Chapter 8
Closed Queuing Network Models
Flexible Machining Systems
CONWIP (CONstant Work In Process)
Flexible Machining System

- **Components**
  - Computer-controlled machines (milling, drilling, etc.)
  - Related work stations (washing, inspection, measurement)
  - Automated material handling system
  - Loading/unloading station(s)
  - WIP storage

- **Centralized computer control**
- **Automatic tool changing**
- **Work pieces mounted on pallets in fixed position**
- **Manufacture a mix of related parts in medium volume**
CONWIP

- Hybrid between “push” and “pull” control strategies
  - Work is pulled into the system, pushed within the system
  - Benefits of kanban without as many card counts to determine
- A new job is released to shop only when an old one is completed
- Originally developed for flow lines
- WIP is easy to control and measure; throughput is observed rather than controlled directly
Closed Queuing Network Models

• $m$ service centers, $c_i$ servers at center $i$

• $h$ material handling devices

• $r$ part types, each with a set of service center sequences
  - In sequence $s$, type $j$ undergoes $v_j(s)$ operations at service centers
    $C_1^{(j)}(s), ..., C_{v_j(s)}^{(j)}(s)$ with expected proc. times $S_1^{(j)}(s), ..., S_{v_j(s)}^{(j)}(s)$
  - Time $T_{ij}$ to transport a part from s.c. $i$ to s.c. $j$

• Scenarios
  - Constant, $n$, number of aggregate parts (single chain)
  - Constant, $n_i$, number of type $i$ parts (multiple chain)
  - Mix: Aggregate types into classes; fix $n_j =$ number of class $j$ parts
Single-Class Closed Jackson Network  
(a.k.a. Gordon-Newell network)

• \( p_{ij} \) is the probability that a part leaving station \( i \) goes next to station \( j \): \( p_{ii} = 0 \) and for the load/unload station \( 0 \),

\[
p_{0j} = \gamma_j; \quad p_{i0} = 1 - \sum_{j=1}^{m} p_{ij}
\]

(note: these probabilities can be found in aggregation process using production ratios \( D_1 : \ldots : D_r \) instead of \( \lambda(l) \))

• Expected number of visits to s/c \( i \) is found from

\[
v_i = \gamma_i + \sum_{j=1}^{m} v_j p_{ji}, \; i = 1,\ldots,m; \quad v_0 = 1
\]

• Service requirement at s/c \( i \) is exponential with mean \( 1/\mu_i \), \( i = 0,\ldots,m \); when \( k \) parts at s/c \( i \), proc. rate is \( r_i(k) \mu_i \)

• FCFS protocol at each s/c
Continuous Time Markov Process

$N_i(t)$ is the number of parts at s/c $i$ at time $t$, $i = 0, 1, \ldots, m$

$N(t) = (N_0(t), \ldots, N_m(t))$ is a CTMC on state space

$\mathbb{S}_n = \{ \mathbf{k} : k_i \geq 0, \text{ integer}; \sum k_i = n \}$

Stationary distribution of $N$, $p(\mathbf{k}) = \lim_{t \to \infty} P[N(t) = \mathbf{k}]$ has a product-form solution: $p(\mathbf{k}) = K \prod_{i=0}^{m} p_i(k_i)$

where

$p_i(k_i) = \frac{v_i^{k_i}}{\prod_{j=1}^{k_i} \mu_{i} r_i(j)}$; $p_i(0) \equiv 1$

and $K$ is the normalizing constant that makes $\sum_{\mathbf{k} \in \mathbb{S}_n} p(\mathbf{k}) = 1$
Convolution Algorithm

In principle, the normalizing constant, $K$, can be found by solving

\[ K \sum_{k \in S_n} \prod_{i=0}^{m} p_i(k_i) = 1. \]

But a system with $n$ jobs and service centers 0, …, $m$ has \( \binom{n+m}{n} \) possible states.

The Convolution Algorithm (Algorithm 8.1, p. 371) is an efficient, recursive method for computing $p(k)$.

Throughput: \[ TH(n) = E\left[ \mu_0 r_0(N_0) \right] \]
Marginal Distribution Analysis

What is the effect on throughput of adding one more job? The key is that each job “sees” the system as it would be if this job did not exist! (may be more than one server at each s/c)

Let $p_i(k_i;n)$ be the probability that $k_i$ parts are at s/c $i$ if the system has $n$ parts total. The arrival rate to s/c $i$ is $TH(n) v_i$.

The probability that an arrival to s/c $i$ sees $k_i - 1$ parts there is $p_i(k_i - 1;n - 1)$. Therefore,

$$TH(n) v_i p_i(k_i - 1;n - 1) = \mu_i r_i(k_i) p_i(k_i;n)$$

rate into all states with $k_i$ at s/c $i =$ rate out of all such states
MDA (cont.)

The expected number of parts at s/c \( i \) when there are \( n \) total is

\[
E \left[ N_i (n) \right] = \sum_{k_i=1}^{n} k_i p_i (k_i; n) = TH (n) \sum_{k_i=1}^{n} \frac{\nu_i k_i}{\mu_i r_i (k_i)} p_i (k_i - 1; n - 1)
\]

\[
= TH (n) \sum_{k_i=0}^{n-1} \frac{\nu_i (k_i + 1)}{\mu_i r_i (k_i + 1)} p_i (k_i; n - 1)
\]

and from Little’s formula,

\[
E \left[ T_i (n) \right] = \sum_{k_i=0}^{n-1} \frac{k_i + 1}{\mu_i r_i (k_i + 1)} p_i (k_i; n - 1)
\]

The MDA Algorithm 8.2 is just a recursive application of these.
Mean Value Analysis

In most practical situations, we don’t need to know the entire probability distribution for the states of the CTMC. If each service center has a single server, we can use MVA (Algorithm 8.3) to get the performance measures:

1. Set $E\left[ N_i \left( 0 \right) \right] = 0, i = 1,\ldots,m$.

2. For $l = 1,\ldots,n$, compute:

   $E\left[ T_i \left( l \right) \right] = \left( E\left[ N_i \left( l - 1 \right) \right] + 1 \right) / \mu_i, i = 0,\ldots,m$

   $TH \left( l \right) = l / \left( \sum_{i=0}^{m} v_i E\left[ T_i \left( l \right) \right] \right)$

   $E\left[ N_i \left( l \right) \right] = v_i TH \left( l \right) E\left[ T_i \left( l \right) \right], i = 0,\ldots,m$
Other Product-Form Networks (Medhi)

• BCMP Networks (named for authors Baskett, Chandy, Muntz and Palacios of a 1975 article)
  – $k$ nodes
  – $R \geq 1$ classes of customers
  – Customers may change class

\[ p_{ir,js} = \Pr \left\{ \begin{array}{l}
\text{a customer of class } r \text{ completing service at node } i \\
\text{moves to node } j \text{ as a customer of class } s
\end{array} \right\} \]

\[ \mu_{ir} = \text{the mean service rate for class } r \text{ at node } i \]

Allowing class changes means that a customer can have different mean service rates for different visits to the same node.
BCMP (cont.)

- Nodes may be only of four types:
  - Single server, FCFS, where service times at node $i$ have the same distribution for all classes: $\exp(\mu_i)$
  - Single server with processor-sharing discipline. At any node, each class may have a different service time distribution but these distributions must be differentiable
    - processor sharing would be applicable to a computer system with multiple simultaneous users; not so applicable in manufacturing
  - Infinite number of servers (no queue, e.g. a self-service node). Each class may have a distinct differentiable service time dist’n.
  - Single server with *preemptive* last come first served discipline. A new arrival interrupts service and the displaced customer returns to the head of the queue (also known as HOL - Head-of-Line). Each class may have a distinct differentiable service time dist’n.
Multiple-Class Closed Networks

• These come in two types:
  
  – *Single-chain*, in which a job can change class
    
    • To model a central material handling system, a job changes class whenever it is moved from one service center to another: a class can be thought of as a type together with the number of operations that have been completed on it so far.
    
    • The total number, $n$, of jobs in the system is constant. When one part is completed, it may be replaced by a new part of any type (probabilistically or deterministically to maintain a specified product mix).
  
  – *Multiple-chain*, in which a number of part types that use the same pallet may be aggregated. A part changes class when it moves to a new service center. The number, $n_s$, of type $s$ pallets is constant.
Multi-Chain, Multi-Class Model

- Part types $R = \{1, ..., r\}$
- Pallet types $\{1, ..., p\}$
- Subsets of $R$: $R_1, ..., R_p$: any part type in $R_s$ uses a type $s$ pallet
- Part types in $R_s$ change among classes $\{(s,1), ..., (s, p_s)\}$
- $p_{ij}^{(s,l),(s,l')}$ = Probability that a part completing service at s/c $i$ as a class $(s,l)$ part goes next to s/c $j$ as class $(s,l')$
- Visit rates for $R_s$ satisfy:

$$v_i^{(s,l)} = \sum_{j=0}^{m} \sum_{l'=1}^{p_s} v_j^{(s,l')} p_{ji}^{(s,l),(s,l')}$$  \hspace{1cm} i = 1, ..., m; l = 1, ..., p_s$$

$$\sum_{l=1}^{p_s} v_0^{(s,l)} = 1$$
Multi-Chain, Multi-Class CTMC

• FCFS at each s/c
• Service times at s/c $i$ are exponential with mean $1/\mu_i$, independent of class
• Service rate of s/c $i$ multiplied by $r_i(k_i)$ when $k_i$ parts are there
• Let $N_i(t)$ be the number of parts at s/c $i$ at time $t$, $X_{ij}(t)$ be the class index of the part in the $j^{th}$ position of the queue at service center $i$ at time $t$
• Then $\{X(t), t \geq 0\}$ is a continuous-time Markov chain.
Performance Measures

- Throughput rate $TH_s(n)$ of “class” $s$ parts
- Mean number $E[N_{is}]$ of “class” $s$ parts at s/c $i$
- Average flow time $E[T_{is}]$ of “class” $s$ parts through s/c $i$

can be found along with marginal probabilities $p_{is}(k_{is})$ that there are $k_{is}$ “class” $s$ parts at s/c $i$ in steady state when there are $n = (n_1, \ldots, n_p)$ pallets of each type, using Multiclass Marginal Distribution Analysis (MDA) or Multiclass Mean Value Analysis (MVA) if $c_i = 1$. 
Multiclass MVA (Schweitzer-Bard)

Alternative to Algorithm 8.10

The following (taken from Suri & Hildebrant (1984)) applies if \( c_i = 1 \) but in the article they also show how to approximately extend it to several machines at a station.

Initialize: \( E\left[N_{i,s}(\mathbf{n})\right] = n_s / (m+1) \), for \( i = 0, 1, \ldots, m \) and \( s = 1, \ldots, p \)

or, if \( 1/\mu_0 = 0, E\left[N_{0,s}(\mathbf{n})\right] = 0 \) and \( E\left[N_{i,s}(\mathbf{n})\right] = n_s / m \), for \( i = 1, \ldots, m \)

Repeat: For \( i = 0, 1, \ldots, m \) and \( s = 1, \ldots, p \),

\[
E\left[T_{i,s}(\mathbf{n})\right] = \left(1 + \frac{n_s - 1}{n_s} E\left[N_{i,s}(\mathbf{n})\right] + \sum_{r \neq s} E\left[N_{i,r}(\mathbf{n})\right]\right) \frac{1}{\mu_i}
\]

For \( s = 1, \ldots, p \), \( TH_s(\mathbf{n}) = n_s / \sum_{i=0}^{m} v_i^{(s)} E\left[T_{i,s}(\mathbf{n})\right] \)

For \( i = 0, 1, \ldots, m \) and \( s = 1, \ldots, p \), \( E\left[N_{i,s}(\mathbf{n})\right] = v_i^{(s)} TH_s(\mathbf{n}) E\left[T_{i,s}(\mathbf{n})\right] \)

Until: Successive iterations yield small enough change in \( E\left[N_{i,s}(\mathbf{n})\right] \)
Throughput Properties

1. $TH_s(n)$ is increasing in $n_s$ for each $s$.
2. $TH_l(n)$ need not increase in $n_s$ for $s \neq l$
3. $TH(n)$ (total) need not increase in $n_s$.
4. Pooling of service centers need not increase the total throughput.

(Note: Some of these characteristics follow from assumption that system will be operated “blindly” without good service protocols, feedback or part input controls.)
Throughput Property 3

$TH(n)$ (total) need not increase in $n_s$.

**Example 8.7:**

Two-class closed Jackson network with s/c’s $\{0, 1, \ldots, m\}$; Transition probability matrix for class 1 is $\mathbf{P} = [p_{ij}]$.

For class 2, for some $0 < q < 1$, transition probabilities are:

\[
\hat{p}_{ij} = p_{ij} + (1 - q) p_{i0} p_{0j}, \quad i, j = 1, \ldots, m
\]

\[
\hat{p}_{i0} = q p_{i0}
\]

\[
\hat{p}_{0j} = p_{0j}
\]

Then

\[
v_i^{(2)} = v_i^{(1)}/q \quad \text{(class 2 spends more time in system)}
\]
Throughputs for Example

Compare with throughput of a single-class network of \( n_1 + n_2 \) type 1 parts, \( \hat{TH}(n_1 + n_2) \).

If \( 1/\mu_0 = 0 \), can show that

\[
TH_1(n_1, n_2) = \frac{n_1}{n_1 + n_2} \hat{TH}(n_1 + n_2)
\]

\[
TH_2(n_1, n_2) = \frac{qn_2}{n_1 + n_2} \hat{TH}(n_1 + n_2)
\]

\[
TH(n_1, n_2) = q \hat{TH}(n_1 + n_2) + \frac{(1-q)n_1}{n_1 + n_2} \hat{TH}(n_1 + n_2)
\]

Then increasing \( n_1 \) increases the total throughput, but increasing \( n_2 \) can decrease the total throughput if \( q \) is small.
A Remedy

With multiple classes, adopt a single chain policy: instead of always replacing a completed class $l$ part with a raw class $l$ part, use a mixed feedback policy. If $d_1, \ldots, d_p$ are the desired production ratios, then

- Replace with a class $r$ part with probability $d_r$, $r=1, \ldots, p$.
- Or use a predefined loading sequence of part types such that the long run ratios of the part types loaded is $d_1: \ldots : d_p$.

Then $TH_l = d_l TH$, where $TH$ is the throughput of a single-class (aggregated) network.

And since $TH$ increases in the total number of parts, each $TH_l$ must increase, too, as more parts are added.
Duenyas’ (1994) simulation study of several small queuing networks indicates that a multiple-chain policy can achieve specified throughput targets with less WIP (fewer parts in the system) than a single-chain policy.

His example:
The 50-50 throughput mix could be achieved in a single-chain policy by releasing parts in the order ABAB…

However, if s/c 2 had a failure, then the queue of type A parts in front of s/c 2 would increase, while type B parts would be processed quickly. Since the total number of parts in the system is fixed, eventually, all of them would be type A parts waiting for s/c 2, and s/c’s 3 and 4 would be idle.

A multiple chain policy would avoid this by limiting the number of type A parts, and allowing production of type B to continue.

In general, if the different part types have different bottleneck s/c’s, a multiple chain policy seems to work better.
Congratulations to the graduates!

Have a great summer!