Netherlands Organisation for Scientific Research

Finite buffer fluid networks with overflows

Stijn Fleuren, Erjen Lefeber, Yoni Nazarathy

IWAP 2012, Jerusalem

Technische Universiteit **Eindhoven** University of Technology

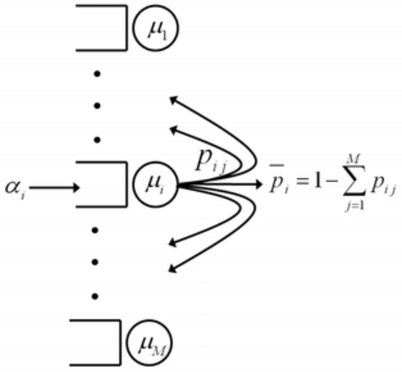
Where innovation starts

June 11, 2012

Open Jackson Networks

Jackson (1957); Goodman, Massey (1984); Chen, Mandelbaum (1991)

Jackson 1957



Problem data

α, μ, P

Traffic equations (stable case)

$$\lambda_i = \alpha_i + \sum_{j=1}^{N} \lambda_j p_{ji}$$

$$\lambda = \alpha + P'\lambda$$

$$\lambda = (I - P')^{-1}\alpha$$

Open Jackson Networks

Theorem (Jackson 1957)

Given an $(M/M/1)^N$ system where every node can be filled and drained, let $\lambda = [\lambda_1, \ldots, \lambda_N]'$ be the solution of the throughput equation

$$\lambda = \alpha + P'\lambda$$

If $\rho_i = \lambda_i/\mu_i$ and $\rho_i < 1$ for all i, then

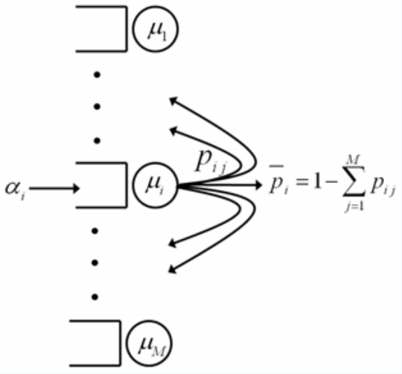
$$\lim_{t\to\infty} P(X_1(t) = n_1, \ldots X_N(t) = n_N) = \prod_{i=1}^{N}(1-\rho_i)\rho_i^{n_i}$$

for all integers $n_i \geq 0$.

Open Jackson Networks

Jackson (1957); Goodman, Massey (1984); Chen, Mandelbaum (1991)

Jackson 1957



Problem data

α, μ, P

Traffic equations (general)

$$\lambda_i = \alpha_i + \sum_{j=1}^{N} \min(\lambda_j, \mu_j) p_{ji}$$

$$\lambda = \alpha + P' \min(\lambda, \mu)$$

Open Jackson Networks

Theorem (Goodman, Massey 1984)

Given an $(M/M/1)^N$ system where every node can be filled and drained, let $\lambda = [\lambda_1, \ldots, \lambda_N]'$ be the solution of the throughput equation

$$\lambda = \alpha + P' \min(\lambda, \mu)$$

If $\rho_i = \lambda_i / \mu_i$ and $U = \{i \mid \rho_i < 1\}$, then

$$\lim_{t\to\infty} P(X_i(t) = n_i; i \in U) = \prod_{i \in U} (1 - \rho_i) \rho_i^{n_i}$$

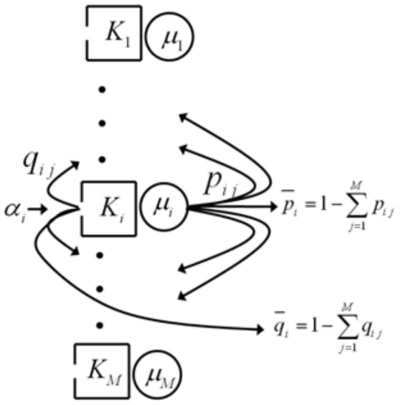
for all integers $n_i \geq 0$ with $i \in U$. Moreover, if $j \notin U$ then

$$\lim_{t\to\infty} P(X_j(t) = n) = 0$$

for all integers $n \geq 0$

Finite buffers and overflows

Network



Problem data

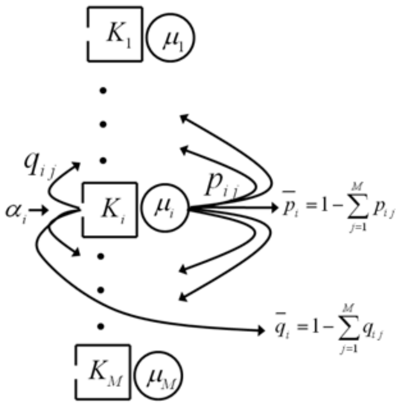
α, μ, P, Q, K

Our contribution (in progress)

- Limiting traffic equations
- Efficient algorithm for unique solution
- Limiting deterministic trajectories
- Limiting sojourn time distribution

Finite buffers and overflows

Network



Problem data

α, μ, P, Q, K

Our contribution (in progress)

- Limiting traffic equations
- Efficient algorithm for unique solution
- Limiting deterministic trajectories
- Limiting sojourn time distribution

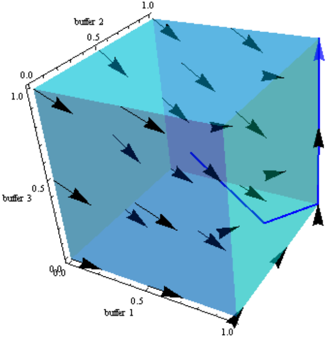
Scaling yields a fluid system

A sequence of systems: $N = 1, 2, \ldots$

$$\alpha^N = N\alpha \qquad \mu^N = N\mu \qquad K^N = NK$$

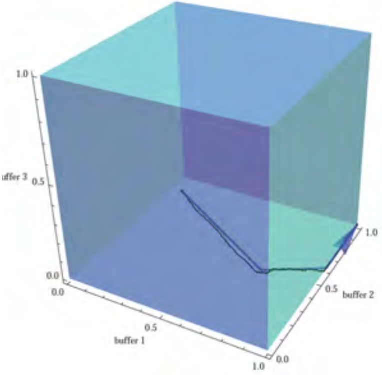
Make the jobs fast and the buffers big by taking $N \to \infty$.

The proposed limiting model is a deterministic fluid system



Technische Universiteit Eindhoven University of Technology

Fluid trajectories as an approximation



$$\lim_{N\to\infty} \sup_{t} \left\{ \left| \frac{X^N(t)}{N} - x(t) \right| \right\} = 0$$

Traffic equations (at equilibrium point)

outflow rate: $\min(\lambda, \mu)$
overflow rate: $\lambda - \min(\lambda, \mu) = \max(0, \lambda - \mu)$

Traffic equations

$$\lambda_i = \alpha_i + \sum_{j=1}^{N} \min(\lambda_j, \mu_j) p_{ji} + \sum_{j=1}^{N} \max(0, \lambda_j - \mu_j) q_{ji}$$

or

$$\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$$

Question

How to (efficiently) solve traffic equations for given α, μ, P, Q, K?

Traffic equations (at equilibrium point)

outflow rate: $\min(\lambda, \mu)$
overflow rate: $\lambda - \min(\lambda, \mu) = \max(0, \lambda - \mu)$

Traffic equations

$$\lambda_i = \alpha_i + \sum_{j=1}^{N} \min(\lambda_j, \mu_j) p_{ji} + \sum_{j=1}^{N} \max(0, \lambda_j - \mu_j) q_{ji}$$

or

$$\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$$

Question

How to (efficiently) solve traffic equations for given α, μ, P, Q, K?

Traffic equations (at equilibrium point)

outflow rate: $\min(\lambda, \mu)$
overflow rate: $\lambda - \min(\lambda, \mu) = \max(0, \lambda - \mu)$

Traffic equations

$$\lambda_i = \alpha_i + \sum_{j=1}^{N} \min(\lambda_j, \mu_j) p_{ji} + \sum_{j=1}^{N} \max(0, \lambda_j - \mu_j) q_{ji}$$

or

$$\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$$

Question

How to (efficiently) solve traffic equations for given α, μ, P, Q, K?

Traffic equations

$$\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$$

Let $w = \lambda - \min(\lambda, \mu)$ and $z = \mu - \min(\lambda, \mu)$. Then $\lambda = w - z + \mu$ and

$$w \geq 0 \qquad z \geq 0 \qquad w'z = 0$$

Furthermore we obtain for the traffic equation

$$\begin{aligned} w - z + \mu &= \alpha + P'(\mu - z) + Q'w \\ (I - Q')w &= \alpha - (I - P')\mu + (I - P')z \\ w &= \underbrace{(I - Q')^{-1}(\alpha - (I - P')\mu)}_{q} + \underbrace{(I - Q')^{-1}(I - P')}_{M} z \end{aligned}$$

Traffic equations

$$\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$$

Let $w = \lambda - \min(\lambda, \mu)$ and $z = \mu - \min(\lambda, \mu)$. Then $\lambda = w - z + \mu$ and

$$w \geq 0 \qquad z \geq 0 \qquad w'z = 0$$

Furthermore we obtain for the traffic equation

$$\begin{aligned} w - z + \mu &= \alpha + P'(\mu - z) + Q'w \\ (I - Q')w &= \alpha - (I - P')\mu + (I - P')z \\ w &= \underbrace{(I - Q')^{-1}(\alpha - (I - P')\mu)}_{q} + \underbrace{(I - Q')^{-1}(I - P')}_{M} z \end{aligned}$$

TU/e Technische Universiteit Eindhoven University of Technology

Linear Complementarity Problem

LCP

LCP(q, M): Find z, w such that

$$w - Mz = q \qquad w, z \geq 0 \qquad w'z = 0$$

For our system: $q = (I - Q')^{-1}(\alpha - (I - P')\mu)$, $M = (I - Q')^{-1}(I - P')$

Theorem

LCP(q, M) has unique solution for all q iff M is a P-matrix, i.e. determinants of all $2^N - 1$ principal submatrices are positive

Observation

No polynomial time algorithm (yet) exists for solving the P-matrix LCP

Technische Universiteit Eindhoven University of Technology

Linear Complementarity Problem

LCP

LCP(q, M): Find z, w such that

$$w - Mz = q \qquad\qquad w, z \geq 0 \qquad\qquad w'z = 0$$

For our system: $q = (I - Q')^{-1}(\alpha - (I - P')\mu)$, $M = (I - Q')^{-1}(I - P')$

Theorem

LCP(q, M) has unique solution for all q iff M is a P-matrix, i.e. determinants of all $2^N - 1$ principal submatrices are positive

Observation

No polynomial time algorithm (yet) exists for solving the P-matrix LCP

TU/e Technische Universiteit Eindhoven University of Technology

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

Technische Universiteit Eindhoven University of Technology

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

Technische Universiteit Eindhoven University of Technology

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

Technische Universiteit Eindhoven University of Technology

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

An efficient algorithm: No overflow

Algorithm of Goodman and Massey (1984)

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu)$

Observation: If we would know the stable and unstable nodes, we can solve for λ.

Step 1: Assume all queues are unstable, i.e. output rate μ_i, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an over-estimate.
Let $I(1) = \{i \mid \lambda_i(1) < \mu_i\}$ denote the set of stable nodes.

Step 2: Assume nodes $i \notin I(1)$ are unstable and solve for the arrival rate: $\lambda(2) \leq \lambda(1)$.

Repeat until: $I(n) = I(n+1)$.

An efficient algorithm with overflow

Algorithm

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$

Step 1: Assume all nodes have infinite capacity, i.e. $K_i = \infty$ and no overflow, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an under-estimate.
Let $J(1) = \{j \mid \lambda_j(1) > \mu_j\}$ denote the overflowing nodes.

Step 2: Assume nodes $j \notin J(1)$ have infinite capacity and solve for the arrival rate: $\lambda(2) \geq \lambda(1)$.

Repeat until: $J(n) = J(n+1)$.

Number of iterations

Worst case: $O(N^2)$.
Practice (max 800 nodes): $O(\log(N))$

Algorithm

Worst case: $O(N^5)$.
Practice: $O(N^3 \log(N))$

Technische Universiteit Eindhoven University of Technology

An efficient algorithm with overflow

Algorithm

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$

Step 1: Assume all nodes have infinite capacity, i.e. $K_i = \infty$ and no overflow, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an under-estimate.
Let $J(1) = \{j \mid \lambda_j(1) > \mu_j\}$ denote the overflowing nodes.

Step 2: Assume nodes $j \notin J(1)$ have infinite capacity and solve for the arrival rate: $\lambda(2) \geq \lambda(1)$.

Repeat until: $J(n) = J(n+1)$.

Number of iterations

Worst case: $O(N^2)$.
Practice (max 800 nodes): $O(\log(N))$

Algorithm

Worst case: $O(N^5)$.
Practice: $O(N^3 \log(N))$

An efficient algorithm with overflow

Algorithm

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$

Step 1: Assume all nodes have infinite capacity, i.e. $K_i = \infty$ and no overflow, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an under-estimate.
Let $J(1) = \{j \mid \lambda_j(1) > \mu_j\}$ denote the overflowing nodes.

Step 2: Assume nodes $j \notin J(1)$ have infinite capacity and solve for the arrival rate: $\lambda(2) \geq \lambda(1)$.

Repeat until: $J(n) = J(n+1)$.

Number of iterations

Worst case: $O(N^2)$.
Practice (max 800 nodes): $O(\log(N))$

Algorithm

Worst case: $O(N^5)$.
Practice: $O(N^3 \log(N))$

An efficient algorithm with overflow

Algorithm

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$

Step 1: Assume all nodes have infinite capacity, i.e. $K_i = \infty$ and no overflow, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an under-estimate.
Let $J(1) = \{j \mid \lambda_j(1) > \mu_j\}$ denote the overflowing nodes.

Step 2: Assume nodes $j \notin J(1)$ have infinite capacity and solve for the arrival rate: $\lambda(2) \geq \lambda(1)$.

Repeat until: $J(n) = J(n+1)$.

Number of iterations

Worst case: $O(N^2)$.
Practice (max 800 nodes): $O(\log(N))$

Algorithm

Worst case: $O(N^5)$.
Practice: $O(N^3 \log(N))$

Technische Universiteit Eindhoven University of Technology

An efficient algorithm with overflow

Algorithm

Problem: Solve $\lambda = \alpha + P' \min(\lambda, \mu) + Q' \max(0, \lambda - \mu)$

Step 1: Assume all nodes have infinite capacity, i.e. $K_i = \infty$ and no overflow, and solve for arrival rate: $\lambda(1)$.

Observation: $\lambda(1)$ is at worst an under-estimate.
Let $J(1) = \{j \mid \lambda_j(1) > \mu_j\}$ denote the overflowing nodes.

Step 2: Assume nodes $j \notin J(1)$ have infinite capacity and solve for the arrival rate: $\lambda(2) \geq \lambda(1)$.

Repeat until: $J(n) = J(n+1)$.

Number of iterations

Worst case: $O(N^2)$.
Practice (max 800 nodes): $O(\log(N))$

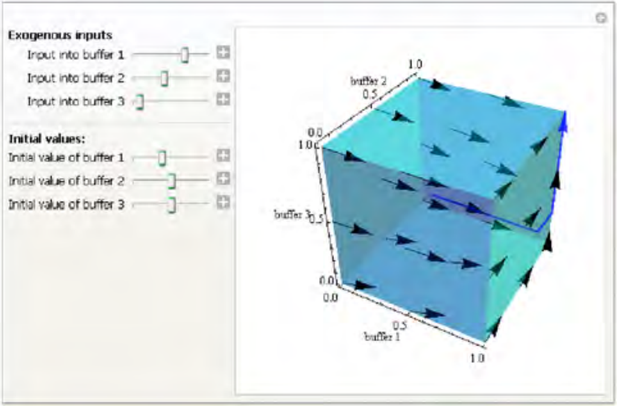
Algorithm

Worst case: $O(N^5)$.
Practice: $O(N^3 \log(N))$

Transient behavior

Algorithm can also be used for determining transient behavior

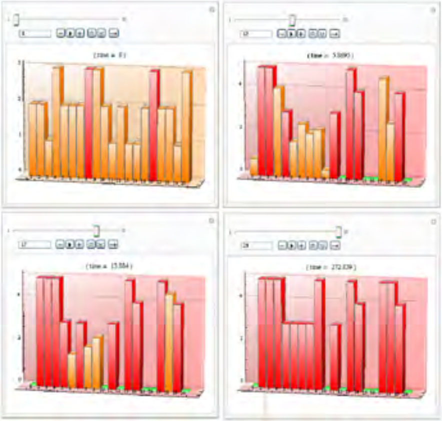
See also http://demonstrations.wolfram.com/DynamicsOfADeterministicOverflowFluidNetwork/



Transient behavior

Algorithm can also be used for determining transient behavior

See also http://demonstrations.wolfram.com/DynamicsOfADeterministicOverflowFluidNetwork/



Sojourn time distributions

Definition

Sojourn time: time in system of customer arriving to steady state FCFS system

Definition

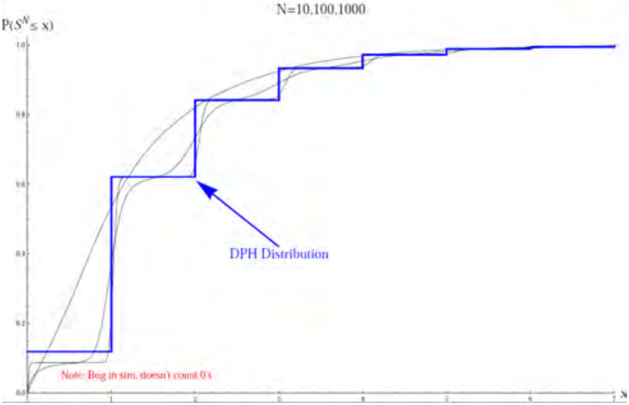
S^N: sojourn time of customer in Nth scaled system

Problem

We want to find the limiting distribution of S^N, i.e. $P(S^N \leq x)$, for $N \to \infty$.

Technische Universiteit Eindhoven University of Technology

Sojourn time distributions



Observation

Sojourn times scale to a discrete distribution

Technische Universiteit Eindhoven University of Technology

$$F = \{1, \ldots, s\} \qquad \lambda_i > \mu_i \text{ for } i \in F$$

$$\bar{F} = \{s+1, \ldots, N\} \qquad \lambda_i < \mu_i \text{ for } i \in \bar{F}$$

Observation

Time through $i \in F \approx NK_i/(N\mu_i) = K_i/\mu_i$, time through $i \notin F \approx 1/(N\mu_i) \approx 0$.

For job at entrance of buffer $i \in F$:

- enters buffer i w.p. $\approx \mu_i/\lambda_i$
- routed to entrance of buffer j w.p. $\approx (1 - \mu_i/\lambda_i)q_{ij}$
- leaves system w.p. $\approx (1 - \mu_i/\lambda_i)\bar{q}_i$

Job at entrance of buffer $i \in \bar{F}$:

- routed according to P almost immediately

Sojourn time distributions

$$F = \{1, \ldots, s\} \qquad \lambda_i > \mu_i \text{ for } i \in F$$

$$\bar{F} = \{s+1, \ldots, N\} \qquad \lambda_i < \mu_i \text{ for } i \in \bar{F}$$

Observation

Time through $i \in F \approx NK_i/(N\mu_i) = K_i/\mu_i$, time through $i \notin F \approx 1/(N\mu_i) \approx 0$.

For job at entrance of buffer $i \in F$:

- enters buffer i w.p. $\approx \mu_i/\lambda_i$
- routed to entrance of buffer j w.p. $\approx (1 - \mu_i/\lambda_i)q_{ij}$
- leaves system w.p. $\approx (1 - \mu_i/\lambda_i)\bar{q}_i$

Job at entrance of buffer $i \in \bar{F}$:

- routed according to P almost immediately

Technische Universiteit Eindhoven University of Technology

$$F = \{1, \dots, s\} \qquad \lambda_i > \mu_i \text{ for } i \in F$$
$$\bar{F} = \{s+1, \dots, N\} \qquad \lambda_i < \mu_i \text{ for } i \in \bar{F}$$

Observation

Time through $i \in F \approx NK_i/(N\mu_i) = K_i/\mu_i$, time through $i \notin F \approx 1/(N\mu_i) \approx 0$.

For job at entrance of buffer $i \in F$:

- enters buffer i w.p. $\approx \mu_i/\lambda_i$
- routed to entrance of buffer j w.p. $\approx (1 - \mu_i/\lambda_i)q_{ij}$
- leaves system w.p. $\approx (1 - \mu_i/\lambda_i)\bar{q}_i$

Job at entrance of buffer $i \in \bar{F}$:

- routed according to P almost immediately

Technische Universiteit Eindhoven University of Technology

Sojourn time distributions

$$F = \{1, \dots, s\} \qquad \lambda_i > \mu_i \text{ for } i \in F$$
$$\bar{F} = \{s+1, \dots, N\} \qquad \lambda_i < \mu_i \text{ for } i \in \bar{F}$$

Observation

Time through $i \in F \approx NK_i/(N\mu_i) = K_i/\mu_i$, time through $i \notin F \approx 1/(N\mu_i) \approx 0$.

For job at entrance of buffer $i \in F$:

- enters buffer i w.p. $\approx \mu_i/\lambda_i$
- routed to entrance of buffer j w.p. $\approx (1 - \mu_i/\lambda_i)q_{ij}$
- leaves system w.p. $\approx (1 - \mu_i/\lambda_i)\bar{q}_i$

Job at entrance of buffer $i \in \bar{F}$:

- routed according to P almost immediately

Technische Universiteit Eindhoven University of Technology

Sojourn time distributions

$$F = \{1, \ldots, s\} \qquad \lambda_i > \mu_i \text{ for } i \in F$$
$$\bar{F} = \{s+1, \ldots, N\} \qquad \lambda_i < \mu_i \text{ for } i \in \bar{F}$$

Observation

Time through $i \in F \approx NK_i/(N\mu_i) = K_i/\mu_i$, time through $i \notin F \approx 1/(N\mu_i) \approx 0$.

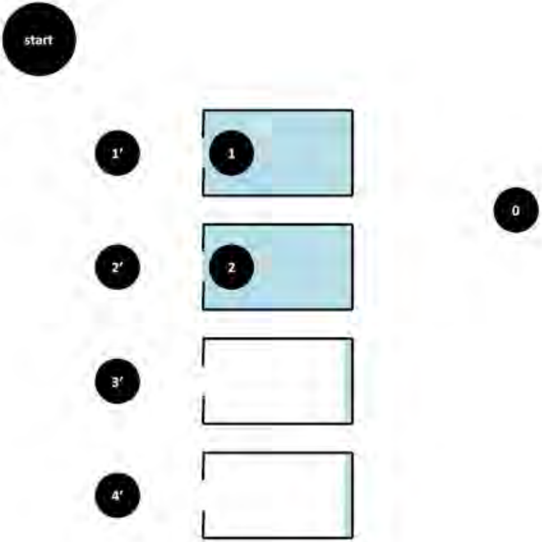
For job at entrance of buffer $i \in F$:

- enters buffer i w.p. $\approx \mu_i/\lambda_i$
- routed to entrance of buffer j w.p. $\approx (1 - \mu_i/\lambda_i)q_{ij}$
- leaves system w.p. $\approx (1 - \mu_i/\lambda_i)\bar{q}_i$

Job at entrance of buffer $i \in \bar{F}$:

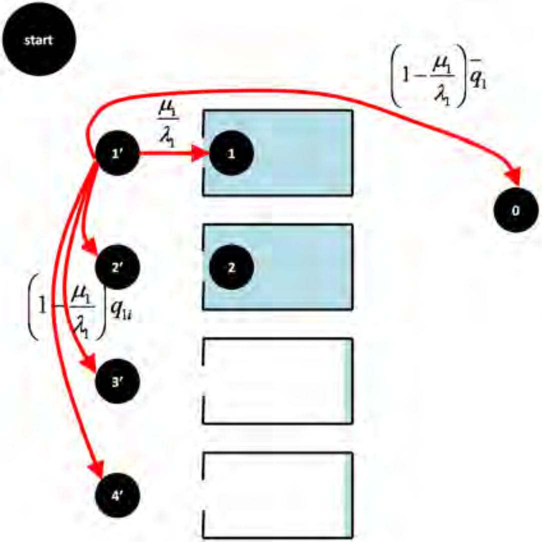
- routed according to P almost immediately

Sojourn time distributions



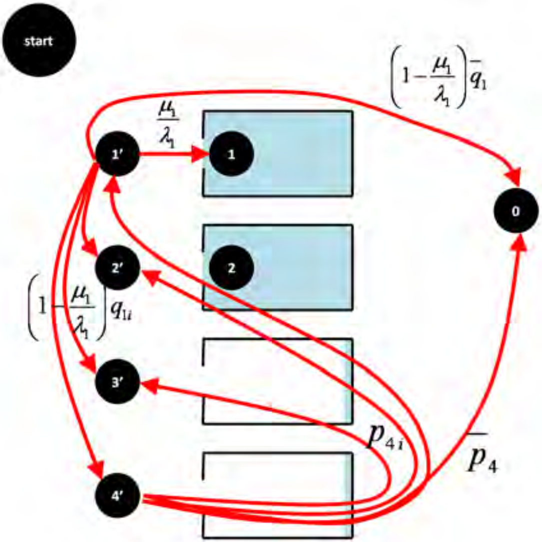
Fast chain on
$\{0, 1, 2, 1', 2', 3', 4'\}$

Sojourn time distributions



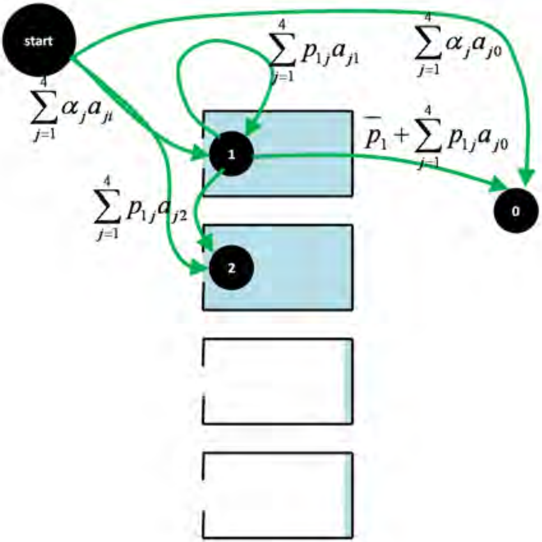
Fast chain on
$\{0, 1, 2, 1', 2', 3', 4'\}$

Sojourn time distributions



a_{ij} absorbtion probability in $j \in \{0, 1, 2\}$ starting in i'

Sojourn time distributions



Slow chain on $\{0, 1, 2\}$, transitions based on fast chain.

Conclusions

Finite buffer networks with overflows.

Contributions

- Limiting traffic equations
- Efficient algorithm for unique solution
- Limiting deterministic trajectories
- Limiting sojourn time distribution

Future work

Work on limits (Chen and Mandelbaum, 1991)