4.13 Multi-Dimensional Erlang-B Class.

복습: 기본적인 Erlang-B 모델은 단일 자원으로, 서비스요청(ex) 전화통화시도)이 도착하고 시스템에 충분한 리소스가 없어서 서비스는 받지 못하고 차단되는 상황 모델링

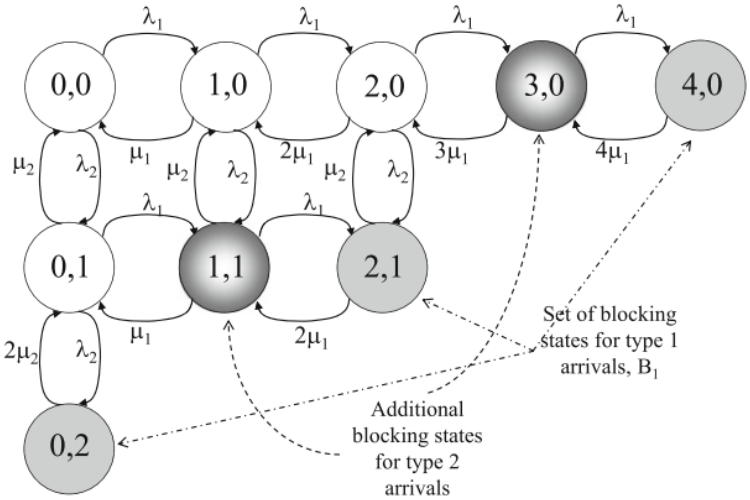


Fig. 4.21: Example of two-dimensional Markov Chain with limit capacity $C = 4$, $C_1 = 1$, and $C_2 = 2$ units of capacity

① We consider 2 independent Poisson arrival processes with mean rates λ_1 and λ_2. ② The service times of these two arrivals are exponentially distributed with mean rates μ_1 and μ_2, respectively, with $\mu_1 \neq \mu_2$. ③ There are no waiting places. These arrivals of requests share the resources from a common pool of capacity C. Each type 1 arrival needs a capacity C_1; each type 2 arrival needs a capacity C_2. A new arrival is blocked and lost if it finds the system when not enough capacity is available, that is the residual capacity is lower than C_1 for type 1 arrivals and the residual capacity is lower than C_2 for type 2. ④

⑤ To study this system, we have to use a two-dimensional (2D) Markov chain where the generic state (i, j) represents how many requests of type 1 and how many requests of type 2 are simultaneously present in the system.

Because of the capacity limit C, not all the states (i, j) are possible, but only those of the set Ω defined below:

$$\Omega = \{(i, j) \mid iC_1 + jC_2 \leq C\} \tag{4.73}$$

① 두 가지 독립적인 포아송 프로세스를 가정. 각각 평균비율 λ_1, λ_2 라고 가정

② 두 프로세스의 서비스 시간은 평균 비율 μ_1, μ_2 로 분포되어 있으며 $\mu_1 \neq \mu_2$이다

③ There are no waiting places = 대기 장소가 없다. 즉, 시스템은 도착하는 요청을 대기열에 넣어 처리할 수 있는 여유가 있을 때까지 기다리게 하지 않는다. 요청이 들어오면 즉시 처리를 시작하거나, 필요한 리소스가 없다면 거절된다.

④ 요청으로 인한 도착들이 모두 C 라는 고정된 총 용량을 갖는 공통 리소스 풀에서 리소스를 공유한다. 이는 각각의 요청이 독립적으로 자신만의 리소스를 가지고 있지 않고, 모든 요청이 같은 리소스를 경쟁적으로 사용한다는 것을 의미. 예를들어, 여러 사용자가 동시에 데이터 센터의 메모리 같은 컴퓨터 자원에 접근할 때, 이런 자원들은 모든 사용자에게 공통된 용량으로 제공된다. 따라소 이러한 시스템에서는 리소스 요청이 동시에 발생할 때, 총 용량 C를 넘지 않는 범위 내에서만 요청을 수용할 수 있으며, 그 범위를 초과하는 요청은 서비스를 받지 못하고 차단된다.

Type1 arrivals needs a capacity C1; 타입 1이 용량으로 C1을 필요로 한다는 것은, 타입 1 요청이 서비스를 받기 위해 고정된 양의 리소스를 점유해야 한다는 의미이다. 이 경우 각 타입 1의 요청은 리소스 풀에서 C1단위의 리소스를 차지한다.

Type2 arrival needs a capacity C2; 위 설명과 동일

2차원 마르코프 체인을 사용하여 (i,j)가 현재 시스템에 존재하는 타입1과 타입2의 요청의 수를 나타내는 generic상태를 표현한다. 용량제한 C 때문에 모든 상태 (i,j)가 가능한 것은 아니며, 오직 집합Ω에 정의된 상태들만 가능하다.이 집합은 식 4.73에 의해 정의된다.

Let us denote the traffic intensity due to type 1 arrivals as $\rho_1 = {}^{\lambda_1}/_{\mu_1}$. Moreover, the traffic intensity due to type 2 arrivals is $\rho_2 = {}^{\lambda_2}/_{\mu_2}$.
Figure 4.21 shows an example of the 2D Markov chain modeling the system for $C_1 = 1$ and $C_2 = 2$ units of capacity with $C = 4$ units of capacity. This chain has 9 states. For numerical results we will also use $\rho_1 = 2$ Erl and $\rho_2 = 1$ Erl. Note that the system would become mono-dimensional if $\mu_1 = \mu_2$.
We can solve the state probabilities by writing equilibrium conditions to balance the flows around each state. We obtain a linear system with 9 equations, where one of them is redundant so that we need to add the normalization condition to express $P(0, 0)$. Each of the following equations represents an equilibrium condition related to the state for which the probability is on the left side.

1페이지의 첫번째 그림 4.21은 C1=1, C2=2 용량을 가진 시스템의 2D 마르코프 체인 모델링 예를 모여준다. 식 4.73에서 정의된 집합 오메가에 따라 총 9가지의 상태가 나올 수 있다.

mu1=mu2일 때의 수치결과를 위해 lo1=2, lo2=1을 사용한다. Mu1=mu2 라는것은 시스템이 단일 차원의 특성을 가지게 될 것이라는 것을 의미한다. 일반적으로 다차원 시스템은 여러 유형의 요청이 있을 때 각각 다른 서비스율을 가지고 있다. 이는 각 요청 유형이 시스템을 통과하는 속도가 다르다는 것을 의미하며, 이로 인해 시스템의 동작이 더 복잡해진다. 예를들어, 하나의 요청 유형이 다른 요청 유형보다 더 빠르게 처리될 수 있다. 그러나 만약 모든 유형의 서비스율이 같다면 시스템은 더 단순해진다. 모든 요청이 같은 속도로 서비스되기 때문에, 마치 하나의 요청 유형만 존재하는 것처럼 시스템을 분석할 수 있다.

그림에서 점선 화살표가 가리키는 additional blocking states for type 2는 타입 1의 요청이 차단될 추가적인 상태들을 나타낸다. 타입 2요청은 2단위의 용량을 필요로 하기 때무누에. 시스템이 3단위의 용량만 남았을 경우, 새로운 타입2 요청은 수용될 수 없다. 예를들어 상태 (1,1)은 1*1+2*1=3으로 타입 2의 요청을 하나 더 수용할 수 없다. type1의 경우의 화살표도 마찬가지로 해석하면 된다.

마르코프 체인을 사용하여 시스템의 각 상태에 대한 확률을 구하는 방법은 도착률과 서비스율이 균형을 이루는 균형조건을 사용하는 것이다. 우리는 9개의 방정식으로 이루어진 선형 시스템을 얻을 수 있고 이러한 방정식 중 하나가 다른 방정식들로부터 유도될 수 있어 필요하지 않다. 다시 말해서, 9개의 방정식 중 8개만이 독립적이고, 마지막 방정식은 다른 방정식들과 선형적으로 관련되어 있기 때문에 추가적인 정보를 제공하지 않는다. 그렇기 때문에 모든 상태 확률의 합이 1이 되어야 한다는 정규화조건을 추가해야한다.

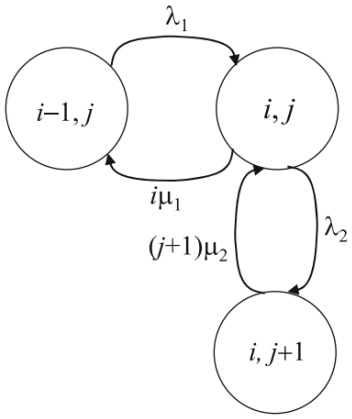


Fig. 4.22: Local equilibrium cases

local equilibrium

그림 4.22는 마르코프 체인에서 특정상태(i,j)로부터 다른 상태들로의 전이를 나타내며, 이를 통해 시스템에서 서비스를 받는 요청의 변화를 모델링하고 있다.

i,j는 타입 1의 요청이 i 개 이고 타입 2의 요청이 j 개 있는 상태를 나타낸다.

lamda1은 앞에서 언급했듯이 타입 1의 요청의 도착률로 새로운 타입1 요청이 시스템에 들어오는 비율을 의미한다. lamda1 밑의 화살표는 상태 (i-1,j)에서 (i,j)로의 전이를 나타낸다.

그다음 i*mu화살표를 보자. 서비스 완료율이 i*mu인 이유는 시스템이 동시에 여러 요청을 처리할 수 있고, 각각의 요청은 독립적으로 서비스를 완료한다는 가정 때문이다. 이 시스템에서 각 타입 1 요청이 별도의 서비스 채널을 통해 처리된다. 따라서 시스템에 i 개의 타입 1 요청이 있으면, 각 요청은 서비스 채널 하나를 점유하고 각각의 채널은 평균적으로 mu1의 비율로 요청을 서비스 완료한다.

$$
\begin{aligned}
&1)\ (\lambda_1+\lambda_2)\,P(0,0) = \mu_1 P(1,0) + \mu_2 P(0,1)\\
&2)\ (\lambda_1+\lambda_2+\mu_1)\,P(1,0) = 2\mu_1 P(2,0) + \mu_2 P(1,1) + \lambda_1 P(0,0)\\
&3)\ (\lambda_1+\lambda_2+\mu_2)\,P(0,1) = \mu_1 P(1,1) + 2\mu_2 P(0,2)\\
&4)\ (\lambda_2+\mu_1+\mu_2)\,P(1,1) = \lambda_1 P(0,1) + 2\mu_1 P(2,1) + \lambda_2 P(1,0)\\
&5)\ (\lambda_1+\lambda_2+2\mu_1)\,P(2,0) = 3\mu_1 P(3,0) + \mu_2 P(2,1) + \lambda_1 P(1,0)\\
&6)\ (2\mu_1+\mu_2)\,P(2,1) = \lambda_2 P(2,0) + \lambda_1 P(1,1)\\
&7)\ (\lambda_1+3\mu_1)\,P(3,0) = \lambda_1 P(2,0) + 4\mu_1 P(4,0)\\
&8)\ 2\mu_2 P(0,2) = \lambda_2 P(0,1)\\
&9)\ 4\mu_1 P(4,0) = \lambda_1 P(3,0)
\end{aligned}
\tag{4.74}
$$

위의 수식들은 지역 평형 상태를 나타내며, 각 상태로 들어오는 흐름과 나가는 흐름이 균형을 이뤄야 한다는 것을 의미한다.

1) 상태(0,0)에서 (1,0)과 (0,1)로 이동하는 전이율의 합이 상태(1,0)과 (0,1)에서 상태 (0,0)으로 돌아오는 전이율의 합과 같아야 함을 의미

2) Quiz 해석써보기

3) Quiz 해석써보기

1,2,3 했으면 4~9까지도 똑같으니까 pass.

$$\begin{aligned}\lambda_1 P(i-1,j) &= i\mu_1 P(i,j)\\ \lambda_2 P(i,j) &= (j+1)\,\mu_2 P(i,j+1)\end{aligned} \tag{4.75}$$

이 방정식은 인접한 상태들 간의 지역 평형을 설명한다.

상태(i-1, j)에서 상태(i , j)로 이동하는 흐름이 (i , j)에서 (i-1,j)로 이동하는 흐름과 균형을 이루어야 한다는 것을 의미

두번째 수식은 상태(i,j)에서 (i,j+1)의 흐름이 (i,j+1)에서 (i,j)의 흐름과 같아야 한다는 것을 의미

이 식을 정리하면 아래의 4.76, 4.77을 얻을 수 있음

$$P(i,j) = \frac{\lambda_1}{i\mu_1} P(i-1,j) = \ldots = \frac{\rho_1^i}{i!} P(0,j) \tag{4.76}$$

$$P(i,j) = \frac{\lambda_2}{j\mu_2} P(i,j-1) = \ldots = \frac{\rho_2^j}{j!} P(i,0) \tag{4.77}$$

위 두식을 통해 아래 4.78식 유도가능

$$P(i,j) = \frac{\rho_1^i \rho_2^j}{i!j!} P(0,0) \tag{4.78}$$

4.78은 상태 (i,j)에 대한 확률을 최초 상태 P(0,0)에 대한 확률로부터 계산하는 공식이다.

$$G(\Omega) = \sum_{(i,j)\in\Omega} \frac{\rho_1^i \rho_2^j}{i!j!} \tag{4.79}$$

According to Fig. 4.21, the set Ω of all the possible states is

$$\Omega = \left\{ \begin{matrix} (0,0)\ (1,0)\ (2,0)\ (3,0)\ (4,0) \\ (0,1)\ (1,1)\ (2,1) \\ (0,2) \end{matrix} \right\} \tag{4.80}$$

4.78+4.79.

$$P(0,0)\,G(\Omega) = P(0,0)\left[\frac{\rho_1^0 \rho_2^0}{0!\,0!} + \frac{\rho_1^1 \rho_2^0}{1!\,0!} + \cdots + \frac{\rho_1^0 \rho_2^2}{0!\,2!} \right]$$

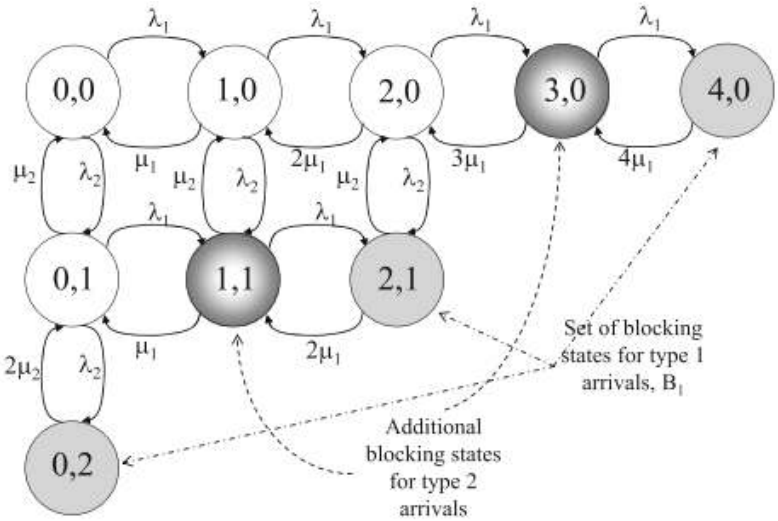
$$= P(0,0) + P(1,0) + \cdots + P(0,2) = 1$$

$$\therefore P(0,0) = \frac{1}{G(\Omega)}$$

(결론) : 2D 마르코프 체인을 위한 단일 해결 방법은 없음. 손실 대기 시스템과 같이 다양한 서비스 속도를 가진 시스템을 분석하기 위한 간단한 방법부터, 복잡한 시스템을 효율적으로 분석할 수 있는 알고리즘 방법까지 여러 방법이 존재함.

(Continue..) **Erlang-B 공식을 사용하는 방식으로 차단 확률 계산**

시스템의 용량 한계가 있고, 두 유형의 도착이 동일한 차단 조건을 경험하는 특수한 경우를 다루자.



1) 시스템 용량(C) : 전체 시스템의 용량은 C로 주어지며, 이는 두 서비스 클래스 C1과 C2가 공유한다.

2) 상태 공간(Ω) : 상태 공간은 가능한 모든 상태 (i,j)의 집합으로, 여기서 i와 j의 합이 시스템의 최대 수용량 l($l = \frac{C}{C_1} = \frac{C}{C_2}$)을 초과하지 않는다. (i는 유형 1 요청의 수, j는 유형 2 요청의 수)

3) 차단 상태(B) : 차단 상태는 i와 j의 합이 정확히 l과 같을 때로 정의된다. (이는 시스템이 가득 차서 더 이상 요청을 받아들일 수 없는 상태를 의미한다.)

4) 차단 확률(P_B) 계산 : 차단 확률은 도착하는 요청이 서비스를 받지 못하고 차단되는 비율을 나타낸다.

$$P_B = \sum_{B} P(i,j) = \sum_{i=0}^{\ell} P(i,\ell-i) = \frac{1}{G(\Omega)} \sum_{i=0}^{\ell} \frac{\rho_1^i}{i!} \frac{\rho_2^{\ell-i}}{(\ell-i)!}$$

$$= \frac{1}{G(\Omega)} \sum_{i=0}^{\ell} \binom{\ell}{i} \frac{\rho_1^i \rho_2^{\ell-i}}{\ell!} = \frac{1}{G(\Omega)} \frac{(\rho_1+\rho_2)^{\ell}}{\ell!}$$

→ 뉴턴의 이항 정리 : $(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$

(P(i,j) : 시스템이 상태 (i,j)에 있을 확률 / l : 시스템이 수용할 수 있는 최대 요청 수)

정규화 상수 G(Ω) : 상태 공간 Ω 내의 모든 상태 확률의 합

$$G(\Omega) = \sum_{i=0}^{\ell} \sum_{j=0}^{\ell-i} \frac{\rho_1^i}{i!} \frac{\rho_2^j}{j!} = \underset{k=i+j}{\text{using}} = \sum_{i=0}^{\ell} \sum_{k=i}^{\ell} \frac{\rho_1^i}{i!} \frac{\rho_2^{k-i}}{(k-i)!} = \begin{matrix}\text{reorganizing the} \\ \text{terms of the double} \\ \text{sum in a different way}\end{matrix}$$
$$= \sum_{k=0}^{\ell} \sum_{i=0}^{k} \frac{\rho_1^i}{i!} \frac{\rho_2^{k-i}}{(k-i)!} = \sum_{k=0}^{\ell} \frac{\rho_2^k}{k!} \sum_{i=0}^{k} \binom{k}{i} \left(\frac{\rho_1}{\rho_2}\right)^i = \begin{matrix}\text{using the Newton} \\ \text{binomial formula}\end{matrix}$$
$$= \sum_{k=0}^{\ell} \frac{\rho_2^k}{k!} \left(1 + \frac{\rho_1}{\rho_2}\right)^k = \sum_{k=0}^{\ell} \frac{(\rho_2+\rho_1)^k}{k!}$$

(∴)

$$P_B = \frac{(\rho_1 + \rho_2)^{\ell}}{\ell! \sum_{k=0}^{\ell} \frac{(\rho_2+\rho_1)^k}{k!}}$$

5) 본 방법 적용 예시 : 이 방법은 두 개 이상의 클래스를 가진 손실 대기열 시스템을 해결하는 데에 적용할 수 있다. 예를 들어, 유형 1 도착이 C1의 용량을, 유형 2 도착이 C2의 용량을, 유형 3 도착이 전체 용량 C 중에서 C3의 용량을 필요로 하는 일반적인 경우를 고려하자.

① 상태 공간(Ω) : 가능한 모든 시스템 상태의 집합이다. 각 상태는 세 가지 유형의 도착에 대한 수 (i, j, k)를 나타낸다.

$$\Omega = \{(i,j,k) \mid iC_1 + jC_2 + kC_3 \leq C\}$$

② 상태 확률 (P(i, j, k)) : 시스템이 특정 상태 (i, j, k)에 있을 확률을 나타낸다.

$$P(i,j,k) = \frac{1}{G(\Omega)} \frac{\rho_1^i \rho_2^j \rho_3^k}{i!j!k!}$$

③ 정규화 상수 (G(Ω))

$$G(\Omega) = \sum_{(i,j,k)\in\Omega} \frac{\rho_1^i \rho_2^j \rho_3^k}{i!j!k!}$$

Kaufman과 Robert의 반복적 방법으로 차단 확률 계산

1) 사용 중인 총 용량

$$\mathbf{n} \cdot \mathbf{b} = \sum_{i=1}^{N} n_i C_i$$

- C : 시스템 전체가 처리할 수 있는 최대 트래픽 양이나 서비스 용량
- $C_1, C_2, \ldots, C_N$: 각 서비스 클래스가 요구하는 링크 용량
- n_i : 각 클래스별로 현재 진행 중인 서비스의 수
- n : 각 클래스별 서비스의 수를 나타내는 벡터로, $n = \{n_1, n_2, \ldots, n_N\}$로 표현됨.
- b : 각 클래스별로 필요한 용량을 나타내는 벡터로, $b = \{C_1, C_2, \ldots, C_N\}$로 표현됨.

2) 정규화 상수 (G(Ω))

$$G(\Omega) = \sum_{\mathbf{n} \in \Omega} \frac{\rho_1^{n_1}}{n_1!} \frac{\rho_2^{n_2}}{n_2!} \cdots \frac{\rho_N^{n_N}}{n_N!}$$

3) Ω(c) : 시스템이 총 용량 c를 사용하고 있는 모든 상태의 집합

$$\Omega(c) = \{\mathbf{n} : \mathbf{n} \cdot \mathbf{b} = c\}, \text{for } c = 0, \ldots, C$$

4) G(c) : Ω(c)에 속하는 각 상태에 대한 비정규화된 확률

$$G(c) = G(\Omega(c)) = \sum_{\mathbf{n} \in \Omega(c)} \frac{\rho_1^{n_1}}{n_1!} \frac{\rho_2^{n_2}}{n_2!} \cdots \frac{\rho_N^{n_N}}{n_N!}$$

5) 차단 확률 P_{B_i} : i번째 클래스의 차단 확률. 이는 사용 가능한 용량이 C_i보다 적어서 i번째 클래스의 요청을 수용할 수 없는 상태에 있는 확률.

$$P_{B_i} = \frac{\sum_{c=C-C_i+1}^{C} G(c)}{\sum_{c=0}^{C} G(c)}, \text{for } i = 0, \ldots, N$$

6) Kaufman-Robert 반복 알고리즘 사용 : G(c)를 반복적으로 계산하여 각 c값에 대해 업데이트한다.

$$G(c) = \sum_{i=1}^{N} \frac{\rho_i C_i}{c} G(c - C_i)$$

(ρ_i는 i번째 클래스의 서비스 요청의 도착률과 서비스율의 비율)

7) 차단 확률 계산 예시

=> 두 개의 트래픽 클래스를 가지고 있다. C1 = 1 unit, C2 = 2 unit, C = 4 units, ρ1 = 2 Erl, and ρ2 = 1 Erl.

$$G(c) = \frac{2}{c}G(c-1) + \frac{2}{c}G(c-2)$$

$$G(c) = \sum_{i=1}^{N} \frac{\rho_i C_i}{c} G(c-C_i)$$

① 초기화

: G(0)을 1로 설정. (시스템이 비어 있는 상태의 확률)

: G(c)는 c가 0보다 작을 때 0으로 설정.

② 반복 계산

: c가 0부터 4까지의 다음 반복 계산을 수행.

$$G(1) = 2G(0) + 2G(-1) = 2$$
$$G(2) = G(1) + G(0) = 3$$
$$G(3) = \frac{2}{3}G(2) + \frac{2}{3}G(1) = \frac{10}{3}$$
$$G(4) = \frac{1}{2}G(3) + \frac{1}{2}G(2) = \frac{19}{6}$$

③ 차단 확률 P_{B_1}, P_{B_2}

$$P_{B_1} = \frac{G(4)}{G(0)+G(1)+G(2)+G(3)+G(4)} = \frac{19}{75} \approx 0.253$$
$$P_{B_2} = \frac{G(3)+G(4)}{G(0)+G(1)+G(2)+G(3)+G(4)} = 0.52$$