K - 토큰 링 상호배제 알고리즘

김용현, 김재훈
아주대학교

twilight8@ajou.ac.kr, jaikim@ajou.ac.kr

K - Token Ring Mutual Exclusion Algorithm

Kim Yong Hyun, Kim Jai-Hoon
Ajou University

요 약

본 논문에서는 상호배제(Mutual Exclusion) 알고리즘의 두 가지 알고리즘(권한 기반 접근법과 토큰 기반 솔루션) 중에서 토큰 기반 솔루션에 대한 문제점을 파악하고, 이를 해결하기 위한 k-토큰 링 알고리즘을 제안하였다. k-토큰 링 알고리즘은 k 개의 토큰이 링 형태의 네트워크를 순환하면서 프로세스가 리소스에 접근 권한을 획득할 수 있게 한다. 제안 알고리즘은 기존 토큰 링 방식에 비해 자원 사용의 지연 시간을 줄일 수 있으며, k-상호배제 알고리즘에 적용할 수 있다.

Ⅰ. 서 론

다양한 분야에서 데이터를 효율적이고 정확하게 처리하기 위해 분산 시스템이 적용되고 있다. 그 중에서도 프로세스가 리소스를 사용하기 위해 동시에 접근하는 것을 방기하기 위한 상호배제(mutual exclusion) 알고리즘에서 토큰사용을 기반한 알고리즘에 대한 연구가 많이 진행되었다. 우리는 토큰 기반 알고리즘에서도 노드가 링 형태로 구성되어 있어 토큰이 돌아가면서 노드가 토큰을 사용하는 토큰 링 알고리즘에 대해 연구하였다.

토큰 링 기반의 알고리즘은 노드가 순차적으로 링 형태로 구성되어 있는 환경에서 토큰을 가지고 있는 노드가 임계영역에 접근할 수 있는 권한을 가지게 된다. 임계영역에 접근한 노드는 리소스를 사용할 수 있고 리소스 사용을 마치면 임계영역에서 나와 토큰을 다음 노드로 전달한다. 토큰을 사용하지 않는다면 바로 다음 노드에게 토큰을 전달하게 된다.

본 연구에서 제시하고자 하는 k-토큰 링 알고리즘은 토큰사이의 새로운 k 개의 토큰이 동시에 임계구역(Critical Section)에 입장하여 상호배제를 달성하기 위한 방법을 제시한다.

Ⅱ. 관련연구

상호배제에서 토큰 기반 알고리즘은 대표적으로 Ricart 의 토큰 기반 알고리즘[1]은 임계영역(Critical Section)에 들어갈 수 있는 권한의 수단으로 토큰을 선택해 토큰을 가진 프로세스가 임계영역에 들어가 리소스를 사용할 수 있다. Suzuki 와 Kasami 알고리즘[2]은 Privilege 라는 메시지를 사용하여 임계영역(Critical Section)에 진입하기 위한 특권을 전송한다. Singhal 알고리즘[3]은 Suzuki 와 Kasami 의 개선된 알고리즘으로써 각 노드는 시스템 정보를 저장하는 두 개의 상태 벡터를 가지고 있어 각 노드의 메시지 정보나 토큰을 받을 수 있는 상태 등을 관리한다.

Ⅲ. k-토큰 링 알고리즘

k -토큰 링 알고리즘은 기존 토큰 링 알고리즘에서 토큰이 k개 있을 때 상호배제를 달성하기 위한 방법이다. 기존의 토큰 링 알고리즘은 1 개의 토큰이 있을 때 N개의 프로세스를 순환하기 때문에 방금 지나간 토큰을 다시 갖기 위해서는 $N-1$ 번의 토큰 순환을 거쳐야 임계구역에 들어갈 수 있다. 만약 N값이 크다면 토큰을 다시 받기 위해서는 오랜 시간이 걸릴 것이다. k-토큰 링 알고리즘은 이런 지연시간을 줄이고자 일정 프로세스 이상이 되면 토큰을 추가해 여러 개의 토큰으로 프로세스를 순환하는 시스템이다.

3.1 k-상호배제 알고리즘

S. Bulgannawar 의 분산 시스템 환경에서 k-상호배제 알고리즘[4]에서는 k 개의 노드가 임계 구역에 있어야한다. 여기서 제안된 알고리즘은 k토큰을 사용하여 이를 달성한다.

k -토큰 링 알고리즘에서 k 개의 토큰이 생성되고 프로세스는 토큰을 획득하면 임계 구역에 들어가기 위해 공유 리소스에 액세스해야 하는지 확인한다. k 개의 토큰을 동시에 프로세스가 획득하여 임계구역에 접근할 수 있다.

토큰을 획득한 k 개의 프로세스는 동작을 수행하고 수행이 끝나면 토큰을 다음 프로세스로 전달하게 된다. 기존의 토큰 링 알고리즘과 달리 k 개의 토큰이 동시에 임계구역에 접근할 수 있다. 또한, 각각의 토큰이 현재 토큰의 위치와 프로세스가 토큰을 획득하고 동작을 수행하고 다음 프로세스에게 토큰을 전달하는 시간을 로그 메시지 형태로 기록하여 토큰의 위치여부와 손상여부를 파악한다.

3.3 분산시스템의 상호배제 알고리즘

중앙집중 시스템은 1 개의 조정자가 여러 개의 프로세스를 관리하여 순차적으로 임계구역에 들어가기

위한 메시지를 보내면 조정자의 타임 스탬프가 낮은 순서대로 임계구역(CS)에 들어갈 수 있는 권한을 주지만, k-토큰 링 알고리즘은 N개의 프로세스 중에서 k 개의 프로세스가 동시에 임계구역에 진입할 수 있는 분산된 시스템이라고 할 수 있다. 분산된 시스템은 조정자가 일부 손실이 발생하여도 토큰 사이에 로그 메시지 기록을 통해 손실여부를 확인하여 새로운 토큰을 생성하여 프로세스를 진행한다. 자세한 사항은 다음 챕터에서 다루도록 한다.

3.4 토큰 손실 발생

토큰을 가진 프로세스가 충돌이 발생하여 토큰이 손실이 된 경우는 k 개의 토큰에서 한 개의 토큰이 손실되어 $k-1$개의 토큰이 프로세스를 순환할 때 로그를 통해 기록되었던 한 개의 토큰이 로그메시지를 안보내는 것을 통해서 토큰의 손실 발생을 알 수 있다.

토큰이 손실되었을 경우 $k-1$ 개의 토큰은 로그 메시지를 통해 서로의 위치를 공유하고 손실된 토큰의 위치를 파악해 그동안의 로그 기록을 통해 새로운 토큰이 생성될 위치를 선정하여 생성한다.

Ⅳ. 성능 비교

지금까지 제시한 k 토큰 링 알고리즘과 기존방식인 토큰 링 알고리즘과의 성능을 비교한다. 우선 토큰 링 알고리즘에서 N 개의 프로세스를 한번 순환할 때 총 N 번의 프로세스를 지나야 한다. 그러나, k 토큰 링 알고리즘은 $\frac{N}{k}$번만 순환하기 때문에 프로세스가 토큰에 접근하는 속도가 빠르다고 할 수 있다.

두 번째로는 토큰이 임계구역에 들어갈 때와 나올 때의 메시지 수에 대한 비교를 해보면 기존방식과 k-토큰 링 알고리즘 둘 다 토큰을 얻기 위해 메시지를 보내는 시점이 토큰이 무한으로 순환하다가 토큰을 사용하려는 메시지를 보내는 순간이 있을 수 있기 때문에 두 알고리즘 전부 1 개부터 무한대 까지라고 할 수 있다.

임계구역에 들어가고자 하는 프로세스가 토큰을 요청하는 메시지 수의 범위는 기존 토큰 링 알고리즘은 N 개의 프로세스가 있을 때 방금 지나친 토큰을 다시 받기위해서는 $N-1$번의 메시지를 전달해 토큰을 획득할 수 있다. 반면에, k-토큰 링 알고리즘은 토큰을 방금 지나쳐도 뒤에 있는 토큰의 간격만큼만 메시지를 전송하면 토큰을 획득할 수 있다. 그러나 토큰을 가진 k 개의 노드사이에서 임계영역에 들어갈 노드를 결정하기 위하여 분산 상호배제 알고리즘을 사용할 경우 $2(k-1)$ 개의 메시지가 추가로 필요하다.

	Token Ring Algorithm	k-Token Ring Algorithm
Token Count	1 개	K 개
Message per Entry / exit	$1, \cdots, \infty$	$1, \cdots, \infty$
Delay before entry	$1, \cdots, N-1$	$1, \cdots, \frac{N}{K}-1 + 2(k-1)$

표 1. 토큰 링 알고리즘과 k-토큰 링 알고리즘의 비교

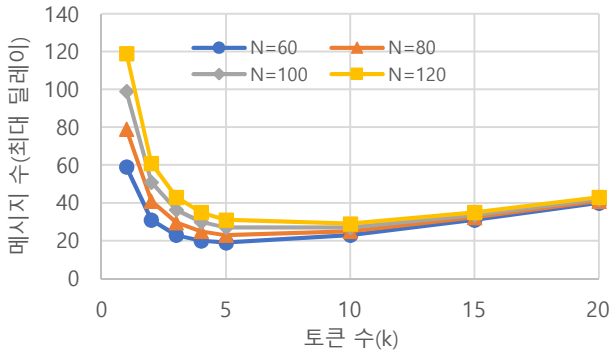

그림 1. 토큰 수에 따른 메시지 개수

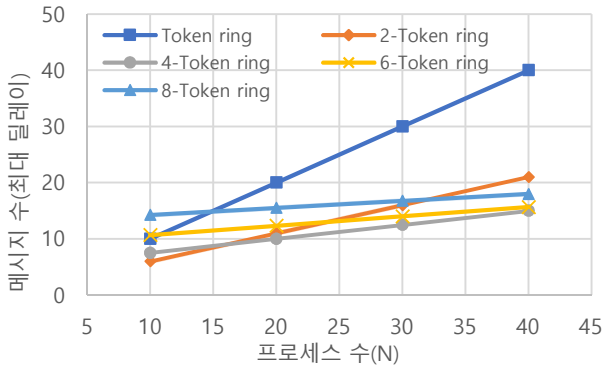

그림 2. 프로세스 수에 따른 메시지 개수

Ⅴ. 결론

본 논문에서는 분산시스템에서 기존의 토큰 링 알고리즘 방식을 개선한 k-토큰 링 알고리즘 방식을 제안하였다. 1 개의 토큰이 순환할 때보다 k개의 토큰이 순환하면서 임계구역에 들어가기 위해 요청하는 메시지의 수가 적어진다는 것을 확인하였다.

ACKNOWLEDGMENT

이 논문은 2018 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업과(NRF-2018R1D1A1B07040573) 2020 년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (P0008703, 2020 년 산업전문인력 역량강화사업).

참 고 문 헌

[1] RICART, Glenn; AGRAWALA, Ashok K. An optimal algorithm for mutual exclusion in computer networks. *Communications of the ACM*, 1981, 24.1: 9-17.

[2] SUZUKI, Ichiro; KASAMI, Tadao. A distributed mutual exclusion algorithm. *ACM Transactions on Computer Systems (TOCS)*, 1985, 3.4: 344-349.

[3] SINGHAL, Mukesh. A heuristically-aided algorithm for mutual exclusion in distributed systems. *IEEE transaction on computers*, 1989, 38.5: 651-662

[4] BULGANNAWAR, Shailaja; VAIDYA, Nitin H. A distributed k-mutual exclusion algorithm. In: *Proceedings of 15th International Conference on Distributed Computing Systems*. IEEE, 1995. p. 153-160.