를 바탕으로 리더 결정 알고리즘을 결합하여 드론 협업을 지속

Algorithm 2 Leader Decision Tree in HITL(Hardware In The Loop)

Input: Voltage
Initialize drone D
Drone external environment D_{ee}, Drone position status D_{ps}, Drone internal status D_{is}
Wind W, Drone relative position D_{rp}, Drone battery status D_{bs}
for Drone battery status $D_{bs} \in D_{ee}$ **do**
if D_{bs} is "scarced" OR other stopping criteria met **then**
terminate
end if
for Drone relative position $D_{rp} \in D_{ps}$ **do**
if D_{rp} is "insufficient" OR other stopping criteria met **then**
terminate
end if
for Wind $W \in D_{ee}$ **do**
if W is "strong" OR other stopping criteria met **then**
terminate
end if
end for
end if
end for
a decision tree for predicting class labels of target drone
end for
return

Fig. 6. Leader Decision Tree Algorithm in HITL.

하기 위한 시스템이다.

결정하기 위한 측정 방식은 총 3가지로 나눌 수 있다. 첫째 드론 간의 GPS 및 상대적 위치를 통해 리더를 결정하는 방법, 두번째 드론의 통신 에러 비율, 통신 결함 비율, 드론 배터리 잔량 등 드론 내부적인 요인에 따라 리더를 결정하는 방법, 세번째 드론의 임무수행에 적합한 외부요인에 따라 리더를 결정하는 방법이 있다.

먼저 드론 간의 GPS 및 상대적 위치를 통해 리더를 결정하는 방법은 드론이 크기와 무게 때문에 배터리 용량에 대한 한계가 있어 리더에 따라 이동하거나 메시지를 주고받을 때 배터리 절약할 수 있는 형태가 있다. 또한 이동함에 있어서 각도 및 이동방법에 따라 배터리 소요량이 달라진다[5]. 드론의 통신 에러 비율, 통신 결함 비율, 드론 배터리 잔량 등 드론 내부적인 요인에 따라 리더를 결정하는 방법은 드론의 성능을 기반으로 리더 역할을 제대로 수행할 수 있는지에 관련이 있다[18-21]. 드론이 자체 결함이 있는데 리더로 결정하게 되면 해당 역할을 수행하지 못해 협업의 지속력을 갖을 수 없다[22].

드론의 임무수행에 적합한 외부요인에 따라 리더를 결정하는 방법은 외부의 장애물, GPS가 수신이 안되는 지역과 같은 외부 환경에 따라 결정을 하는 것으로 제대로 수행할 수 있는 환경인지에 대한 피드백을 통해 리더를 결정할 수 있게 된다. 그림 6은 위에서 제안한 하드웨어기반의 리더 결정 알고리즘이다.

4. 실험 및 분석

본 논문에서는 위에 제안한 모델을 테스트하는 방법으로 시뮬레이션을 설정하였다. 소프트웨어로 테스트하는 것이 하드웨어 테스트에 비해 값비싼 사고를 피할 수 있는 것은 기존 연구된 바 있다[7, 15, 17].

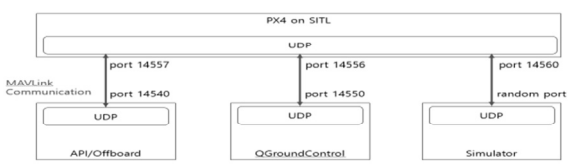

Fig. 7. Robot Operating System with Gazebo Simulation.

드론이 시뮬레이션 되는 형태는 다음 그림 7과 같다. QGroundControl은 Ground Control Station(GCS) 중 하나이다[24]. Simulation은 Gazebo라는 소프트웨어를 사용하였다[25]. SITL(Software In The Loop)는 하드웨어를 대신하여 소프트웨어로 구현한 것이다. 이 중 API/Offboard를 통한 실험, QGroundControl를 대신하여 구현한 Agent를 통한 실험을 진행하였다.

4.1 API/Offboard

API/Offboard로 통신을 하기 위해 MAVLink 수신되는 데이터를 분석하였다. API/Offboard의 경우 MAVLink 데이터가 아닌 Robot Operating System(ROS)전용 API를 주고받으며 현재 제공되는 API는 DroneCore와 MAVROS가 있다[26, 27].

```
struct EulerAngle {
    float roll_deg; /**< @brief Roll angle in degrees, positive is banking to the right. */
    float pitch_deg; /**< @brief Pitch angle in degrees, positive is pitching nose up. */
    float yaw_deg; /**< @brief Yaw angle in degrees, positive is clock-wise seen from above. */
};
```

Fig. 8. Telemetry of Drone API.

그림 8에 보이는 것과 같이 Offboard 모드에서도 데이터를 주고받을 수 있으나 API가 MAVLink에 비해 간단한 데이터들 위주로 착륙 및 이륙 등의 데이터만 가지고 판단하며 날개가 고정된 차량에서는 지원하지 않는다.

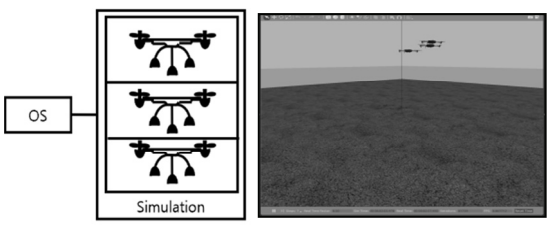

Fig. 9. Simulation using drone API.

드론 API를 이용하여 시뮬레이션으로 실험을 진행하였는데 좌표에 맞춰 이동하는 것을 볼 수 있다. 그림 9를 통해 Offboard API를 이용하여서도 리더를 결정할 수 있다는 것을 알 수 있다.

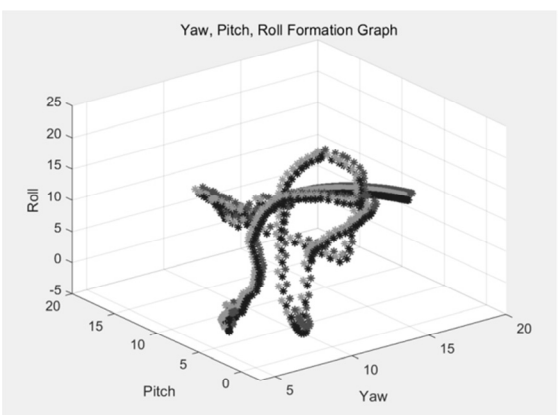

Fig. 10. Offboard Mode Formation Flight Result.

실험을 진행하면서 나온 정보를 바탕으로 Offboard모드에서 나오는 값으로만 시각화를 진행한 결과 그림 10과 같은 센서 모션 트래킹을 볼 수 있었다. Offboard모드 상태에서는 주로 yaw, pitch, roll 등 센서 값을 바탕으로 구성하여 외부환경 요인, 자체 결함으로 인한 리더 변환을 볼 수 없었다. 하지만 GPS에 의존하지 않아도 센서 값을 기반으로 편대를 구성할 수 있다. 또한 주행 위치에 따라 리더가 될 수도 있다. 비행계획에서 얼마나 효율적으로 전력 소비를 줄일 수 있는지에 따라 변한다. 리더가 자신의 데이터를 다른 드론에게 전송함으로써 위치를 제어할 수 있다. 비행계획에서 얼마나 효율적으로 전력 소비를 줄일 수 있는지에 따라 변한다. 리더가 자신의 데이터를 다른 드론에게 전송함으로써 위치를 제어할 수 있다.

4.2 IApilotAgent

IApilotAgent에서 MAVLink를 수신하는 방법은 다음 그림 11과 같다. IApilotAgent는 SITL(Software In The Loop)로부터 데이터를 받아 데이터를 분석한다. 해당 데이터는 MAVLink로 이루어져 있어 MAVLink를 해석할 수 있는 Parser로 읽을 수 있으나 해당 시스템에서는 Unpack을 통해 프로토콜 자체를 사용하였다.

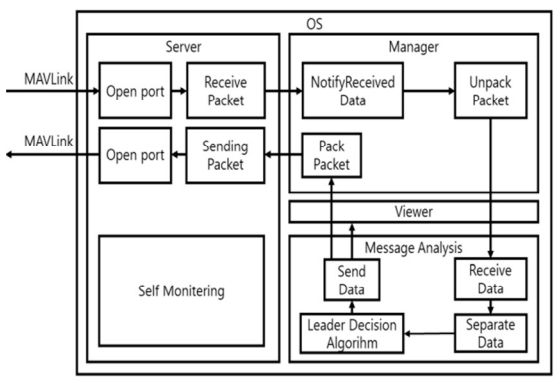

Fig. 11. IApilotAgent Single Structure.

Server에서는 다른 드론에서의 데이터를 받을 포트를 열고 패킷을 수신한다. 패킷을 수신한 것을 Manager가 알게 되면 패킷을 푸는 작업을 진행한다. Message Analysis에서는 해당 데이터를 각 메시지 별로 분리하는 작업을 진행한다. 이를 리더 결정 알고리즘에 넣어서 데이터를 보여주는 View로 전송을 하고 리더에 대한 정보를 포함한 데이터를 보내는 역할을 한다. Server에서는 자기 자신에 대한 모니터링을 통해 이러한 작업이 제대로 동작하고 있는지 확인한다. 그림 12는 군집화 하였을 때 작동되는 IApilotAgent구조이다.

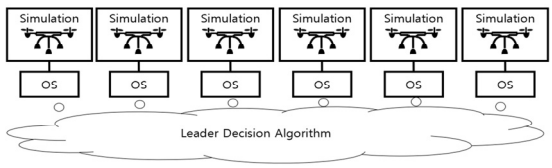

Fig. 12. IApilotAgent Multiple Structure.

자가 진단으로 자신이 리더가 될 수 있는지 판단하고 리더 상태가 악화되었을 때 다른 리더로 변경할 수 있게 한다. 해당 조건이 맞을 때는 그림 13과 같이 콘솔상으로 출력된다. 그림에서 리더로 사용 가능할 때에는 여러 조건이 충족했다는 정보로 출력되고, 리더로 불가능할 때에는 어떤 조건에서 불가능한 점에 대해 출력한다. 리더가 가능한 상태인 드론에게 직접 전송함으로써 모든 드론을 다 거쳐야하는 다른 리더 선택 알고리즘보다 복잡도를 줄일 수 있다.

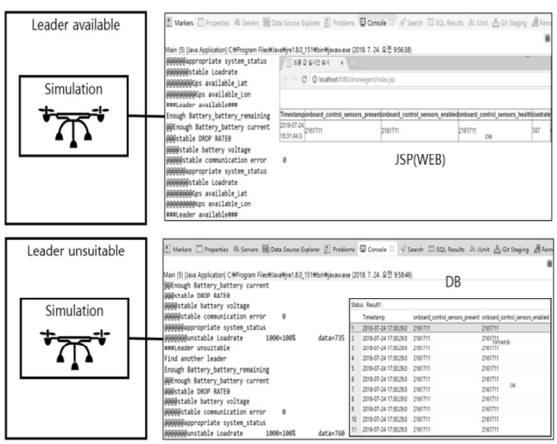

Fig. 13. IApilotAgent Leader Self Evaluation.

세부적인 내용을 DB로 저장하여 추후 문제가 발생했을 때 DB에 저장된 로그 파일로 확인할 수 있고 드론의 상태를 실시간 웹모니터링으로 보여주는 작업까지 가능하다. 임무에 맞춰진 비행을 마치고 나면 RTL(Return to Launch) 기능으로 실행했던 위치에 되돌아오게 된다. 이는 처음에 있던 위치로부터 호버링하고 있는 모습을 보여줄 수 있고 이 상태에서 Land 기능으로 호버링하고 있던 위치의 지상에 착륙하게 된다.

5. 결 론

본 논문에서는 드론 협업 지속을 위한 프레임워크를 제안하였으며, 드론 각각을 프로세스 및 운영체제라고 정의하고 드론의 임무를 Ground Station System(GCS)이 아닌 리더로 관리하고 상태 점검을 통해 협업이 지속될 수 있는 환경을 갖추었다. 또한 환경을 설계하여 실제 테스트를 진행할 수 있으며, MAVLink를 사용하여 작업하여 MAVLink가 지원되는 하드웨어에서도 동작할 수 있는 IApilotAgent를 설계하였다. IApilotAgent로 GPS 의존성, 전력 소비 등을 해결하여 더 나은 임무 수행능력을 가질 수 있다는 것을 확인하였다. 점점 발전하고 있는 IoT분야에서도 해당 모델을 사용할 수 있을 것이다.

본 논문에 사용한 방법이 추후 연구에 또 다른 리더 결정 알고리즘과 비교하여 더 나은 선택지인지 비교하는 연구에 사용할 예정이다. 또한 하드웨어적 제한이 걸린 드론이기 때문에 최적화 문제도 발생할 수 있다. 이를 해결하는 연구도 진행될 수 있다. 드론에 부착된 DB와 WEB VIEW를 원격으로 모니터링하는 장치도 구성하여 실시간으로 드론이 어디있는지 파악하는 연구도 진행할 예정이다.

감사의 글

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 지역신산업선도인력양성사업 성과임 (No.NRF-2016H1D5A1909989)

참고문헌

1. Watts, Adam C., Vincent G. Ambrosia, and Everett A. Hinkley. "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use." Remote Sensing 4.6 (2012): 1671-1692.
2. https://github.com/ArduPilot/ardupilot.
3. https://github.com/mavlink/mavlink.
4. Dietrich, Thomas, et al. "Towards a unified decentralized swarm management and maintenance coordination based on mavlink." Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on. IEEE, 2016.
5. Dietrich, Thomas, Silvia Krug, and Armin Zimmermann. "A discrete event simulation and evaluation framework for multi UAV system maintenance processes." Systems Engineering Symposium (ISSE), 2017 IEEE Interna tional. IEEE, 2017.
6. Koubâa, Anis, and B. Qureshi. "Dronetrack: Cloud-based real-time object tracking using unmanned aerial vehicles." IEEE Access (2018).
7. Śmigielski, Piotr, Mateusz Raczyński, and Łukasz Gosek. "Visual simulator for MavLink-protocol-based UAV, applied for search and analyze task." Computer

Science and Information Systems (FedCSIS), 2017 Federated Conference on. IEEE, 2017.
8. Limbu, Narendra, et al. "Outdoor co-operative control of multiple quadcopters using decentralized gps localisation." Robot Motion and Control (RoMoCo), 2015 10th International Workshop on. IEEE, 2015.
9. https://en.wikipedia.org/wiki/Bully_algorithm.
10. Arghavani, A., E. Ahmadi, and A. T. Haghighat. "Improved bully election algorithm in distributed systems." Information Technology and Multimedia (ICIM), 2011 International Conference on. IEEE, 2011.
11. https://en.wikipedia.org/wiki/Ring_network.
12. EffatParvar, MohammadReza, et al. "Improved algorithms for leader election in distributed systems." Computer engineering and technology (ICCET), 2010 2nd International Conference on. Vol. 2. IEEE, 2010.
13. Dolev, Danny, Maria Klawe, and Michael Rodeh. "An O (n log n) Unidirectional Distributed Algorithm." Journal of Algorithms 3 (1982): 245-260.
14. Chang, Ernest, and Rosemary Roberts. "An improved algorithm for decentralized extrema-finding in circular configurations of processes." Communications of the ACM 22.5 (1979): 281-283.
15. Modares, Jalil, Nicholas Mastronarde, and Karthik Dantu. "Ub-anc emulator: An emulation framework for multi-agent drone networks." Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization. ACM, 2016.
16. Schwarting, Wilko, Javier Alonso-Mora, and Daniela Rus. "Planning and Decision-Making for Autonomous Vehicles." Annual Review of Control, Robotics, and Autonomous Systems 1 (2018): 187-210.
17. Aminzadeh, Ali, Mohammadali Amiri Atashgah, and Alireza Roudbari. "Software in the loop framework for the performance assessment of a navigation and control system of an unmanned aerial vehicle." IEEE Aerospace and Electronic Systems Magazine 33.1 (2018): 50-57.
18. Bounceur, Ahcene, et al. "A new dominating tree routing algorithm for efficient leader election in IoT networks." Consumer Communications & Networking Conference (CCNC), 2018 15th IEEE Annual. IEEE, 2018.
19. Wang, Rui, et al. "Distributed Consensus Based Algorithm for Economic Dispatch in a Microgrid." IEEE Transactions on Smart Grid (2018).
20. Garcia-Molina, Hector. "Elections in a distributed computing system." IEEE transactions on Computers 1 (1982): 48-59.
21. Guo, Meng, Michael M. Zavlanos, and Dimos V. Dimarogonas. "Controlling the relative agent motion in multi-agent formation stabilization." IEEE Transactions on Automatic Control 59.3 (2014): 820-826.
22. Mazeh, Hussein, Majd Saied, and Clovis Francis. "Development of a Multirotor-Based System for Air Quality Monitoring."
23. Yang, Maria C. "Consensus and single leader decision-making in teams using structured design methods." Design Studies 31.4 (2010): 345-362.
24. https://github.com/mavlink/qgroundcontrol.
25. http://gazebosim.org.
26. https://github.com/dronecore/DroneCore.
27. https://github.com/mavlink/mavros.

접수일: 2018년 7월 31일, 심사일: 2018년 9월 15일, 게재확정일: 2018년 9월 19일