'92 韓國自動制御學術會議論文集 1992. 10. 19~21

토큰 버스 네트워크의 동적 타이머 제어 방식 및 성능 해석에 관한 연구

정 범 진 ⁰, 권 욱 현

서울대학교 제어계측 공학과

Dynamic Timer-Controlled Algorithm and its Performance Analysis on the Token Bus Network

Beom Jin Chung, Wook Hyun Kwon

Dept. of Control and Instrumentation, Seoul National University

ABSTRACT

The IEEE 802.4 priority mechanism can be used to handle multiple data access classes of traffic. Several timers are used to realize the priority mechanism. The performance and stability of a token bus network depend on the assignment of such timers. In this paper, we present a dynamic timer assignment algorithm for the token passing bus network. The presented algorithm has simple structure for real-time applications and adaptively controls the set of initial timer values according to the offered traffic load. The assigment of the set of timers becomes easy due to the presented algorithm. Based on the iterative algorithm, some solutions such as mean waiting time are derived.

제 1 장. 서 론

토큰 버스 네트워크에서는 데이타를 전송할 수 있는 권리인 토큰이 어떤 시간내에는 반드시 주어진다는 결정론적인 특성을 갖고 있다. 이러한 결정론적인 특성으로 인하여 토큰 버스 네트워크는 작업 스케쥴이 정해진 생산 라인과 같은 실시간(Real-time)응용 환경에 주로 사용된다. 실시간 네트워크는 일반적으로 20 - 50 msec정도의 응답시간이 요구되는 것으로 알려지고 있다. 토큰 버스 네트워크에 관한 많은 해석이 이루어졌으며 대부분의 경우 네트워크의 성능에 영향을 줄 수 있는 관련 변수들이 주어졌을 때에의 성능을 분석하였다. 토큰 버스 네트워크을 해석함에 있어서 패킷 서비스 방식에 따라서 크게 고갈(Exhaustive)과 비고갈(Non-exhaustive Service)로 분류되며 이 중 고갈 서비스 방식은 한 스테이션에 토큰 도착후 큐(Queue)에 대기중인 모든 패킷을 서비스하여 토큰을 다른 스테이션으로 넘겨줄 경우 큐에 서비스할 패킷이 남아 있지 않는 방식이고, 비고갈 서비스 방식은 토큰 송신시에 큐에 서비스할 패킷이 잔류할 수도 서비스 방식이다. 비고갈 서비스 방식은 다시 게이트 서비스(Gated Service)와 제한 서비스(Limited Service) 방식으로 다시 세분된다[4,5,8]. 이중에서 제한 서비스 방식이 토큰 버스 프로토콜의 결정론적인 특성을 비교적 잘 나타낸 것으로 토큰 수신시 서비스할 수 있는 패킷의 최대 갯수가 제한이 된다. 그러나 제한 서비스 방식의 경우, 2개이상으로 제한하는 경우에 대해서는 아직까지 만족할 만한 연구 결과가 발표되고 있지 않고 있다.

토큰 버스 네트워크에서는 여러 종류의 타이머들이 정의되어 이와같은 결정론적인 특성을 나타낼 수가 있다. 따라서 이들 타이머 값을 어떻게 할당하는 가는 네트워크의 성능을 크게 좌우된다. 타이머 값들을 네트워크 트래픽 부하보다 작게 설정할 경우에는 한 스테이션에서의 데이타 송신 서비스율이 데이타 도착율보다 적게 되어 시스템이 불안정하게 될 수 있다. 반면 타이머 값을 너무 크게 할 경우에는 패킷 도착이 급증할 경우 해당 스테이션이 토큰을 장시간 소유하게 되어 네트워크의 응답성을 떨어지게 한다. 이는 토큰 소유 시간을 제한하여 여러 스테이션이 공평하게 네트워크망을 사용할 수 있도록 하자는 토큰 버스 프로토콜 자체의 취지를 벗어나는 동작이다. 그러므로 이 타이머를 네트워크 환경에 맞게 적절히 설정할 필요가 있는데 사용자가 이 값을 적절하게 설정하기 위해서는 전체 네트워크 트래픽과 타이머 설정 방식에 대한 지식이 요구되며 네트워크의 트래픽 분포가 변화될 시에는 이 값를 재설정해주는 작업이 필요하게 된다. Jayasumana의 연구에서는 이 타이머의 초기값들을 설정하는 방법을 제시하고 있는데 이 경우 시스템의 안정도와 네트워크의 이용도(Utilization)의 두 요인에 대해서 적절한 운전 영역을 식으로 유도하였다[9]. 그러나 이 논문에서 다루고 있는 시스템은 패킷 사이즈의 최대값이 제한된 경우이고, 최악의 경우를 가정하여 타이머 값을 설정하였기 때문에 네트워크 효율성이 저하된다는 점이다. 이외에도 타이머를 고려한 네트워크 성능을 다루는 여러 편의 논문이 발표되었는데 이 들 논문에서는 타이머의 설정 방법에 대한 방법론이 제시되어 있지는 않고 있다[6].

본 논문에서는 이들 타이머 값을 네트워크 트래픽 부하 정도에 따라서 동적으로 할당하는 알고리듬을 제시하고자 한다. 제시되는 알고리듬은 실시간 토큰 버스 네트워크를 고려

하여 간단한 구조 및 용이한 구현 방식을 갖도록 하였다. 본 논문에서는 평균 대기 시간을 성능 지표로 하였는데 이는 대기 시간이 실시간 네트워크에서 특히 중요한 성능 지표이기 때문이다. 제안된 알고리듬은 그 구현의 신뢰성을 향상시키고자 실제 제작된 토큰 버스 네트워크를 참조하였다. 본 알고리듬은 학습 알고리듬과 같이 최적해를 구하거나 수렴성의 개념은 갖고 있지 않다. 이러한 알고리듬은 구현이 용이하고 네트워크 환경 변화에 대해서 강인성을 갖고 있으며 학습 알고리듬과 같은 많은 데이타 저장 문제가 발생하지 않기 때문에 구현의 부담이 매우 적다. 본 논문의 순서는 다음과 같다. 2장에서는 알고리듬이 제안되고 3장에서는 제안된 알고리듬의 성능 해석을 하며 4장에서 결론을 내리고자 한다.

제 2 장 동적 타이머 제어 알고리듬

전술한 바와 같이 토큰 버스 네트워크는 실시간 네트워크용으로 사용되는 경우가 많으므로 타이머 설정 알고리듬 역시 실시간 네트워크 환경을 고려하여 제안되어야 한다. 알고리듬이 복잡하고 많은 정보를 필요할 경우에는 그 신뢰성은 향상될 수 있겠으나 수행 소요 시간이 증가하여 실시간 네트워크에 적용하기 어려게 된다. 따라서 간단한 형태의 알고리듬을 사용하여 수행 소요 시간을 최소로 하는 것이 바람직하다. 이러한 이유로 제안된 알고리듬은 자신의 스테이션이 갖고 있는 정보량 즉, 과거의 데이타 패킷(Packet) 대기 상태와 현재의 패킷 대기 상태량만을 사용하여 패킷 서비스율을 현재 네트워크 상태에 맞게끔 최우선 순위 토큰 유지 시간값을 조절한다. 다음은 본 논문에서 사용되는 기호들을 나타낸 것이다.

N : 토큰 버스 네트워크상에 연결된 스테이션 수
C : 데이타 분류 집합($C = \{ 0, 2, 4, 6 \}$)
$\lambda_{i,j}$: 종속 스테이션 i,j에 도착하는 패킷의 평균 도착율
$\mu_{i,j}$: 스테이션 i,j의 패킷 평균 서비스율
$Q_{i,j}^{(n)}$: 토큰 수신시 스테이션 i,j에 대기중인 패킷 수[1]
$K_{i,j}^{(n)}$: 토큰 소유시 스테이션 i,j에서 서비스되는 패킷의 수
$R_{i,j}^{(n)}$: 토큰 송신시 스테이션 i,j에 잔류중인 패킷의 수
$B_{i,j}^{(n)}$: 스테이션 i,J가 토큰 소유시 서비스될 수 있는 최대 예상 서비스 패킷의 수
$M_{i,j}$: Bi,j(n)의 최대값
$L_{i,j}$: Bi,j(0)의 초기값
$THT_{i,j}^{(n)}$: 스테이션 i,j가 토큰 수신시 대기중인 패킷들에 대한 최대 서비스 가능 시간값
$HPTHT_i^{(n)}$: 최우선 순위를 갖는 데이타에 대한 최대 서비스 가능 시간
$TTRT_{i,j}^{(n)}$: 스테이션 i,j에 대한 기준 토큰 순환 시간
$TRT_{i,j}^{(n)}$: 스테이션 i,j에 토큰이 도착시 이에 해당하는 토큰 순환 시간

제안되는 알고리듬은 패킷의 도착 분포와 서비스 분포 형태에 관계없이 동작하나 해석의 용이성을 위하여 패킷 도착간격은 지수 분포를 갖는 것으로 한다. 토큰 버스 네트워크를 해석하고자 할때에 토큰의 수신 시각에서 시스템을 관찰하는 방법이 대부분의 연구에서 사용되었으나 실제 토큰 버스 프로토콜에서는 토큰 도착시 패킷 전송을 하지 않으면서 토큰을 갖지 못하도록 되어 있다. 이러한 이유로 본 논문에서 제안되는 알고리듬의 시작시각은 토큰 수신 시각이 아닌 토큰 송신 시각으로 하였다.

알고리듬의 동작을 설명하기 위해서 임의의 i번째 스테이션에서 n번째 토큰의 송신 시점을 가정한다. 토큰 버스 프로토콜에서는 하나의 스테이션에서 여러 종류의 데이타를 상호 독립적으로 처리할 수 있도록 규정하고 있다. 이는 마치 스테이션 내부에 여러개의 종속 스테이션들이 있는 것과 같은데 설명의 편의를 위하여 여러 데이타 종류를 스테이션 i의 종속 스테이션 j들이 처리하는 것으로 설명하고자 한다. 스테이션 i,j에서 토큰 송신후 현재의 최대 예상 서비스 패킷의 수, $B_i^{(n)}$의 값을 비교하여 다음번 즉, (n+1)번째 토큰 수신시에 전송될 수 있는 $B_i^{(n+1)}$를 다음과 같이 설정한다.

경우 1) $L_{i,j} < B_{i,j}^{(n)} < M_{i,j}$:

$$B_{i,j}^{(n+1)} = B_{i,j}^{(n)} + I_i^{(n)} \qquad (1)$$

여기서

$$I_i^{(n)} = \begin{cases} 1 & , R_{i,j}^{(n)} > R_{i,j}^{(n-1)} \\ -1 & , R_{i,j}^{(n)} = R_{i,j}^{(n-1)} = 0 \\ 0 & , \text{Otherwise} \end{cases} \qquad (2)$$

경우 2) $B_{i,j}^{(n)} = L_{i,j}$:

$$B_{i,j}^{(n+1)} = \begin{cases} B_{i,j}^{(n)} + 1 & , R_{i,j}^{(n)} > R_{i,j}^{(n-1)} \\ B_{i,j}^{(n)} & , \text{Otherwise} \end{cases} \qquad (3)$$

경우 3) $B_{i,j}^{(n)} = M_{i,j}$:

$$B_{i,j}^{(n+1)} = \min \{ M_{i,j} , R_{i,j}^{(n)} \} \qquad (4)$$

알고리듬에 의하면 대기중인 팻킷 수가 증가시에는 이를 감소시키기 위해서 최대 예상 서비스 패킷 수 또한 증가하게 되여 스테이션 i가 매우 오랜 시간동안 토큰을 갖는 경우가 발생할 수 있다. 실시간 네트워크에서 기본적으로 최대 토큰 순환 시간이 제한되어야 하므로 이러한 상황은 사전에 방지되어야 한다. 경우 3의 식은 이러한 상황을 방지하기 위하여 최대 예상 서비스 패킷 수의 그 한계치를 규정하기 위해 도입된 식이다.

1) 윗 첨자 (n)은 n 번째 토큰을 의미한다.

상기 식들은 최대 예상 서비스 패킷 수를 나타내는 식이므로 이를 다시 타이머 시간 값으로 바꾸어야 한다. 패킷의 서비스 시간이 상수일 경우에는 타이머 값과 패킷 수간에 선형의 관계가 성립되므로 타이머 값으로 변환하는 것이 문제가 없겠으나, 일반적인 서비스 분포를 갖는 경우에는 설정된 타이머 값이 최대 전송 패킷 수만큼 정확히 변환될 수가 없다. 따라서 평균의 개념에서 다음의 식(5)로서 타이머 값 설정을 정의하였다.

$$HPTHT_i^{(n)} = B_{i,j}^{(n)} \mu_{i,j}^{-1} \quad , j = 6 \qquad (5.1)$$

$$TTRT_{i,j}^{(n)} = B_{i,j}^{(n)} \mu_{i,j}^{-1} \quad , j = 0, 2, 4 \qquad (5.2)$$

토큰 송신시각에서 위 식으로 각 데이타 분류에 대한 타이머 값들이 정해질 경우, n번째 토큰의 수신시에 토큰 유지 시간, $THT_{i,j}^{(n)}$은 다음과 같은 값으로 할당된다.

$$THT_{i,j}^{(n)} = \begin{cases} HPTHT_i^{(n)} & , j = 6 \\ TRT_{i,j}^{(n)} - TTRT_{i,j}^{(n)}, & j = 0, 2, 4 \end{cases} \qquad (6)$$

이상의 동적 타이머 제어 알고리듬을 통하여 원하는 평균 대기 시간이내로 네트워크의 성능을 유지하기 위하여 Little's Law를 사용한다. 즉, $B_i^{(n)}$의 최대값 M_i를 평균 대기 시간과 패킷 도착율의 곱으로 정의하면, 네트워크가 안정할 경우 알고리듬에 의하여 M_i이하로 대기중인 패킷수를 유지하려 하므로 결국 사용자가 원하는 평균 대기 시간이내 값이 유지되게 된다.

제 3 장 동적 타이머 제어 알고리듬에 대한 성능 해석

토큰 버스 네트워크을 해석하기 위하여 임의의 한 스테이션을 선정하여 이 스테이션에 토큰이 도착하는 순간에서부터 서비스를 완료하고 토큰을 다음 스테이션으로 넘겨주기까지의 큐와 알고리듬에 의한 네트워크 변수의 변화를 분석한다. 임의의 종속 스테이션 i, j에서 n번째 토큰이 도착할 때를 가정하면 서비스되는 패킷수 $K_j^{(n)}$에 대한 확률은 다음 식(7)과 같이 나타낼 수 있다.[2]

$$\Pr\{K_j^{(n)} = k\} = k_j^{(n)}(k) = \begin{cases} \Pr\{Q_j^{(n)} = k\} & , k = 0 , j = 6 \\ \Pr\{Q_j^{(n)}=k\} + \Pr\{TRT_j^{(n)} > TTRT_j^{(n)}\} & , k = 0 , j = 0,2,4 \\ \sum_{m=Lj}^{Mj} \Pr\{THT_j^{(n)}=m\mu_j^{-1}\} k_{j,m}^{(n)}(k) & , k > 0 \end{cases} \qquad (7)$$

위 식에서 $k_{j,m}^{(n)}(k)$는 $\Pr\{K_j^{(n)} = k \mid THT_j^{(n)} = m\mu_j^{-1}\}$로 정의된다. 위 식(7)에서 조건부 확률식은 다음과 같이 구해진다.

$$k_{j,m}^{(n)}(k) = \sum_{l=k}^{\infty} \Pr\{Q_j^{(n)}=l\} \Pr\{Y_{j,m}^{(n)}=k\} + \sum_{l=k}^{\infty} \Pr\{Y_{j,m}^{(n)}=l\} \Pr\{Q_j^{(n)}=k\} - \Pr\{Q_j^{(n)}=k\} \Pr\{Y_{j,m}^{(n)}=k\} \qquad (8)$$

여기서 $\Pr\{Y_{j,m}^{(n)} = k\}$는 $THT_j^{(n)}$이 $m\mu_j^{-1}$인 조건하에서 서비스될 수 있는 패킷의 최대수가 k일 확률을 나타낸다. 따라서 패킷 서비스 시간을 m번 합한 값으로 $\tau_{j,m}^{(n)}$를 정의하면 $\Pr\{Y_{j,m}^{(n)} = k\}$는 $\Pr\{\tau_{j,k-1}^{(n)} < m\mu_j^{-1} < \tau_k^{(n)}\}$으로 표현될 수 있다. 여기서 패킷 서비스 시간은 상호 독립 동일 분포(Idenitically Independent Distribution)를 갖으므로 $\tau_{j,m}^{(n)}$는 Erlang분포를 갖게 된다.

서비스가 완료되고 토큰이 다음 스테이션으로 넘겨진 후에 큐에 잔류하고 있는 패킷 수에 대한 확률 변수는 다음 식(9)과 같다.

$$R_j^{(n)} = Q_j^{(n)} - K_j^{(n)} \qquad (9)$$

위 식에서 알 수 있듯이 서비스동안 새로 도착하는 패킷들은 큐에 저장되지 않는다. 큐에 잔류하고 있는 패킷 수, $R^{(n)}$에 대한 확률 분포는 식 (10)과 같다.

$$r_j^{(n)}(k) = \Pr\{R_j^{(n)} = k\} = \begin{cases} \sum_{m=Lj}^{Mj} \Pr\{HPTHT^{(n)}=m\mu_j^{-1}\} r_{m,j}^{(n)}(k) & , j = 6 \\ \sum_{m=Lj}^{Mj} \Pr\{TTRT_j^{(n)}=m\mu_j^{-1}\} r_{m,j}^{(n)}(k) & , j = 0,2,4 \end{cases} \qquad (10)$$

여기서

$$r_{m,j}^{(n)}(k) = \begin{cases} q_j^{(n)}(0) + \sum_{i=1}^{\infty} q_j^{(n)}(i) \sum_{j=i}^{\infty} y_{j,m}^{(n)}(j) & , k = 0 \\ \sum_{m=Lj}^{Mj} \Pr\{THT_j^{(n)}=m\mu_j^{-1}\} \sum_{i=k}^{\infty} q_j^{(n)}(i)\, y_{j,m}^{(n)}(i-k) & , k > 0 \end{cases} \qquad (11)$$

위식에서 $q_j^{(n)}(i)$와 $y_{j,m}^{(n)}(i)$는 각각 다음과 같이 정의된다.

$$q_j^{(n)}(i) = \Pr\{Q_j^{(n)} = i\}, \quad y_{j,m}^{(n)}(i) = \Pr\{Y_{j,m}^{(n)} = i\} \qquad (12)$$

n번째 토큰의 서비스 완료후 다음번 (n+1)번째 토큰의 수신시 최대 서비스 가능 시간값, $THT^{(n+1)}$이 알고리듬에 의하여 새

2) 스테이션 i는 임의로 선정된 것이므로 이후 기호 사용상의 편의를 위하여 스테이션 i를 나타내는 아래 첨자는 생략하고 데이타 분류 첨자 j만을 사용한다.

로 할당되는데 이의 확률을 구하기 위해서는 먼저 최대 예상 서비스 패킷수, $B_j^{(n)}$에 대한 확률 분포를 구해야 한다. 따라서 상태 변수를 $B_j^{(n)}$로 두고서 동적 시간 제어 알고리듬에 의한 상태 천이 관계를 나타내면 다음과 같다.

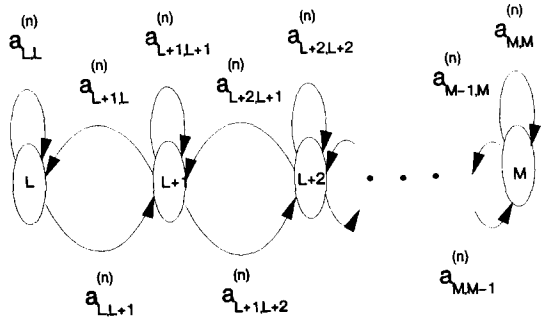

그림 1. 동적 타이머 제어 알고리듬에 의한 상태 천이도

$a_{i,j}^{(n)}$은 n번째 도착시 상태 i에서 j로의 상태 천이 확률을 의미한다. 위의 상태 천이도로부터 (n+1)번째의 상태 확률 $b^{(n+1)}(i)$는 다음과 같이 나타내어진다. 상태 천이 확률 및 상태 확률은 데이타 분류에 관계 없이 동일하게 표현되므로 기호 사용의 편의를 위하여 스테이션 j를 나타내는 아래 첨자는 표시하지 않도록 한다.

$$b^{(n+1)}(j) = \begin{cases} a_{L,L}^{(n)}\, b^{(n)}(L) + a_{L+1,L}^{(n)} b^{(n)}(L+1) & , i = L \\ a_{i-1,i}^{(n)}\, b^{(n)}(i-1) + a_{i,i}^{(n)} b^{(n)}(i) + a_{i+1,i}^{(n)} b^{(n)}(i+1) & , L < i < M \\ a_{i-1,i}^{(n)}\, b^{(n)}(i-1) + a_{i,i}^{(n)} b^{(n)}(i) & , i = M \end{cases} \quad (13)$$

여기서 상태 천이 확률 $a_{i,j}^{(n)}$은 다음 식(14~16)과 같다.

$$a_{k,k-1}^{(n)} = \Pr\{R_k^{(n)}=0 \mid R^{(n-1)}=0\}\ \Pr\{R^{(n-1)}=0\} \quad , k = L+1, 2, \ldots, M \quad (14)$$

$$a_{k,k}^{(n)} = \begin{cases} \sum_{r=0}^{\infty} \Pr\{R_L^{(n)} \le r \mid R^{(n-1)}=r\}\ \Pr\{R^{(n-1)}=r\} & , k = L \\ 1 - a_{k,k-1}^{(n)} - a_{k,k+1}^{(n)} & , k = L+1, 2, \ldots, M-1 \\ 1 - a_{k,k-1} & , k = M \end{cases} \quad (15)$$

$$a_{k,k+1}^{(n)} = \sum_{r=0}^{\infty} \Pr\{R_k^{(n)} > r \mid R^{(n-1)}=r\}\ \Pr\{R^{(n-1)}=r\} \quad , k = 1, 2, \ldots, M-1 \quad (16)$$

(n-1)과 n번째 토큰 송신시 큐에 잔류하는 패킷 분포는 스테이션 독립 가정으로부터 상호 독립이므로 위 식의 조건부 확률식들은 실질적으로는 두 확률의 곱으로서 나타낼 수 있다. 이는 네트워크에 연결된 스테이션 수가 많을수록 현실성이 있다. n번째 토큰 수신시 종속 스테이션 i,j에서 패킷들의 서비스 소요 시간, $E_j^{(n)}$에 대한 확률 분포는 서비스되는 패킷 수에 대한 확률 분포를 사용하여 식(17)과 같이 구할 수 있다.

$$\Pr\{E_j^{(n)} \le t\} = \sum_{k=0}^{\infty} \Pr\{\tau_{j,m}^{(n)} \le t \mid K_j^{(n)}=k\} \times \Pr\{K_j^{(n)}=k\} \quad (17)$$

스테이션 i에서 소요되는 총 시간, $S^{(n)}$는 패킷 서비스 소용 시간, $E_j^{(n)}$와 스테이션 i에서 스테이션 (i+1)로의 토큰 패싱 시간, $P^{(n)}$의 합으로 나타낼 수 있다. 토큰 패싱 시간을 평균이 $1/\rho_i$인 지수 분포로 가정하면 $S^{(n)}$의 확률 분포, $F_S^{(n)}(t)$는 토큰의 패싱 시간, $F_P^{(n)}(t)$과 패킷들의 서비스 소요 시간, $F_E^{(n)}(t)$에 대한 확률 분포의 Convolution으로 다음의 식과 같이 얻을 수 있다.

$$F_S^{(n)}(t) = F_P^{(n)}(t) * \prod_{j \in C} F_{Ej}^{(n)}(t) \quad (18)$$

식 (18)로부터 토큰 순환 시간 $C^{(n)}$에 대한 식을 표현하면 다음과 같다.

$$C^{(n)} = \sum_{j=i}^{N} S_j^{(n-1)} + \sum_{j=1}^{i-1} S_j^{(n)} \quad (19)$$

토큰 순환 시간은 위의 식에서 보듯이 다른 스테이션 시간과의 합으로 표시되는데, 이 들간은 상호 독립이 아니므로 위의 식으로부터 확률 분포를 계산하기는 매우 어려운 일이 된다. 따라서 대개의 관련 연구에서는 이들간의 상호 독립을 가정하고 식을 전개한다. 이러한 상호 독립 가정을 바탕으로하여 토큰 순환 시간에 대한 Laplace 변환을 구하면 다음과 같다.

$$\Phi_C^{(n)} = \prod_{j=i}^{N} \Phi_{cj}^{(n-1)}(s) \prod_{j=1}^{i-1} \Phi_{cj}^{(n)}(s) \quad (20)$$

토큰 순환 시간에 대한 확률 분포를 구하기 위해서는 식(20)에 대한 역 Laplace 변환을 구하여야 하는데 이를 위해서는 매우 많은 양의 계산이 필요하므로 평균과 분산의 1,2차 모멘트(Moment)를 사용한 근사식을 이용하여 각 스테이션의 순환 시간에 대한 근사 분포식을 구한다[9].

식 (21)은 (n+1)번째 토큰이 스테이션 i에 도착시에 큐에 j개의 패킷이 존재할 확률, $q^{(n+1)}(k)$를 나타낸다.

$$q_j^{(n+1)}(k) = \sum_{l=k}^{\infty} r_j^{(n)}(l)\ g_j^{(n+1)}(l-k) \quad (21)$$

여기서

$$g_j^{(n)}(k) = \int_0^{\infty} \frac{(\lambda_j x)^k}{k!} \exp(-\lambda_j x)\, dF_c^{(n+1)}(x). \quad (22)$$

위 식에서 $F_c^{(n)}()$는 n번째 토큰 순환 시간의 확률 분포이다. 이상의 식에서 볼 수 있듯이 이 식들로부터 정형화된 성능 지표식을 구한다는 것은 매우 어려운 일이 되므로 Tran-Gia의 경우에서와 같은 반복 알고리듬(iterative algorithm)를 사용하여 토큰 도착시의 패킷들의 대기 분포 및 토큰 순환 시간의 정상 상태(steady state) 분포를 근사적으로 구하게 된다.

평균 패킷 대기 시간

임의의 시각에 큐를 보았을 경우, 종속 스테이션 i, j에 k개의 패킷이 대기할 확률을 $P_{j,k}^*$라 하면 $P_{j,k}^*$는 다음과 같이 나타낼 수 있다.

$$P_{j,k}^* = \frac{E[C_{j,0}]}{E[C]} Q_j(0)\, b_{j,k;0}^* + \frac{1}{E[C]} \sum_{l=1}^{M_j} E[C_l]\, Q(l) \sum_{i=0}^{k} r_{j,i;l}\, b_{j,k-i;l}^* \quad (23)$$

여기서

$E[C]$: 평균 토큰 순환 시간

$E[C_l]$: l개의 패킷이 서비스되는 조건하에서의 평균 토큰 순환 시간

$r_{j,k;l}$: l개의 패킷이 서비스되는 조건하에서의 서비스 완료후 큐에 k개의 패킷이 잔류할 확률

$b_{j,k;l}^*$: j개의 패킷이 서비스되는 조건하에서의 역행 재현(Backward recurrence) 토큰 순환 시간동안 k개의 패킷이 도착할 확률

위 식에서 $r_{k;j}$는 토큰 도착시에 큐에 대기중인 패킷수로부터 구할 수 있으므로 $r_{j,k;j}$는 다음과 같다.

$$r_{j,k;j} = q_j(k+j) \quad (24)$$

토큰 순환 시간은 엄밀한 의미에서는 Renewal 프로세서로 볼 수 없으나 스테이션의 갯수가 많은 경우에는 비교적 스테이션간의 종속성이 적어지므로 Renewal 프로세스로 근사화 시킬 수가 있다. 이러한 가정을 바탕으로 j개의 패킷이 서비스되는 조건하에서의 역행 재현 토큰 순환 시간, $C_{j;l}^*$을 구하기 위해서 먼저 조건부 토큰 순환 시간, $C_{j;l}$을 구하면 식(25)과 같다.

$$C_{j;l} = \sum_{\substack{g=1 \\ g \neq j}}^{N} S_{g;l} + \rho_j^{-1} + \sum_{m=1}^{l} T_{j,m} \quad (25)$$

스테이션 상호 독립 가정과 식 (25)로부터 l개의 패킷이 서비스되는 조건하에서의 평균 토큰 순환 시간, $E[C_{k;l}]$과 분산값은 구하면 식 (26)과 식 (27)로 나타낼 수 있다.

$$E[C_{j;l}] = \sum_{\substack{g=1 \\ g \neq j}}^{N} E[S_{g;l}] + \rho_j^{-1} + 1/\mu_j \quad (26)$$

$$Var[C_{j;l}] = \sum_{\substack{g=1 \\ g \neq j}}^{N} Var[S_{g;l}] + \rho_j^{-1} + 1/\mu_j^2 \quad (27)$$

1,2차 모멘트(Moment)를 사용한 근사식을 이용하면 역행 재현 토큰 순환 시간에 대한 확률 분포를 얻을 수 있다.

$$F_{C_{j;l}}^*(t) = (1 - F_{C_{j;l}}(t))/E[C_{j;l}] \quad (28)$$

식 (28)로부터 역행 재현 토큰 순환 시간동안의 패킷 도착 분포를 구하면 다음과 같다.

$$b_{j,k;l}^* = \int_0^{\infty} \frac{(\lambda_j t)^k}{k!} \exp(-\lambda_j t)\, dF_{C_{j;l}}^*(t) \quad (29)$$

식 (24 ~ 29)로부터 임의의 시각에 큐를 보았을 경우의 패킷 대기 확률, P_k^*를 구할 수가 있다. 따라서 평균 대기 패킷수, $E[K_j]$는

$$E[K_j] = \sum_{k=0}^{\infty} k P_{j,k}^* \quad (30)$$

과 같이 나타낼 수 있으며 Little's Law를 사용하면 평균 대기 시간, $E[W_j] = E[k_j]/\lambda_j$ 를 얻을 수 있다.

제 4 장 결 론

본 논문에서는 토큰 버스 네트워크에서 데이타 트래픽 제어에 사용되는 여러 타이머 값들을 네트워크 트래픽 부하 정도에 따라서 적절하게 할당하여 주는 타이머 제어용 실시간 알고리듬을 제시하였다. 제안된 알고리듬은 토큰 버스 네트워크 상의 여러 종류의 데이타 전송을 지원할 수 있도록 설계되었으며 특히 실시간 환경에 적용하기 위하여 복잡하고 많은 계산이나 정보가 필요한 알고리듬 형태를 지양하고 간단하고 안정된 동작을 할 수 있도록 하였다. 따라서 이를 실현시의 부담을 최소화할 수 있도록 하였다.

제안된 알고리듬을 매우 엄격한 실시간 전송이 이루어져야 하는 환경에 사용하고자 할 경우, 실시간 데이타의 최대 부하를 알면 실시간 데이타 전송을 보장할 수가 있다 제안되는 알고리듬은 실질적으로 어떠한 형태의 토큰 버스 네트워크 구성에도 적용할 수 있도록 하였다. 제안된 알고리듬의 평균 대기 시간 식을 반복 알고리듬에 의거하여 제시하였다.

참고 문헌

[1] A.G. Konheim and B. Meister, "Waiting lines and times in a system with polling," J.ACM, vol. 21, no. 3, 1979.

[2] R.B. Cooper and G. Murray, "Queues served in cyclic order," Bell Syst. Tech.J.,vol. 48, no. 3, 1969.

[3] J. M. Tien et al., "Performance analysis of a single-token ring communication network," in Proc. IEEE Conf. Cybern., Soc., 1982.

[4] H.Takagi, Analysis of Polling Systems. Cambridge, MA: MIT Press, 1986

[5] E.G. Coffman, Jr. and I. Mitrani,"A characterization of waiting time performance realizable by single server queues," Oper. Res., vol 28, pp. 810-821, 1980.

[6] R.H. Myers and W.H. Carter, Jr., "Response surface techniques for dual response systems," Technometrics, vol. 15, no. 2, May 1973.

[7] Myungwhan Choi and C. M. Krishma, "An adaptive Algorithm to Ensure Differential Service in a Token-Ring Network," IEEE Trans. pn Computers, vol. 9, no. 1, pp. 19-33, Jan. 1990.

[8] Oliver C. Ibe and Xian Cheng, "Stability Conditions for Multiqueues Systems with Cyclic Service," IEEE Trans. on AC, vol. 33, no. 1, pp. 101-104, Jan. 1988.

[9] P.Tran-Gia and T. Raith," Multiqueue systems with finite capacity and nonexhaustive cyclic service," in Proc. Int.Sem. Comput. Network. Perform. Eval., Tokyo, Japan, 1985.

[10] A.P. Jayasumana, C.G. Jayasumana, "On the Use of the IEEE 802.4 Token Bus in Distributed Real-Time Control Systems," IEEE Trans. on Industrial Electronics, vol.36, No. 3, pp. 391-397, Aug. 1989