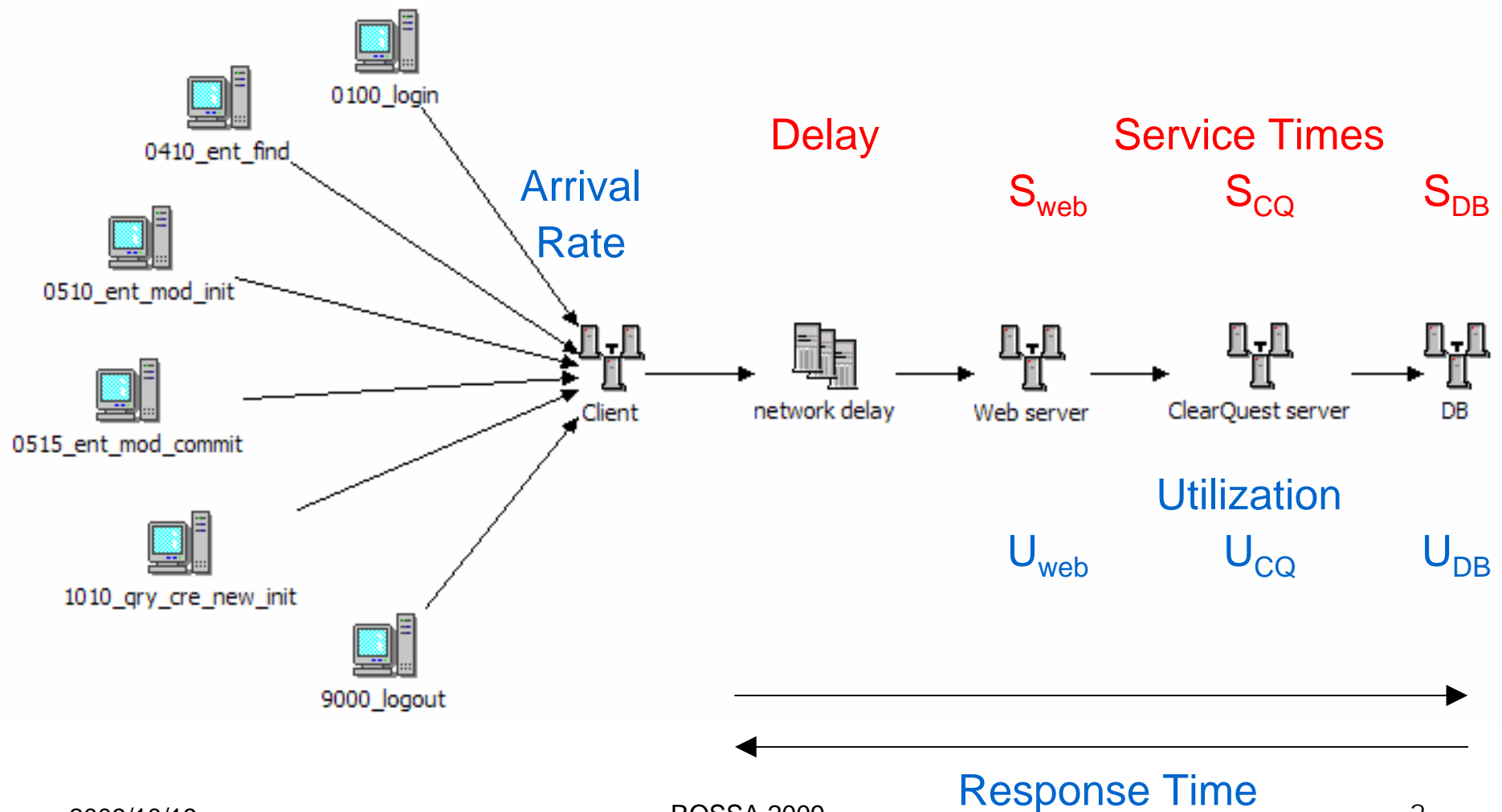




Real-Time Performance Modeling for Adaptive Software Systems

Dinesh Kumar, Asser Tantawi & Li Zhang
IBM T.J. Watson Research Center

Classical queueing theory computes response time, utilization and queue lengths when service time (rate) and delays are given.

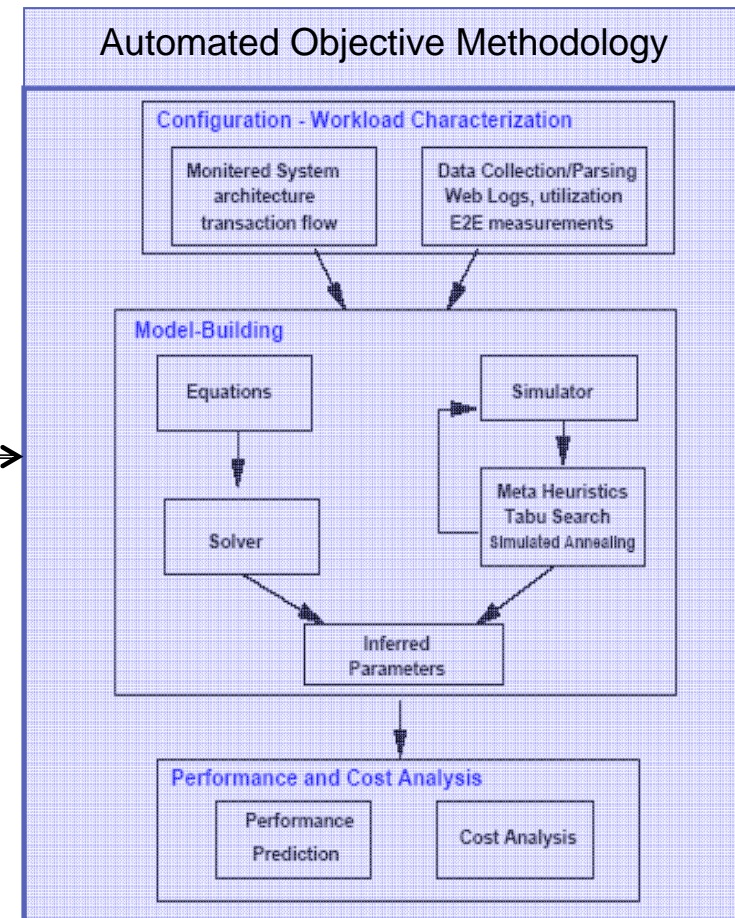
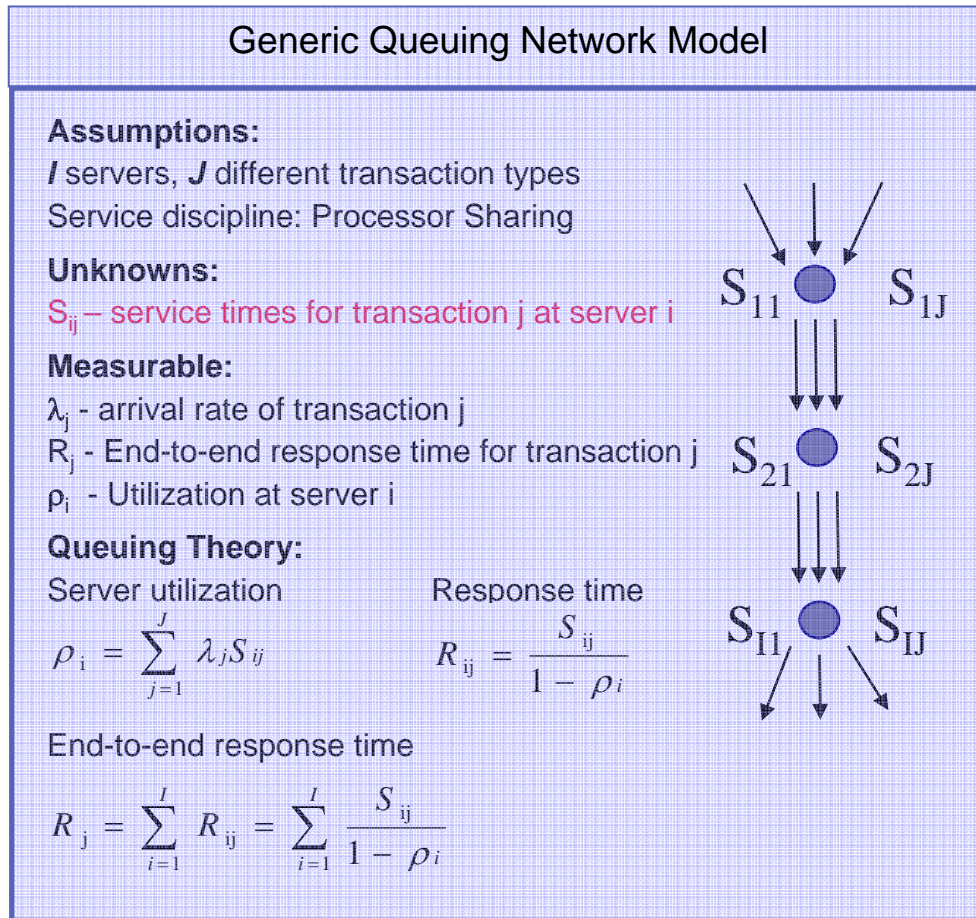




Inverse Problem for Software System Models

- AMBIENCE (*Automatic Model Building using InferENCE*), a performance modeling tool built in IBM Research,
- Employs a powerful *Inferencing* technique to
- Characterize per class service times and delays for a given hardware
- Using end-to-end response time and per box utilization measurements.

AMBIENCE: Inferencing Technique





Real-Time Performance Modeling

- How to **dynamically** update the performance model with real-time measurements ?
- How to modify/extend Inferencing for service time estimation based not on minimization of square of estimation error in **means**, but on minimization of **variance** of estimation error ?
- How to perform dynamic & real-time resource allocation and management ?
- How to account for unpredictable **noise** in measurement data ?



Answer: Use Filtering

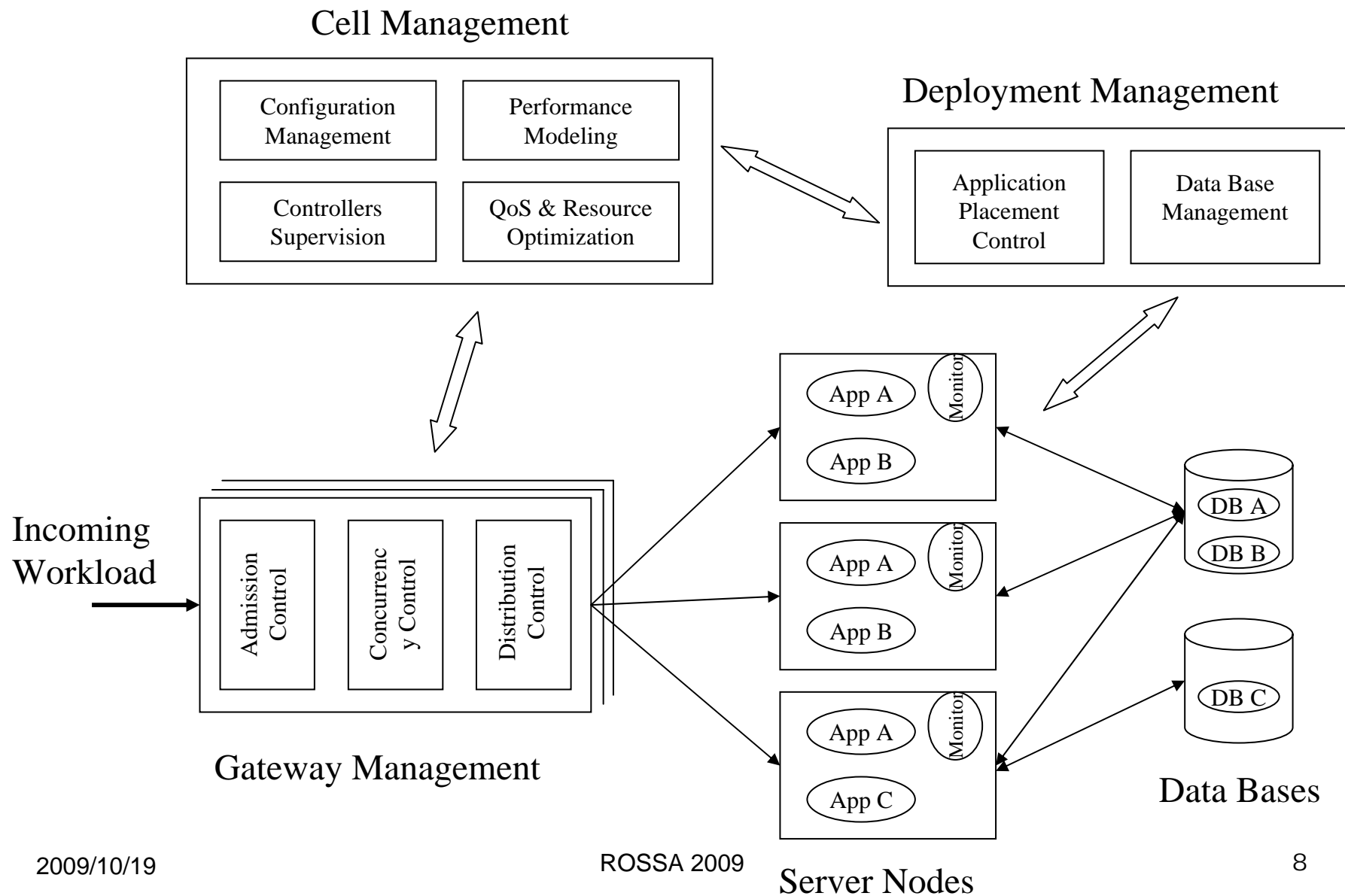
- Optimal estimation & tracking of state parameters of a dynamically evolving system based on a state evolution model, an observation model and real-time measurement data
- Accounts for history of estimation error and goodness of fit
- Minimizes *mean, variance* and *higher order variations* in estimation error
- Well known filters like ***Kalman*** and ***H[∞]-optimal*** filters are extensively used in other fields like guided missile systems, image processing, robotics, etc.
- Have also been applied to time-out estimation in TCP, wireless channel estimation, sensor networks, etc. Woodside et al. have done a partial study in application to performance modeling



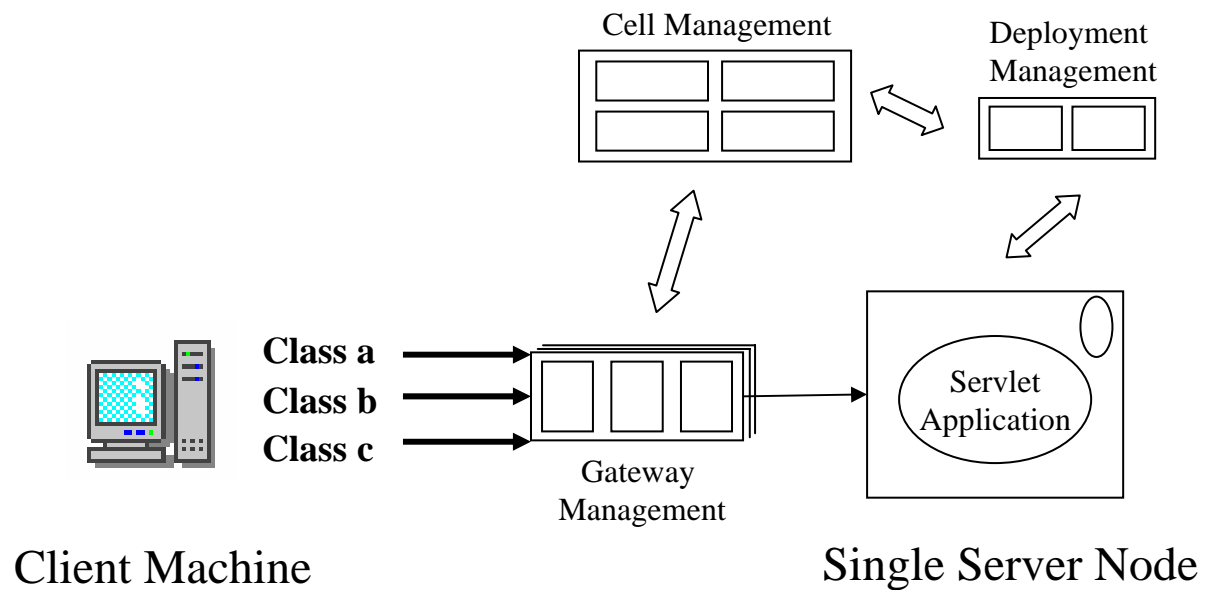
Outline

- **Adaptive Software System**
- **Kalman Filter Design**
- **Problems with Single-Server Multi-class Results**
- **Improved Filter Design and Results**
- **Conclusion**

Adaptive Software System



Single Server, Multi-class Results





Kalman Filter Design

System State: $x = [s^a \quad s^b \quad s^c \quad d^a \quad d^b \quad d^c]^T$

State Evolution: $x_k = F_k x_{k-1} + w_k$

Measurement Model: $z = h(x)$

$$\begin{bmatrix} R^a \\ R^b \\ R^c \\ u \end{bmatrix} = \begin{bmatrix} \frac{s^a}{1-u} + d^a \\ \frac{s^b}{1-u} + d^b \\ \frac{s^c}{1-u} + d^c \\ \frac{1}{P}(\lambda^a s^a + \lambda^b s^b + \lambda^c s^c) \end{bmatrix}$$

Measurement Eq.: $z_k = H_k x_k + v_k$

Process State Noise:

$$w_k \sim \mathcal{N}(0, Q_k)$$

Measurement Noise:

$$v_k \sim \mathcal{N}(0, R_k)$$

Jacobian of Measurement Model:

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{1-u + \frac{\lambda^a s^a}{P}}{(1-u)^2} & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1-u + \frac{\lambda^b s^b}{P}}{(1-u)^2} & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1-u + \frac{\lambda^c s^c}{P}}{(1-u)^2} & 0 & 0 & 1 \\ \frac{\lambda^a}{P} & \frac{\lambda^b}{P} & \frac{\lambda^c}{P} & 0 & 0 & 0 \end{bmatrix}$$

$$H_k = \left[\frac{\partial h}{\partial x} \right]_{\hat{x}_{k|k-1}}$$



Kalman Filter Steps

Predict:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

Update:

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$

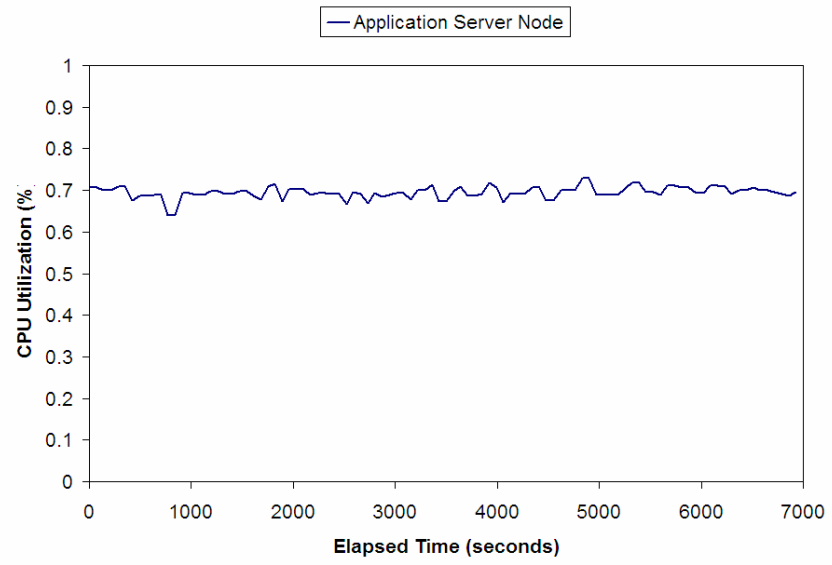
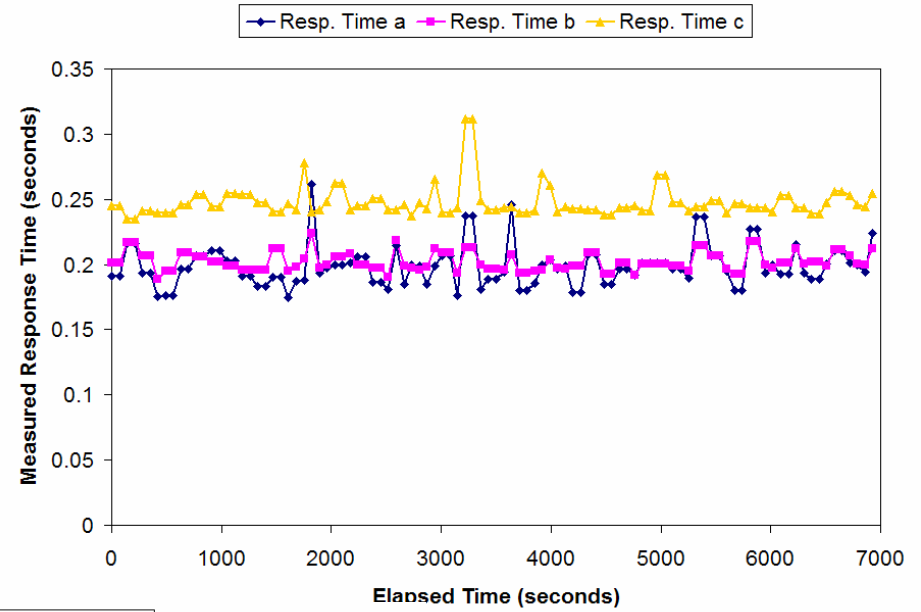
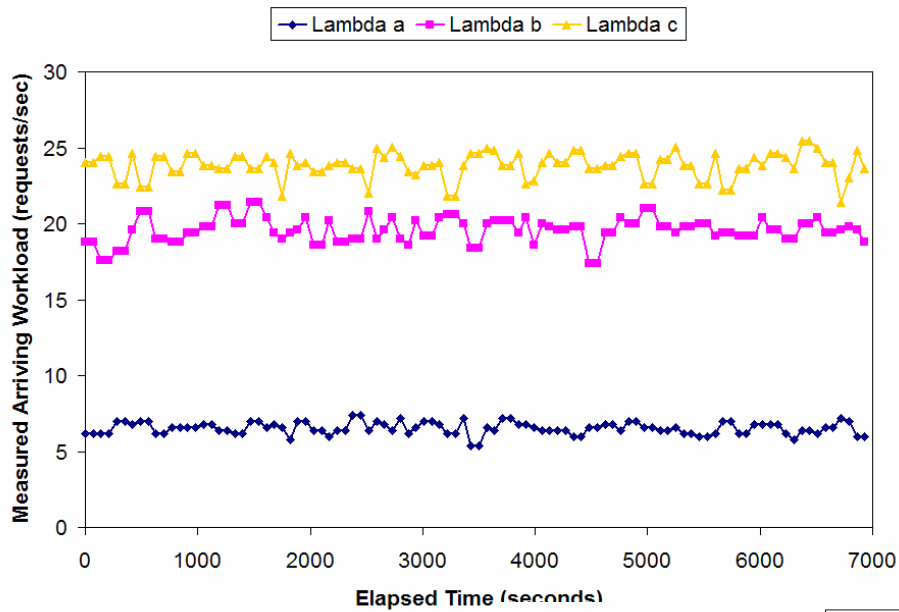
$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

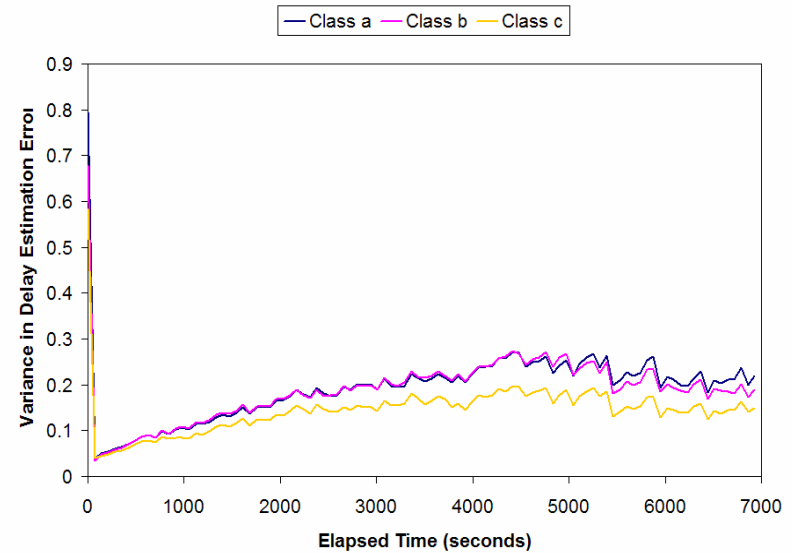
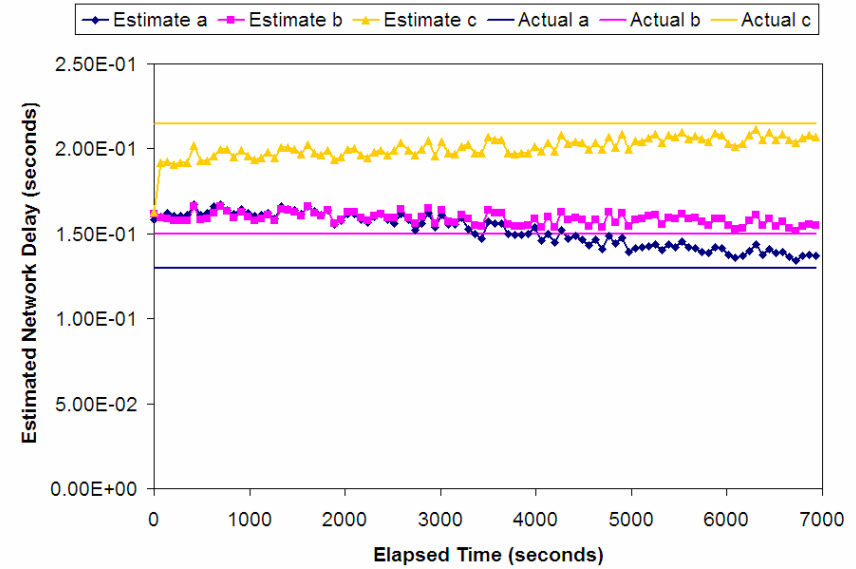
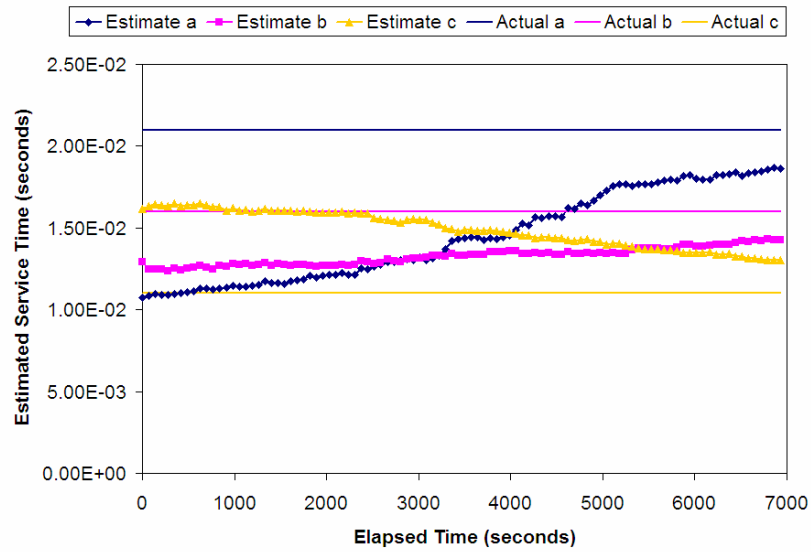
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Low Variation in Workload (LVW) Experiment



LVW Experiment Estimates



Improved Filter with Constraints

$$\begin{bmatrix} z_{\mathbf{k}} \\ d \end{bmatrix} = \begin{bmatrix} H_{\mathbf{k}} \\ D \end{bmatrix} x_{\mathbf{k}} + \begin{bmatrix} v_{\mathbf{k}} \\ 0 \end{bmatrix}$$

where,

$$d = [d_1 \ d_2 \ \cdots \ d_N]^T \quad \text{and} \quad D = [D_1 \ D_2 \ \cdots \ D_N]^T.$$

Here,

$$d_i = \bar{z}_{l_i} = \frac{1}{l_i} \sum_{j=p}^q z_{\mathbf{k}-j} \quad \forall i = 1..N$$

where,

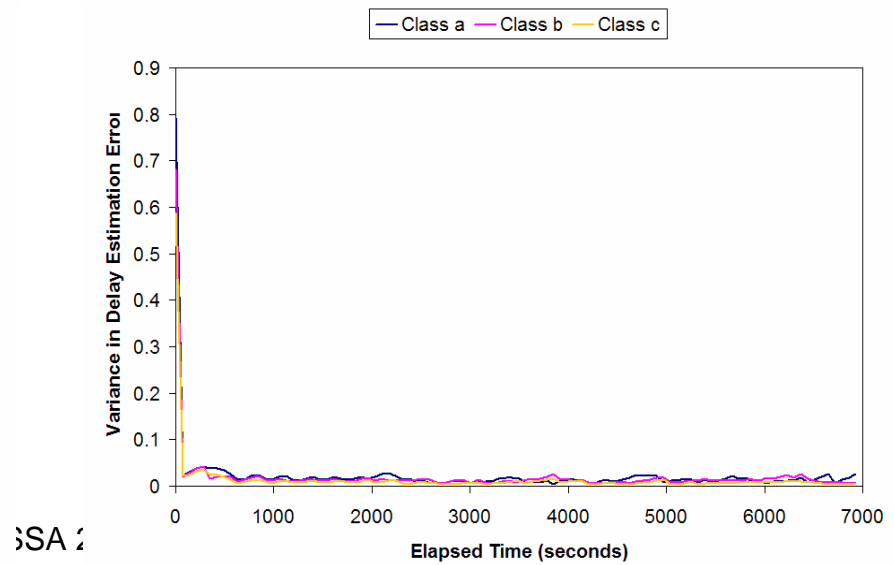
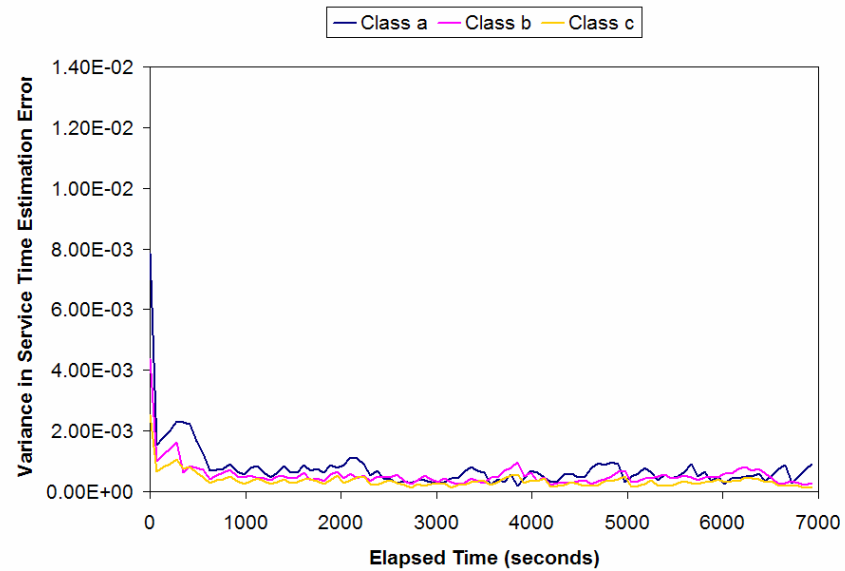
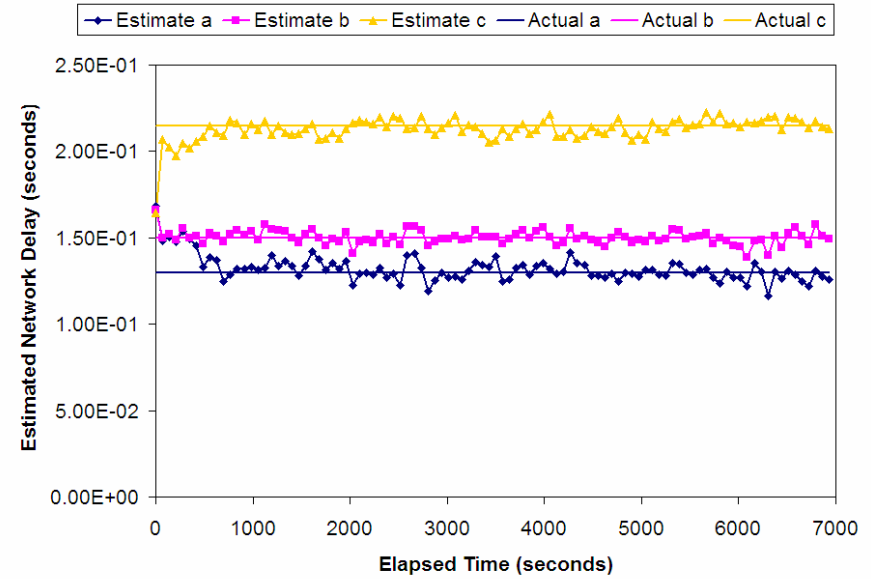
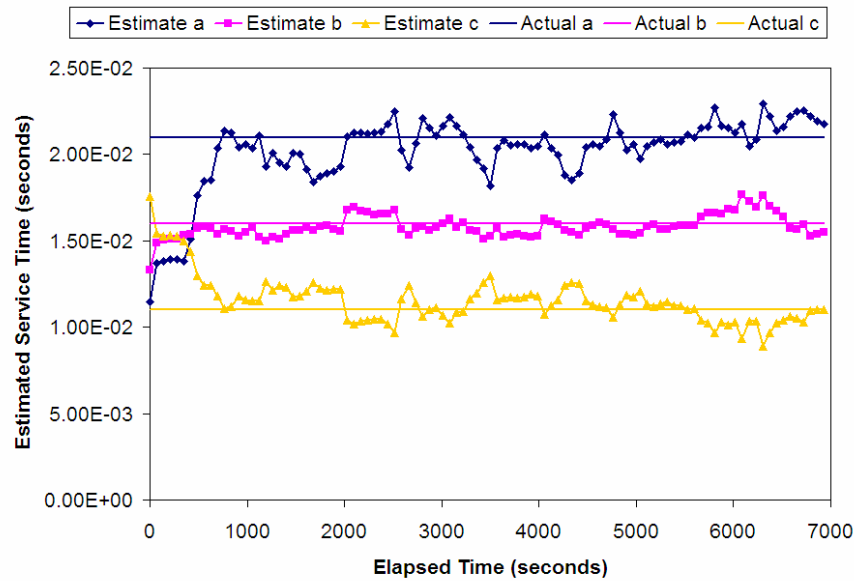
$$p = 1 + \sum_{r=1}^{i-1} l_r \quad \text{and} \quad q = \sum_{r=1}^i l_r.$$

Similarly,

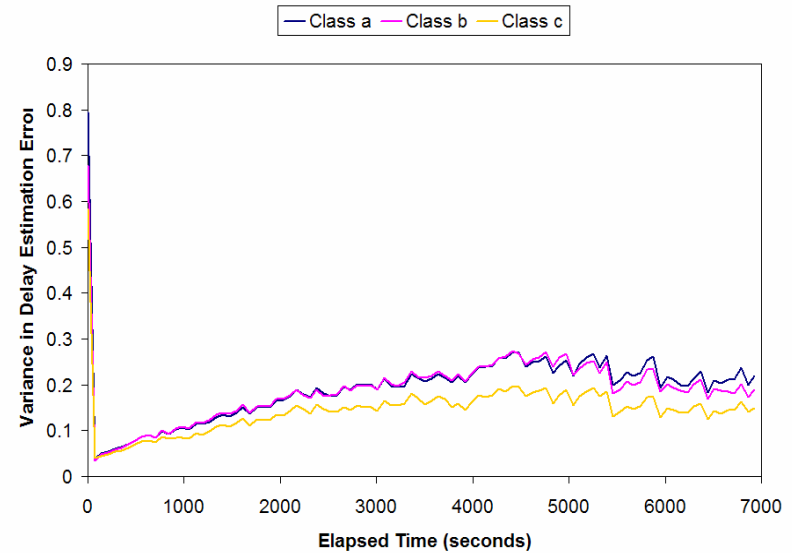
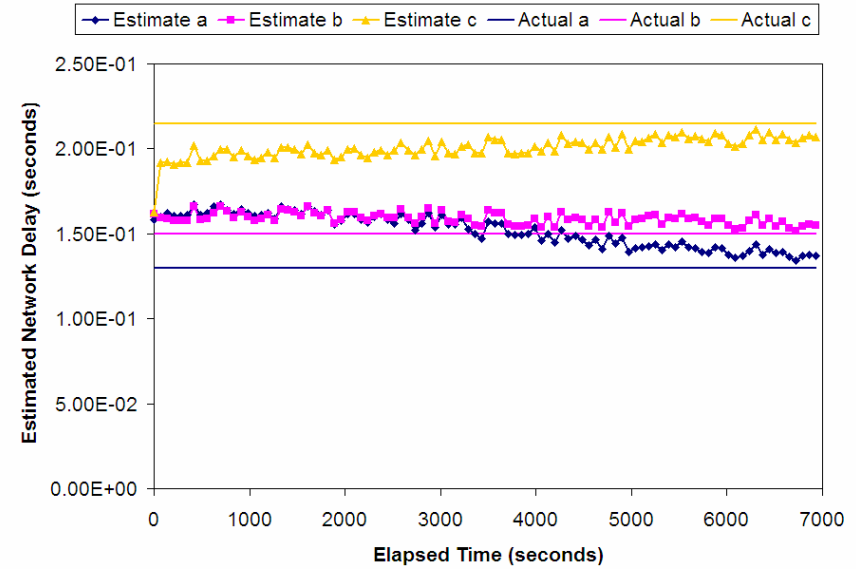
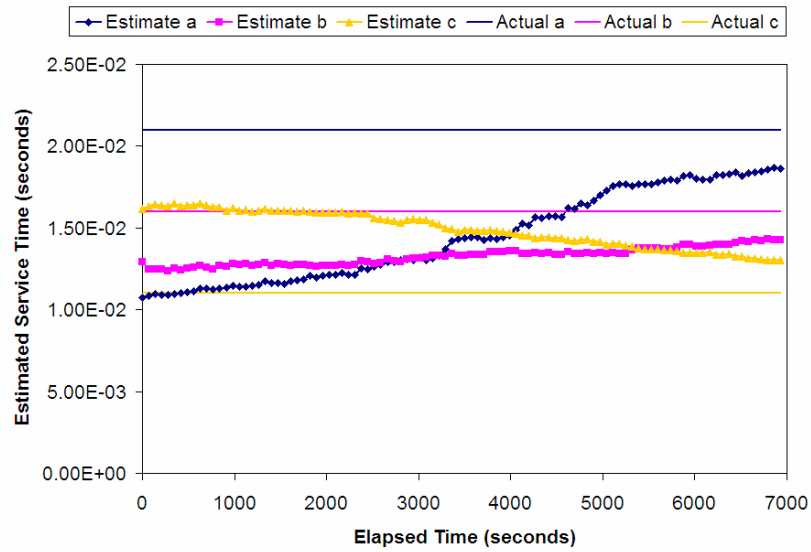
$$D_i = \bar{H}_{l_i} = \frac{1}{l_i} \sum_{j=p}^q H_{\mathbf{k}-j} \quad \forall i = 1..N$$

<i>Experiment</i>	<i>N</i>	<i>l</i> ₁	<i>l</i> ₂
LVW	2	4	3
SSP	1	3	<i>n/a</i>

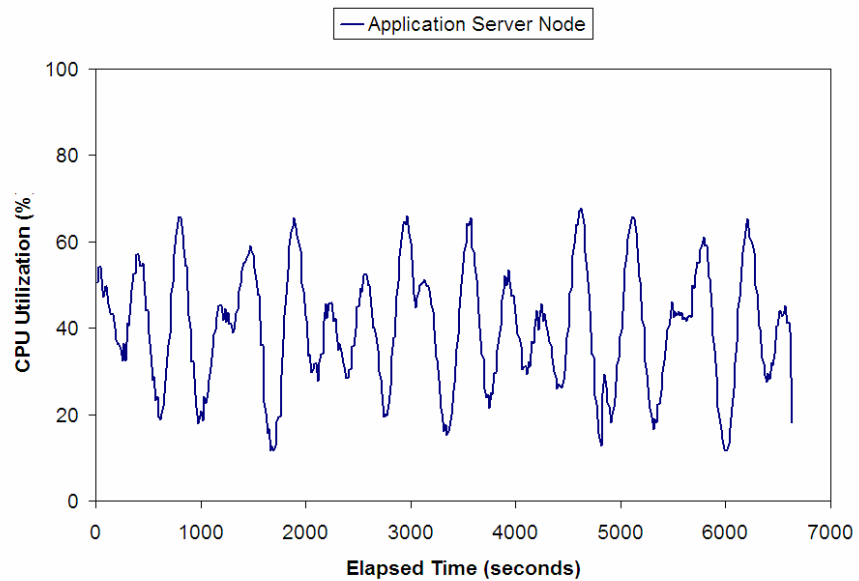
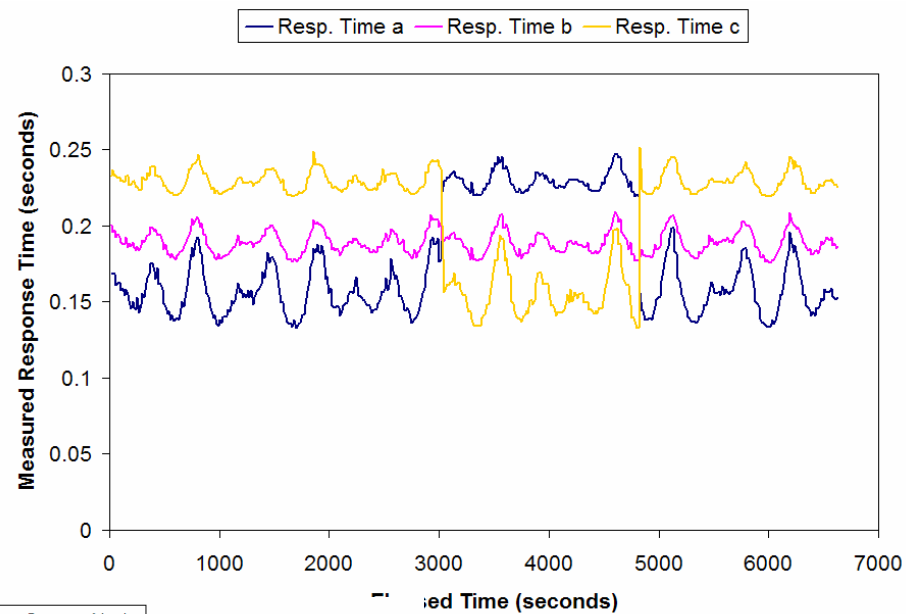
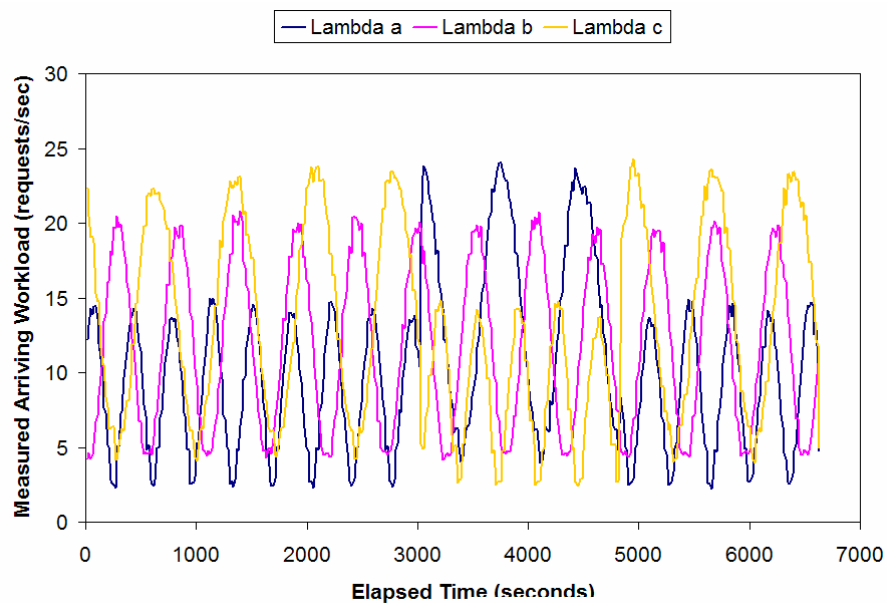
Improved LVW Results



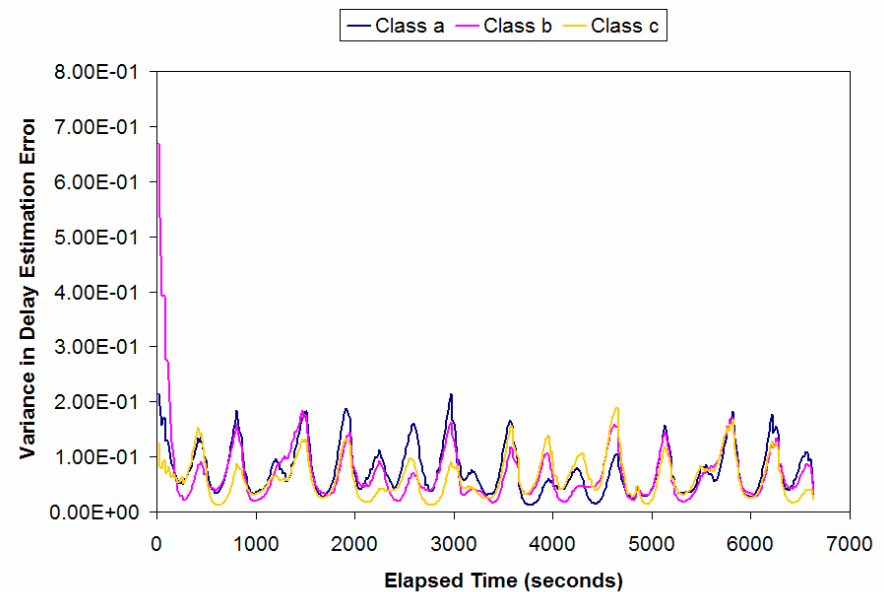
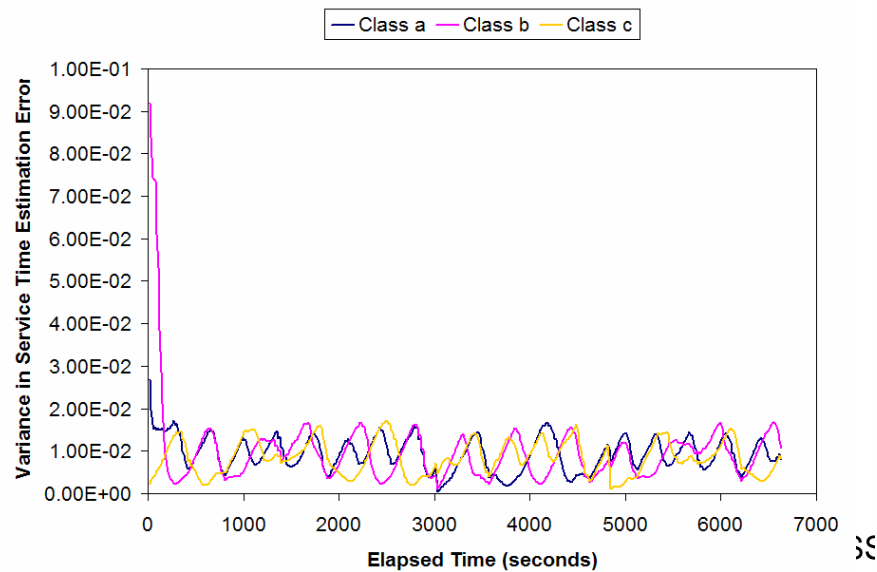
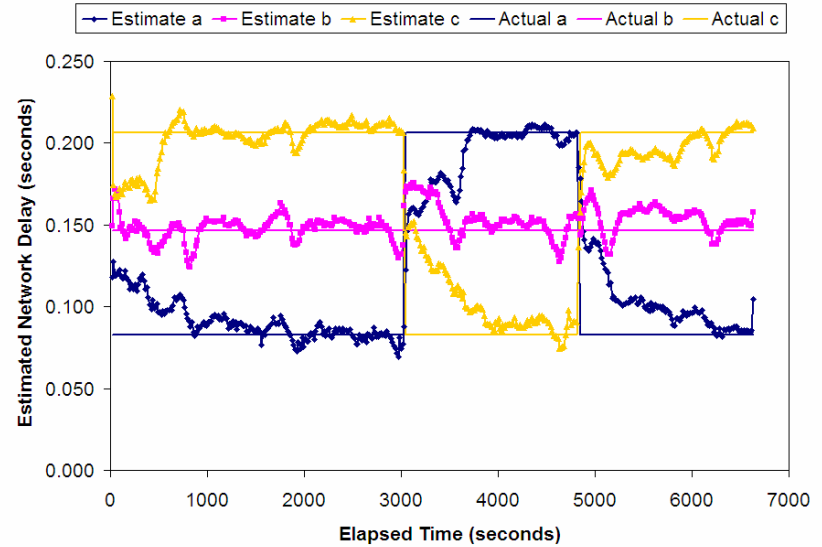
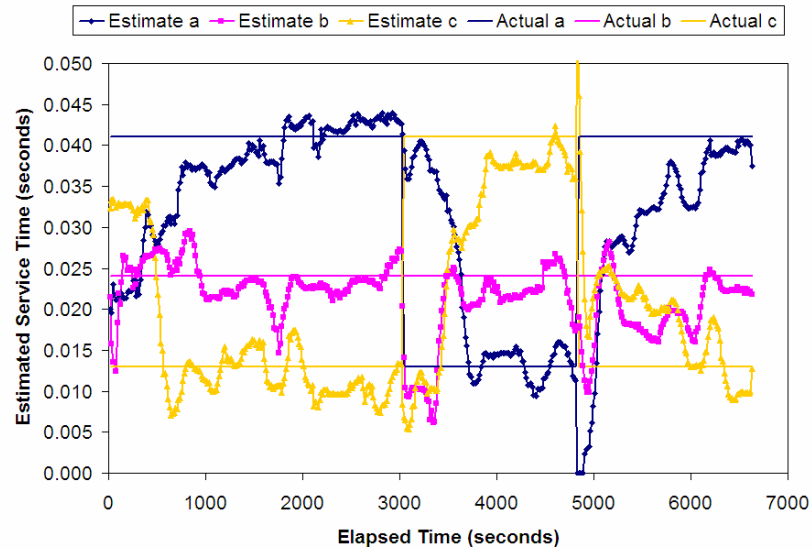
LVW Experiment Estimates



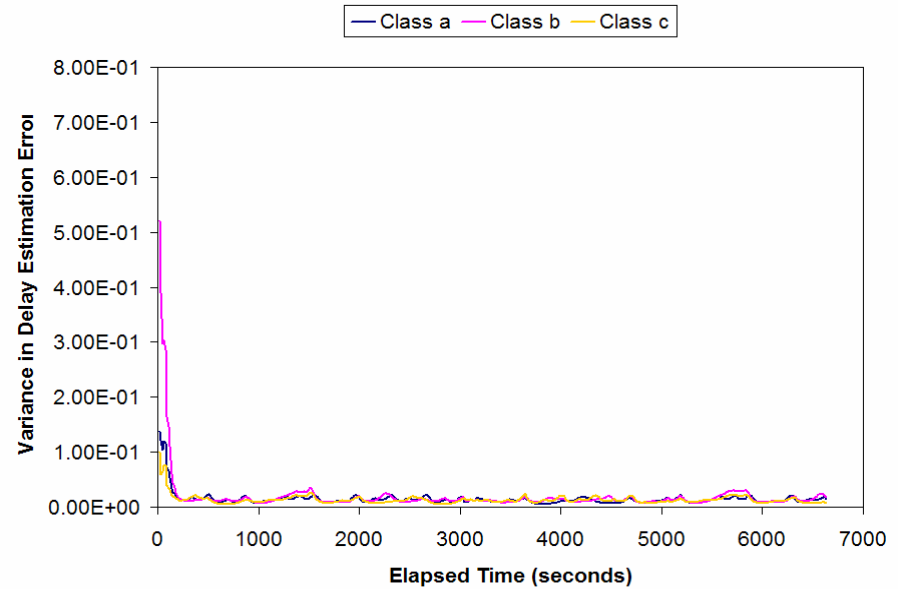
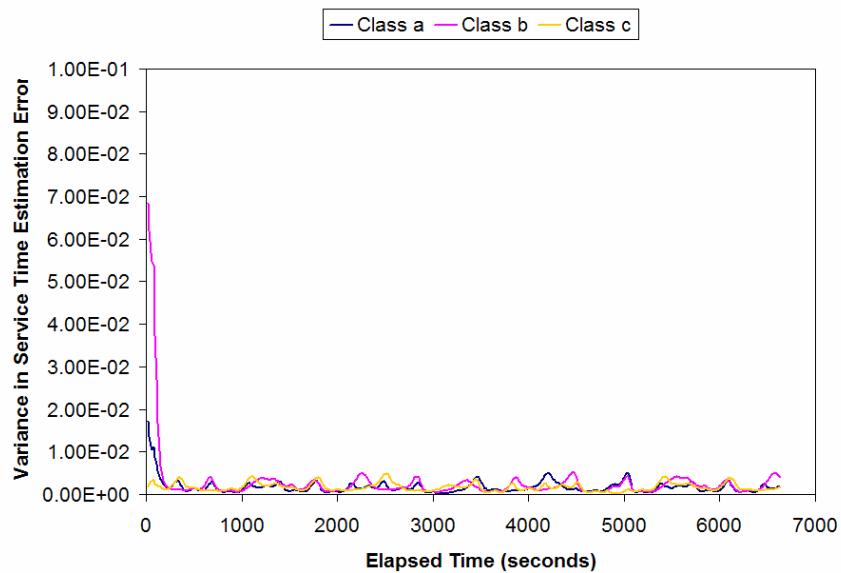
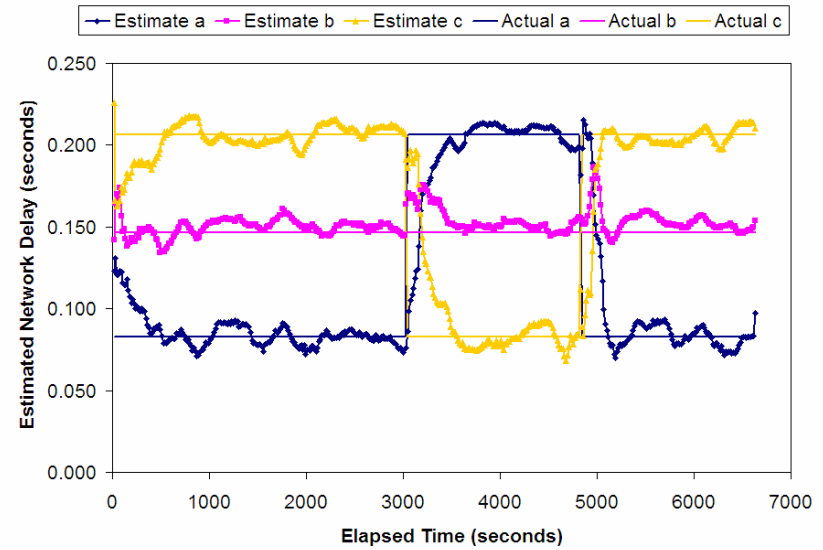
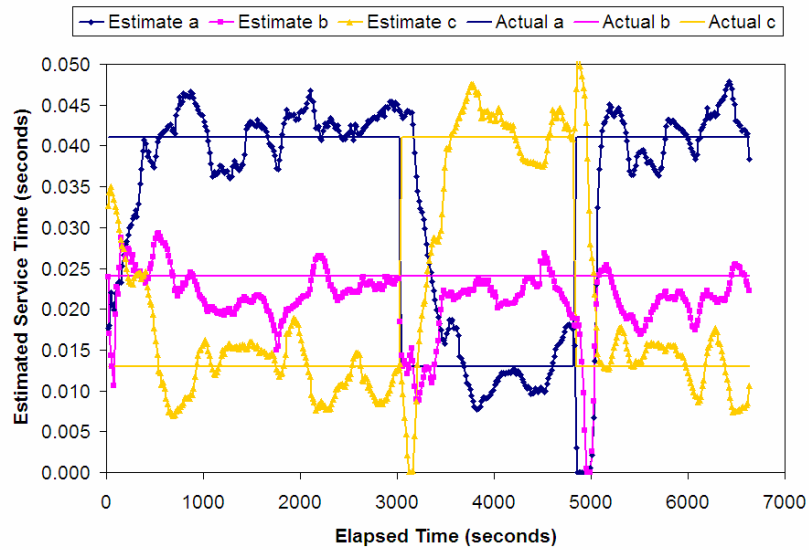
Step-change in System Parameters (SSP) Exp.



SSP Experiment Estimates



Improved SSP Results





Conclusion & Future Work

- **Constrained Kalman Filter gives 4X to 8X improvement for multiple classes of workload**
- **First attempt to propose such a technique**
- **Significant impact on real-time performance modeling of Adaptive Software systems**
- **Can be augmented with control for *real-time* resource management**