Advanced Queueing Theory

• Networks of queues
  (reversibility, output theorem, tandem networks, partial balance, product-form distribution, blocking, insensitivity, BCMP networks, mean-value analysis, Norton's theorem, sojourn times)

• Analytical-numerical techniques
  (matrix-analytical methods, compensation method, error bound method, approximate decomposition method)

• Polling systems
  (cycle times, queue lengths, waiting times, conservation laws, service policies, visit orders)

Richard J. Boucherie
department of Applied Mathematics
University of Twente
http://wwwwhome.math.utwente.nl/~boucherierj/onderwijs/Advanced_Queueing_Theory/AQT.html
M/G/1 queue with server vacations

- Polling system
Advanced Queueing Theory
Today (lecture 8): Vacation models

• J.A.C. Resing. Polling systems and multitype branching processes, Queueing Systems, 13, p 409 – 426
• I. Adan: Queueing Systems, lecture notes

• M/G/1 queue, PK formula
• M/G/1 queue with vacations
• M/G/1 queue with Generalized vacations
• Polling model
**M/G/1 queue**

- Poisson arrival process rate \( \lambda \), single server, General service times, mean \( 1/\mu = \text{E}(B) \)
- Service time distribution \( F_B(.) \), density \( f_B(.) \)
- State: pair \( (n,x) \), \( n = \# \text{customer} \), \( x = \text{received service time} \)

- \# customers left behind by \( k \)-th departing cust.

\[
d_n = \lim_{k \to \infty} P(L^d_k = n)
\]

- \# customers in the system at time \( t \)

\[
p_n = \lim_{t \to \infty} P(L(t) = n)
\]

- \# customers seen by \( k \)-th arriving customer

\[
a_n = \lim_{k \to \infty} P(L^a_k = n)
\]

- PASTA & frequency argument:

\[
a_n = d_n = p_n
\]
**M/G/1 queue**

- # arrivals during k-th service time \( A_{k+1} \)

- \( L_{k+1}^d = L_k^d - 1 + A_{k+1} \) \( \quad L_k^d > 0 \)
  \[ L_{k+1}^d = A_{k+1}, \quad L_k^d = 0 \]

- Embedded Markov chain at departure epochs

\[
P = \begin{pmatrix}
\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots \\
\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \cdots \\
0 & \alpha_0 & \alpha_1 & \alpha_2 & \cdots \\
0 & 0 & \alpha_0 & \alpha_1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}
\]

\[ p_{i,j} = P(L_{k+1}^d = j | L_k^d = i) \]

- pr \( n \) arrivals during service

\[
\alpha_n = \int_{t=0}^{\infty} \frac{(\lambda t)^n}{n!} e^{-\lambda t} f_B(t) dt
\]

\[ d_n = P(L^d = n) = \lim_{k \to \infty} P(L_k^d = n) \]
**M/G/1 queue**

- **Equilibrium equations**
  
  \[ d_n = d_{n+1} \alpha_0 + d_n \alpha_1 + \cdots + d_1 \alpha_n + d_0 \alpha_n \]
  \[ = \sum_{k=0}^{n} d_{n+1-k} \alpha_k + d_0 \alpha_n, \quad n = 0, 1, \ldots \]

- **Gen. functions, for z<1,**

  \[ P_L(z) = \sum_{n=0}^{\infty} d_n z^n, \quad P_A(z) = \sum_{n=0}^{\infty} \alpha_n z^n, \]

- **From equil. eq.**

  \[ P_{Ld}(z) = \sum_{n=0}^{\infty} \left( \sum_{k=0}^{n} d_{n+1-k} \alpha_k + d_0 \alpha_n \right) z^n \]
  \[ = z^{-1} \sum_{n=0}^{\infty} \sum_{k=0}^{n} d_{n+1-k} z^{n+1-k} \alpha_k z^k + \sum_{n=0}^{\infty} d_0 \alpha_n z^n \]
  \[ = z^{-1} \sum_{k=0}^{\infty} \sum_{n=k}^{\infty} d_{n+1-k} z^{n+1-k} \alpha_k z^k + d_0 P_A(z) \]
  \[ = z^{-1} \sum_{k=0}^{\infty} \alpha_k z^k \sum_{n=k}^{\infty} d_{n+1-k} z^{n+1-k} + d_0 P_A(z) \]
  \[ = z^{-1} P_A(z)(P_{Ld}(z) - d_0) + d_0 P_A(z). \]
M/G/1 queue

\[
P_L^d(z) = \frac{(1-\rho)P_A(z)(1-z)}{P_A(z) - z}
\]

- Furthermore

\[
P_A(z) = \sum_{n=0}^{\infty} \sum_{t=0}^{\infty} \frac{(\lambda t)^n}{n!} e^{-\lambda t} f_B(t) dt z^n
\]

- Pollaczek-Khintchine formula for queue length

\[
P_A(z) = \int_{0}^{\infty} \sum_{n=0}^{\infty} \frac{(\lambda tz)^n}{n!} e^{-\lambda t} f_B(t) dt
\]

- By analogy for sojourn time

\[
S(s) = \frac{(1-\rho)\bar{B}(s)s}{\lambda\bar{B}(s) + s - \lambda}
\]

- And for Waiting time

\[
\bar{W}(s) = \frac{(1-\rho)s}{\lambda\bar{B}(s) + s - \lambda}
\]
Advanced Queueing Theory
Today (lecture 8): Vacation models

- J.A.C. Resing. Polling systems and multitype branching processes, Queueing Systems, 13, p 409 – 426
- I. Adan: Queueing Systems, lecture notes

- M/G/1 queue, PK formula
- M/G/1 queue with vacations
- M/G/1 queue with Generalized vacations
- Polling model
**M/G/1 queue with server vacations**

- Consider M/G/1 queue
- Server takes a vacation (general distribution) **when the system becomes idle**
- Upon return from vacation, if system idle new vacation otherwise serve until system idle: exhaustive service
- No preemption
- Queue length? Sojourn time?
- Mild conditions: in distribution
  \[ N = N_{M/Q/1} + N_I \]
  \[ V = V_{M/Q/1} + V_I \]

- \( N_{M/Q/1} \) number at arbitr moment in corresp M/G/1
- \( N_I \) number at arbitr moment in non-serving interval

Independent r.v.
M/G/1 queue with server vacations

• Polling system with exhaustive service
Ancestral line

- $I_0$ group of customers
- $I_1$ set of customers who arrive while members of $I_0$ are being served: \textit{first generation offspring} of $I_0$
- $I_k$, $k>1$ set of customers who arrive while members of $I_{k-1}$ are being served: \textit{k-th generation offspring} of $I_0$
- $\bigcup_{k=0}^{\infty} I_k$ \textit{ancestral line} of $I_0$

- For us: $I_0$ single customer, or set of customers arriving during some vacation
- Vacation customer: arrive while server is on vacation
- To each customer, $C$, corresponds a unique, vacation customer, $A$, such that $C$ is in ancestral line of $A$: $A$ is \textit{ancestor} of $C$. 
Exhaustive service

• No preemption: # cust under FIFO = # under LIFO
• Assume LIFO

• Tag arbitrary customer C upon departure
  let A be its ancestor
• present at departure of C are
  all vacation customers that arrived before A,
  and all remaining customers in ancestral line of A

• A has started busy period,
  C is arbitrary cust in this busy period, this busy period
  equivalent to standard busy period, thus number of
  customers in ancestral line of A left behind by C =
  number of customers left behind in standard M/G/1
  queue.
Exhaustive service

- present at departure of C are all vacation customers that arrived before A, and all remaining customers in ancestral line of A

- C is arbitrary customer, all vacation customers are iid, therefore ancestor of C corresponds to arbitrarily selected vacation cust. Thus : number of vacation customers left behind = number of customers at arbitrary epoch during vacation.

- Theorem: Queue length decomposition $N = N_{M/G/1} + N_I$ where equality is in distribution, and
  - $N$: queue length at arbitrary epoch
  - $N_{M/G/1}$: queue length at arbitrary epoch in corresponding $M/G/1$
  - $N_I$: queue length at arbitrary epoch in vacation period
  - $N_{M/G/1}$ and $N_I$ are independent r.v.
Exhaustive service

- $N_I$ distrib as # cust seen by arbitr arrival in vacation period
- $N_{\text{end},k}$ # customers end k-th vac period, $N_{\text{end}}$ generic rv.

Lemma:

$$P(N_I = s) = \frac{1-P(N_{\text{end}} \leq s)}{EN_{\text{end}}}$$

$$E_Z^{N_I} = \frac{1-E_Z^{N_{\text{end}}}}{(1-z)EN_{\text{end}}}$$

Proof
Exhaustive service

- Alternative formulation:

\[ \psi(.) = \text{p.g.f. stat distrib } \# \text{ cust random dept cust leaves behind or finds or at arbitrary time} \]

- \( \pi(.) = \text{p.g.f. idem in corresp M/G/1 queue} \)

- \( \alpha(.) = \text{p.g.f. } \# \text{ arrivals during vacation period } = \text{Vac}(\lambda - \lambda z) \)

\[
\pi(z) = P_{Ld}(z) = \frac{(1 - \rho)B(\lambda - \lambda z)(1 - z)}{B(\lambda - \lambda z) - z}.
\]

\[
\psi(z) = \frac{1 - \alpha(z)}{(1 - z)\alpha'(1)} \pi(z)
\]
Advanced Queueing Theory
Today (lecture 8): Vacation models

• J.A.C. Resing. Polling systems and multitype branching processes, Queueing Systems, 13, p 409 – 426
• I. Adan: Queueing Systems, lecture notes

• M/G/1 queue, PK formula
• M/G/1 queue with vacations
• M/G/1 queue with Generalized vacations
• Polling model
Generalized vacations: assumptions

- Server can take vacation at any time: vacation = server unavailable
- Service time independent of sequence of vacation periods preceding that service time.
- Order of service independent of service times.
- Service non-preemptive
- Rules that govern when server begins and ends vacations do not anticipate future jumps of the arrival process.
Generalized vacations: examples

• Standard vacation model: vacation upon idling
• N-policy: server waits until exactly N customers present before starting service, then work continues until system empty
• M/G/1 with gated vacations: when server returns from vacation, he accepts only those customers waiting upon return, service of other customers deferred to next visit
• Limited service: serve at most k customers
• Polling model
• Priority system
Generalized vacations

- Decomposition property holds for any vacation system under assumptions stated.

- Theorem: Consider a random (tagged) customer C. Let A the ancestor, I₀ the set of vacation customers who arrived during the same vacation to which A arrived. Let I the ancestral line of I₀, and let X the number of members of I present in system when C departs. X has p.g.f.

\[
\psi(z) = \frac{1 - \alpha(z)}{(1 - z)\alpha'(1)} \pi(z)
\]

- Proof:
**Generalized vacations**

- Assume that the number of customers that arrive during a vacation is independent of the number of customers present in the system when vacation began.

- **Theorem:** Under this assumption

  \[ \psi(z) = \zeta(z) \frac{1 - \alpha(z)}{(1 - z)\alpha'(1)} \pi(z) \]

- **Proof:** LIFO, p.g.f. of number of customers already present at beginning of vacation. Independence yields product.
Generalised vacations

• Theorem: Queue length decomposition \( N = N_{M/G/1} + N_I \)
  where equality is in distribution, and
  \( N \): queue length at arbitrary epoch
  \( N_{M/G/1} \): queue length at arbitrary epoch in corresponding M/G/1
  \( N_I \): queue length at arbitrary epoch in vacation period
  \( N_{M/G/1} \) and \( N_I \) are independent r.v.

• Theorem: Work decomposition \( V = V_{M/G/1} + V_I \)
  where equality is in distribution, and
  \( V \): work at arbitrary epoch
  \( V_{M/G/1} \): work at arbitrary epoch in corresponding M/G/1
  \( V_I \): work at arbitrary epoch in vacation period
  \( V_{M/G/1} \) and \( V_I \) are independent r.v.
Generalized vacations

• Assume that the number of customers that arrive during a vacation is independent of the number of customers present in the system when vacation began.

• Theorem: Under this assumption

\[ \psi(z) = \zeta(z) \frac{1 - \alpha(z)}{(1 - z)\alpha'(1)} \pi(z) \]

• Proof: LIFO, p.g.f. of number of customers already present at beginning of vacation. Independence yields product.
Generalized vacations: Gated service

• Gated service: At time $t_n$ server finds $X_n$ customers. Gate closes. Server serves those $X_n$ then takes vacation of length $V_n$ and returns at time $t_{n+1}$

• Recall

Recursion

$$X_{n+1} = A\left(\sum_{i=1}^{X_n} B_i\right) + A(V_n)$$

Markov chain; from recursion:

$$E[z^{X_{n+1}}] = V(\lambda - \lambda z) E[(B(\lambda - \lambda z))^X_n]$$
Generalized vacations: Gated service

- Assume limiting distribution of $X_n \xrightarrow{n} \infty$ exists, with pgf $X(z)$, then

$$X(z) = V(\lambda - \lambda z)X(B(\lambda - \lambda z))$$

Let $$h(z) := B(\lambda - \lambda z)$$

then

$$X(z) = V(\lambda(1 - z))X(h(z))$$

$$= V(\lambda(1 - z))V(\lambda(1 - h^{(1)})(z))X(h^{(2)}(z))$$

$$= \prod_{j=0}^{M} V(\lambda(1 - h^{(j)})(z))X(h^{(M+1)}(z))$$
Generalized vacations: Gated service

• Let $M \to \infty$ in

$$X(z) = \prod_{j=0}^{M} V(\lambda(1 - h^{(j)}(z))) X(h^{(M+1)}(z))$$

resulting infinite product exists if $\rho < 1$, and

$$h^{(M+1)}(z) \to 1$$

• Hence, if $\rho < 1$

$$X(z) = \prod_{j=0}^{\infty} V(\lambda(1 - h^{(j)}(z)))$$
Convergence

\[ |1 - h^{(j)}(z)| = |1 - h(h^{(j-1)}(z))| \leq \rho |1 - h^{(j-1)}(z)| \]

hence, indeed if \( \rho < 1 \)

\[ h^{(M+1)}(z) \rightarrow 1 \]

\[ \bullet \text{ Observe} \]

\[ |1 - V(\lambda(1 - h^{(j)}(z)))| \leq \lambda E[V] |1 - h^{(j)}(z)| \leq \lambda E[V] \rho^j |1 - z| \]

\[ \bullet \text{ So that infinite product indeed converges} \]
Interpretation: Branching processes (Wolff, sec 3-9)

• Let $Y_r$ (i.i.d) be the first generation off-spring of individual $r$

$$P(Y_r = j) = \gamma_j, \, j = 0, 1, ...$$

• $X_n$ n-th generation off-spring

$$p(X_{n+1} = j \mid X_n = 1) = p_{1j} = \gamma_j$$

$$p(X_{n+1} = j \mid X_n = i) = p_{ij} = \gamma_j^{*i}, i, j = 0, 1, ...$$

• Pgf n-th generation off-spring individual

$$A_n(z) = \mathbb{E}\{z^{X_n} \mid X_0 = 1\}$$

$$A_0(z) = 1$$

$$A_1(z) = A(z) = \sum_{j=0}^{\infty} \gamma_j z^j$$
Branching processes (Wolff, sec 3-9)

- $X_{n+1}$ is sum of descendants of the $j$ individuals of the first generation:

$$E\{z^{X_{n+1}} \mid X_1 = j, X_0 = 1\} = E\{z^{X_{n+1}} \mid X_1 = j\} = [A_n(z)]^j$$

$$A_{n+1}(z) = E\{[A_n(z)]^j \mid X_0 = 1\} = A(A_n(z))$$
Generalized vacations: Gated service

• Gated service:
  recall: \( \bar{B}(\lambda - \lambda z) = P_A(z) \)

Define \( R^{(1)}(z) = \bar{B}(\lambda - \lambda z) \)

\[ R^{(k)}(z) = R^{(1)}(R^{(k-1)}(z)), \quad k \geq 2 \]

is p.g.f. number of the k-th generation offspring

\[ \varsigma(z) = \prod_{k=1}^{\infty} \alpha(R^{(k)}(z)) \]

first previous gated period, second previous gated period
Generalized vacations

• Server can take vacation at any time
• Service time independent of sequence of vacation periods preceding that service time.
• Order of service independent of service times.
• Service non-preemptive
• Rules that govern when server begins and ends vacations do not anticipate future jumps of the arrival process.
• Number of customers that arrive during a vacation is independent of the number of customers present in the system when vacation began.

\[ \psi(z) = \zeta(z) \frac{1 - \alpha(z)}{(1 - z)\alpha'(1)} \pi(z) \]
Advanced Queueing Theory
Today (lecture 8): Vacation models

• J.A.C. Resing. Polling systems and multitype branching processes, Queueing Systems, 13, p 409 – 426
• I. Adan: Queueing Systems, lecture notes

• M/G/1 queue, PK formula
• M/G/1 queue with vacations
• M/G/1 queue with Generalized vacations
• Polling model
Polling models

- $N$ infinite buffer queues, $Q_1, \ldots, Q_N$
- Service time distribution at queue $j$: $B_j(\cdot)$, mean $\beta_j$, LST $\beta_j(\cdot)$
- Poisson arrivals to queue $j$ at rate $\lambda_j$
- Single server in cyclic order
- Switch over times: random variable $S_j$, mean $\sigma_j$, LST $\sigma_j(\cdot)$
Polling models: 2 queues with gated service

• $X_{ij} =$ # customers at $Q_j$ when server polls $Q_i$
• $Y_{ij} =$ # customers at $Q_j$ when server leaves $Q_i$

• Generating functions

\[ F_i(z_1, z_2) = E\{z_1^{X_{i1}} z_2^{X_{i2}}\}, \quad i = 1, 2 \]
\[ G_i(z_1, z_2) = E\{z_1^{Y_{i1}} z_2^{Y_{i2}}\}, \quad i = 1, 2 \]

• Clearly, with $\sigma_i(.)$ the LST of switchover time after visit to $Q_i$

\[ F_2(z_1, z_2) = G_1(z_1, z_2)\sigma_1(\lambda_1 (1 - z_1) + \lambda_2 (1 - z_2)), \]
\[ F_1(z_1, z_2) = G_2(z_1, z_2)\sigma_2(\lambda_1 (1 - z_1) + \lambda_2 (1 - z_2)) \]

in words: the number when server polls $Q_2$ are the sums of the numbers when server left $Q_1$ + the numbers of arrivals during the subsequent switch
Polling models: 2 queues

Lemma

\[ G_1(z_1, z_2) = F_1(B_1(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)), z_2) \]

- With the additional relation we have four formulas

\[ F_1(z_1, z_2) = G_2(z_1, z_2)\sigma_2(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)) \]

\[ G_2(z_1, z_2) = F_2(z_1, B_2(\lambda_1(1 - z_1) + \lambda_2(1 - z_2))) \]

\[ F_2(z_1, z_2) = G_1(z_1, z_2)\sigma_1(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)) \]

\[ G_1(z_1, z_2) = F_1(B_1(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)), z_2) \]

- And we may express, via iteration

\[ \Sigma = \lambda_1(1 - z_1) + \lambda_2(1 - z_2) \]

\[ F_1^{(k+1)}(z_1, z_2) = \sigma_2(\Sigma)\sigma_1(\lambda_1(1 - z_1) + \lambda_2(1 - B_2(\Sigma))) \]

\[ \times F_1^{(k)}(B_1(\lambda_1(1 - z_1) + \lambda_2(1 - B_2(\Sigma))), B_2(\Sigma)) \]
Polling: offspring

- While served at queue $i$, customer is replaced by random population with p.g.f.

- Exhaustive

$$h_i(z_1, ..., z_N) = \theta_i \left( \sum_{j \neq i} \lambda_j (1 - z_j) \right)$$

- gated

$$h_i(z_1, ..., z_N) = \beta_i \left( \sum_j \lambda_j (1 - z_j) \right)$$
Polling models: 2 queues

- Starting with the initial relations

\[ F_1(z_1, z_2) = G_2(z_1, z_2)\sigma_2(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)) \]

\[ G_2(z_1, z_2) = F_2(z_1, h_2(z_1, z_2)) \]

\[ F_2(z_1, z_2) = G_1(z_1, z_2)\sigma_1(\lambda_1(1 - z_1) + \lambda_2(1 - z_2)) \]

\[ G_1(z_1, z_2) = F_1(h_1(z_1, z_2), z_2) \]

- Iterating, we get

\[ \Sigma = \lambda_1(1 - z_1) + \lambda_2(1 - z_2) \]

\[ F_1^{(k+1)}(z_1, z_2) = \sigma_2(\Sigma)\sigma_1(\lambda_1(1 - z_1) + \lambda_2(1 - h_2(z_1, z_2))) \]

\[ F_1^{(k)}(h_1(z_1, h_2(z_1, z_2)), h_2(z_1, z_2)) \]
Polling: offspring

- Assumption
  If the server arrives at $Q_i$ to find $k_i$ customers, then during the course of the servers visit, each of these $k_i$ customers is effectively replaced in an iid manner by a random population with p.g.f. $h_i(z_1, \ldots, z_N)$

- Examples: exhaustive, gated
- Not included: 1-limited, because all but first have pgf $s_i$

- Bernoulli-type. When server arrives all customers present are handled as follows: customer is served, and all offspring served with probability $p_i$. 
For the Bernoulli-type service discipline the p.g.f. $h_i(s_1, \ldots, s_N)$ is given by

$$h_i(s_1, \ldots, s_N) = \Phi_{p_i,i} \left( \sum_{j \neq i} \lambda_j (1 - s_j) \right)$$

$$+ \frac{(1 - p_i) \beta_i \left( \sum_j \lambda_j (1 - s_j) \right)}{s_i - p_i \beta_i \left( \sum_j \lambda_j (1 - s_j) \right)} \left( s_i - \Phi_{p_i,i} \left( \sum_{j \neq i} \lambda_j (1 - s_j) \right) \right),$$

where $\Phi_{p_i,i}(s)$ is the unique solution of

$$\Phi_{p_i,i}(s) = p_i \beta_i (s + \lambda_i (1 - \Phi_{p_i,i}(s))).$$