Online Supplement for

A Decomposition and Guided Simulation Methodology for Large-Scale System Design: A Study in QoS-Capable Intranets with Fixed and Mobile Components

INFORMS Journal on Computing

Sudip Bhattacharjee
School of Business, University of Connecticut, 2100 Hillside Road, Unit 1041 IM, Storrs, Connecticut 06269-1041, USA, sbhattacharjee@business.uconn.edu

Hong Zhang
College of Business Administration, Southwest Missouri State University, Springfield, Missouri, USA, hoz565f@smsu.edu

R. Ramesh
Jacobs Management Center, SUNY at Buffalo, Buffalo, New York 14260-4000, USA, rramesh@acsu.buffalo.edu

Dee H. Andrews
Air Force Research Laboratory, Mesa Arizona 85206, USA, Dee.Andrews@williams.af.mil

Appendix A: Design Parameters and Model Components

Design Environment: MCSS Architecture (Section 2)

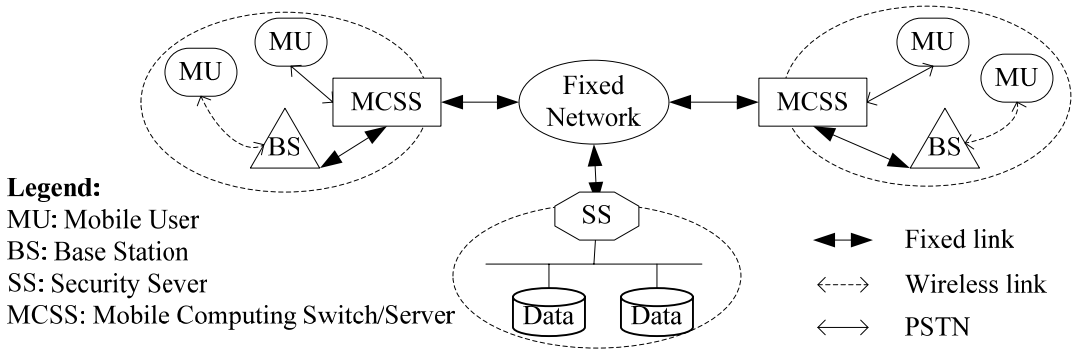

Figure A1: Mobile Computing System Architecture

Figure A1 shows a conceptual view of the physical infrastructure of a mobile computing system and its inter-connection with a fixed network. When a mobile user accesses an organizational network, the requests are sent either to a base station through the wireless network, or directly to a MCSS through the public switch telephone network (PSTN). The base station transfers the requests through a MCSS to fixed network links. The MCSS authenticates user requests and forwards those to specified business components.

Design Variables and Parameters (Section 2.2)

Table A1: DSDP Design Parameters

User Architecture (U)	Network Infrastructure (N)
S: set of data sources/databases; $s \in S$ *B*: set of business components; $b \in B$ G: user groups in enterprise; $g \in G$ Q_{gb} = number of queries/hour/user of group *g* to *b* U_{gb} = number of updates/hour/user of group *g* to *b* S^q = average size of query request (in kB) S^u = average size of update request (in kB) Business Component Architecture (B) *B*: set of business components; $b \in B$ QR_{bs}: number of queries/hour from *b* to *s* UR_{bs}: number of updates/hour from *b* to *s* F_{BC}: component operation and maintenance cost *K*: remote communications cost/kB Data Architecture (D) *S*: set of data sources/databases; $s \in S$ $QL_{bs} = QR_{bs} \times S^q$: query load from *b* to *s* $UL_{bs} = UR_{bs} \times S^u$: update load from *b* to *s* P_{bs} = 1 if *b* accesses *s*, 0 otherwise Q_d = 1 if *d* contains at least one data set, 0 o/w F_{DB}: data set operation and maintenance cost *K*: remote communications cost/kB	**Mobile Network Design Parameters:** *M*: geographical locations/sites of mobile users; $m \in M$ *N:* total number of mobile users; $n \in N$ N_m: number of mobile users in site *m* *Pr(l,m)*: probability of a user moving from site *l* to *m*; $l,m \in M$, $l \neq m$. *Pr(m)* : steady state probability of a mobile user being in site *m* R_m: processing capacity of MCSS in site *m* C_{lm}: fixed cost of connecting site *l* to *m*. $F_m = f * R_m$: overall cost to install and maintain a MCSS in site *m*, expressed as a linear function of R_m. S^c: average message size from client user (Kb) S^s: average message size sent back by server (Kb) *H*: average number of messages a mobile user sends per hour *Pr(s)*: probability of sending a request to database server *s*; $s \in S$

Table A2: DSDP Decision Variables

User Architecture (U)	Network Infrastructure (N)
D: set of domains; $d \in D$ u_{gd}: # users of group *g* in domain *d* Business Component Architecture (B) x_{bd} = 1 if copy of *b* is allocated to *d*, 0 otherwise Data Architecture (D) y_{sd} = 1 if copy of *s* is allocated to *d*, 0 otherwise	Fixed Network Decision Variables: v_1: server processing speeds v_2: input/output rates for multimedia systems v_3: network bandwidth v_4: network protocol v_5: transaction processing model v_6: distributed transaction processing coordination model Mobile Network Decision Variables: p_m = 1 if one MCSS is assigned to *m*, 0 otherwise q_{lm} = 1 if *l* is connected to *m*, 0 otherwise (q_{lm} defined as 0 for $l = m$)

Domain Formation: Sub-Problem DF (Section 3.1)

First consider ART. Query rate over B in domain d is given by $\sum_{b\in B}\sum_{g\in G} Q_{gb}u_{gd}$. Similarly, update rate is $\sum_{b\in B}\sum_{g\in G} U_{gb}u_{gd}$. Total Query Load ($TQL_{sd}$) on s in d is $TQL_{sd} = \sum_{b\in B}\sum_{g\in G} Q_{gb}u_{gd}P_{bs}S^{q}$ and Total Update Load (TUL_{sd}) = $\sum_{b\in B}\sum_{g\in G} U_{gb}u_{gd}P_{bs}S^{u}$. Hence average total load on a database is given by (TQL_{sd} + TUL_{sd}). We assume that a database copy resides on a dedicated server, whose processing capacity (P_{CAP}) is defined as average kB processed per millisecond. Hence average response time from each s in d (assuming negligible transmission delay) is given by

$$ART_{sd} = (TQL_{sd} + TUL_{sd})/(3600000*P_{CAP}),$$

expressed in ms. Hence, average response time in d is $ART_d = \sum_{s} ART_s \Big/ |S|$, where s $\in$ all S present in d and ART = $\sum_{d\in D} ART_d \Big/ |D|$.

Next we define APU. Average utilization for dataset s in d is APU_{sd} = ((TQL_{sd} + TUL_{sd})*3600)/P_{CAP}. Average utilization for component b in d is APU_{bd} = ((TQL_{bd} + TUL_{bd})*3600)/P^{b}_{CAP}, where P^{b}_{CAP} is the processing capacity of a component host (server). Hence average utilization in d is

$$(APU_d) = \frac{\sum_{s} APU_{sd} \Big/ |S| + \sum_{b} APU_{bd} \Big/ |B|}{2},\ s \in S \text{ present in } d,\ b \in B \text{ present in } d.$$

Hence APU is given by $\sum_{d\in D} APU_d \Big/ |D|$.

Finally, $CRAM_d = \sum_{s\in S} F_{DB}\times y_{sd} + \sum_{b\in B} F_{BC}\times x_{bd}$. Assuming each copy of each data set and component is present in each domain,

$$CRAM_{max} = |D|\times\left[|S|\times F_{DB} + |B|\times F_{BC}\right], \text{ and CRAM} = \sum_{d\in D} CRAM_d \Big/ CRAM_{\max}.$$

Resource Allocation: Sub-Problem RA (Section 3.2)

- DLB measures overall domain load balance, and is defined as the ratio of total processing load to

number of domains that contain at least one data set, and is $\frac{\sum_{s\in S}\sum_{b\in B}(QL_{bs}+UL_{bs})}{\sum_{d\in D}Q_d}$.

- MDC identifies the domain that has the maximum component load, which affects system utilization and latency, and is given by $\frac{\underset{d\in D}{Max}\left[\sum_{b\in B}\left(\sum_{d\in D}\sum_{g\in G}u_{gd}\times U_{gb}\right)\times x_{bd}\right]}{\sum_{b\in B}\left(\sum_{d\in D}\sum_{g\in G}u_{gd}\times U_{gb}\right)}$.
- CLB aims to balance query and update load among all data servers, which determines latency, utilization and resource distribution, and is expressed as $\text{Min}\left\{\left|\sum_{b\in B}\sum_{s\in S}(QL_{bs}+UL_{bs})(y_{sd1})-\sum_{b\in B}\sum_{s\in S}(QL_{bs}+UL_{bs})(y_{sd2})\right|\right\}$, $\forall\ d_1\neq d_2;\ d_1,d_2\in D$.
- TC measures operation and maintenance cost (OMC) and remote communication cost (RCC). OMC equals $\sum_{d\in D}\left\{\sum_{b\in B}(x_{bd}\times F_{BC})+\sum_{s\in S}(y_{sd}\times F_{DB})\right\}$, and RCC equals $\sum_{d\in D}\left[\left\{\sum_{b\in B}\sum_{s\in S}(QL_{bs}+UL_{bs})(1-x_{bd}\times y_{sd})\right\}\times K\right]$. Hence, TC = OMC + RCC.
- NL indicates fixed costs to set up network access links of appropriate bandwidth connecting locations, and is given by $\sum_{d\in D}\left[\sum_{b\in B}\sum_{s\in S}(QL_{bs}+UL_{bs})(1-x_{bd}\times y_{sd})\right]$.

Investigating Properties of Objective Function Z_{RA} of Sub-Problem RA:

Z_{RA} = (DLB×W_1 + MDC×W_2 + CLB×W_3 + TC×W_4 + NL ×W_5)

DLB can be rewritten as $c\Big/\sum_{d\in D}Q_d$, where c is the constant term $\sum_{s\in S}\sum_{b\in B}(QL_{bs}+UL_{bs})$. Hence DLB is convex. MDC is convex too: $\sum_{b\in B}\left(\sum_{d\in D}\sum_{g\in G}u_{gd}\times U_{gb}\right)=a$ (say) is constant for a given system.

Hence $a \times x_{bd}$ is linear, and convex. Hence $\underset{d \in D}{Max}[a \times x_{bd}]$ is convex. CLB, on the other hand, is the difference between two linear terms, as can be seen by inspection, which is not necessarily convex. For TC, OMC is convex, as it is a sum of two linear functions. However RCC is a nonlinear function of *x* and *y*, since it includes a product of two decision variables and is non-separable. Even assuming the variables are continuous, applying second order conditions determines that RCC is not convex. Hence TC is not convex. For similar reasons, NL is not convex. Hence the overall objective function (3.2) is non-convex in general.

Domain Formation: Sub-Problem MDF (Section 3.3)

MCSS installation and maintenance cost equals $\sum_{m \in M}(p_m \times F_m)$. Assuming symmetric connection cost (without loss of generality), wireless and PSTN connection costs are $\frac{1}{2}\sum_{l \in M}\sum_{m \in M} q_{lm} \times C_{lm}$. The average number of mobile users connected to the MCSS in *m* is $\sum_{l \in M}(q_{lm} \times N_m)$. Hence, average message load from mobile users on the MCSS in *m* is $\sum_{l \in M}(q_{lm} \times N_m \times H \times S^c)$. The average number of messages received by database server *s* is $g_s = N \times H \times Pr(s)$. If each mobile user request triggers one response from *s*, average number of messages from *s* to MCSS in *m* is $g_{sm} = g_s \times \sum_{l \in M}(q_{lm} \times Pr(l))$. Therefore, average message loads from server *s* to MCSS in *m* is $g_{sm} \times S^s$. Hence average message load from all servers to MCSS in *m* is $\sum_{s \in S} g_{sm} \times S^s$. Hence total message load for MCSS in *m is*

$$TML_m = \sum_{l \in M}(q_{lm} \times N_m \times H \times S^c) + \sum_{s \in S} g_{sm} \times S^s$$

Configuration Selection: Sub-Problem MND (Section 4.2)

A low security protocol requires a MU to be authenticated and authorized only once during each session, while a high security protocol requires authentication and authorization for each request. Figure A2 illustrates a low security routing process, where authentication occurs only at the MCSS level. Figure A3

shows a medium security level protocol, which requires a MU to be verified by the security server once during a session in addition to authentication at the MCSS. A high security level protocol is shown in Figure A4, where each request sent by the MU is authenticated at the MCSS and the security server.

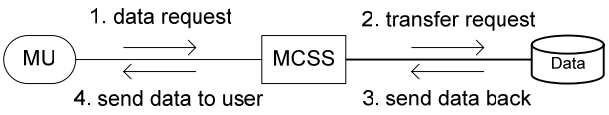

Figure A2: Low security level request routing protocol

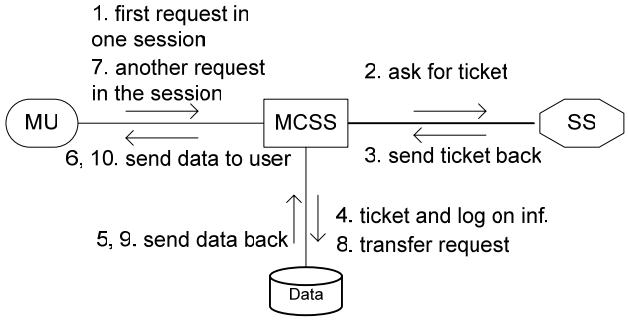

Figure A3: Medium security level routing protocol

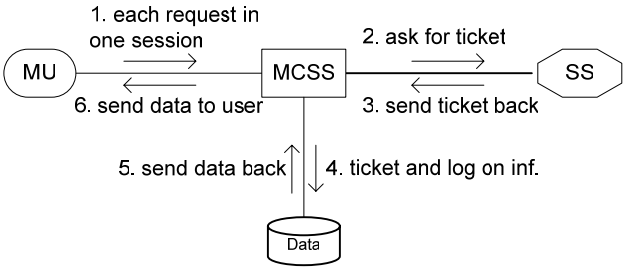

Figure A4: High security level routing protocol

Appendix B: Algorithms

Sub-problem DF: *Variable-Domain Perturbation Algorithm*

Step 0. Start with domain $d = 1$.
Step 1. For each d, perform the following:
Step 2. If $d > D$, STOP. Else, calculate Z_{DF}.
Step 3. Determine a point L in d that is currently farthest from d's centroid. If there is no more point remaining in that domain, set $d \leftarrow d + 1$, and go to Step 2. Else, go to Step 4.
Step 4. Determine the next closest domain centroid j.
Step 5. Allocate L to j determined in Step 4.
Step 6. Recalculate Z_{DF}.
Step 7. If Z_{DF} improves, allocate L to j, and recalculate centroids. Go to Step 3. Else go to Step 8.
Step 8. If Z_{DF} worsens, undo Step 5. Repeat Step 4 to find the next best domain j' available, if any. If any j' is available, redo Step 5 to 7 with j'. If no j' is available, form a new domain with L as the sole member in that domain, and perform Step 6. If estimators improve, go to Step 7, else go to Step 3.

Sub-problem MDF: *MCSS Domain-Creation Algorithm*

Input: M, N, N_m, p_{avg}, k.
Step 1. Sort locations m in descending order of number of mobile users (n_m).
Step 2. Take first k locations to form MCSS domains.
Step 3. Assign each successive non-MCSS location to domain with minimum number of mobile users.
Step 4. For each domain, DO {
For each location within domain, calculate cost if it is designated as MCSS location. Choose location with minimum cost as MCSS location.}
STOP and go to *MCSS location perturbation algorithm*.
Output: Initial φ_{MDF} and Z_{MDF}.

Sub-problem MDF: Tabu Search Meta-Heuristic

Let M denote the set of MCSS locations and N denote the set of non-MCSS locations. Let s_i denote a location, which can either be a MCSS location or a non-MCSS location. For *MCSS location perturbation algorithm*, let $<s_i, s_j>$ denote a move of exchange in which s_i is dropped out of a set it belongs to and s_j is added to the set. The candidate list includes all $<s_i, s_j>$ such that $s_i \in M$ and $s_j \in N$. If an admissible $<s_i, s_j>$ results in a better solution than the incumbent solution, any move trying to drop s_j out of M is a tabu unless at least one of aspiration criteria is satisfied or the tabu length expires. If a move $<s_j, s_k>$ is marked as a tabu but produces a solution better than currently best-known solution, the aspiration criteria is satisfied and its tabu status is disregarded and the move is executed. For *mobile domain reconfiguration algorithm*, let m_i denote a MCSS domain ($m_i \in M$) and n_i denote a non-MCSS location ($n_i \in N$). Let (n_j,

m_i) denote the non-MCSS location n_j located in m_j MCSS domain. For a given m_i, the candidate list includes only those n_j that lie in the remote MCSS domain m_k, such that $i \neq k$. The move is a "single" move, i.e., each move only places a single n_j from m_k to m_i. If moving a non-MCSS location n_j from MCSS domain m_k to m_i results in better solution the incumbent solution, n_j becomes a tabu. The assignment (n_j, m_i) is marked as (n_j, m_i)t, signifying that a tabu move cannot enter solution space during the tabu length. If a tabu move results in a better solution than the best-known solution from *MCSS location perturbation algorithm*, then its tabu status becomes inactive.

Sub-problem MDF: *MCSS Location Perturbation Algorithm*

Input: Output from *MCSS domain creation algorithm*; range of p_{avg}; ***Control:*** *iter* (actual number of iterations); *iter_max* (maximum number of iterations); *tabu1* (tabu list); *len* (length of tabu list); len_{min} (minimum length of tabu list); len_{max} (maximum length of tabu list).

Initialize: iter ← 0; *tabu1* ← { }; *len* ← len_{min}; C_{bc} (cost of best known solution) ← initial Z_{MDF} from *MCSS domain creation algorithm*; C_{inc} (Cost of incumbent solution for current move) ← initial Z_{MDF} from *MCSS domain creation algorithm*; {M*} ← set of MCSS locations.
Do {
Step 1. Form candidate list (CL) from current best domain configuration.
Step 2. Let *(q, k)* denote a move for a non-MCSS location *q* to be reassigned to domain *k*.
Step 3. Call *mobile domain reconfiguration algorithm* to compute Z_{MDF} for new set $C_c(q, k)$. if $C_c(q, k) \geq C_{inc}$, go to step 6.
Step 4. If *q* NOT in *tabu1*, go to step 7.
Step 5. Check aspiration condition. If $C_c(q, k) < C_{bc}$, go to step 7.
Step 6. Set *len* ← *len* + 1. If *len* > len_{max}, set *len* ← len_{min}. Go to step 8.
Step 7. Set C_{inc} ← $C_c(q, k)$; save incumbent configuration; go to step 6.
Step 8. If CL ≠ { }, go to step 2.
Step 9. If $C_{inc} < C_{bc}$, set C_{bc} ← C_{inc}; save configuration. Update *tabu1* to include *q*. *iter* = *iter* + 1.}
While (*iter* < *iter_max*)
Output: φ_{MDF}.

Sub-problem MDF: *Mobile Domain Reconfiguration Algorithm*

Input: Output from *MCSS location perturbation algorithm*; C_{bc} . ***Control:*** *iter2*; *iter_max2*; *tabu2* (tabu list); *len2* (length of tabu list); $len2_{min}$ (minimum length of tabu list); $len2_{max}$ (maximum length of tabu list).

Initialize: iter2 ← 0; *tabu2* ← { }; *len2* ← $len2_{min}$; C_{bc_alloc} (Cost of incumbent solution in this algorithm) ← very large number
Step 1. Sort locations *m* in descending order of number of mobile users (n_m).
Step 2. Assign each non-MCSS location successively to domain with minimum number of mobile users. Compute Z_{MDF}. If $Z_{MDF} < C_{bc\_alloc}$, C_{bc_alloc} ← Z_{MDF}.
Do {

Step 1. Create neighborhood of current best configuration as candidate list (CL) of moves.
Step 2. Let *(s, t)* denote a move for a non-MCSS location *s* to be reassigned to domain *t*. Compute resulting total cost *C(s, t)*. Add penalty cost for infeasible solution, i.e. processing constraints violated. If $C(s, t) \geq C_{bc\_alloc}$, go to Step 8.
Step 3. If *s* NOT in *tabu2*, go to Step 7.
Step 4. If $C(s, t) \geq C_{bc}$, go to Step 8.
Step 5. Reassign *s* to domain *t*; update *tabu2* to include *s*; set $C_{bc\_alloc} \leftarrow C(s, t)$; Go to Step 9.
Step 6. $len2 \leftarrow len2 + 1$; If $len2 > len2_{max}$, then $len2 \leftarrow len2_{min.}$
Step 7. If CL $\neq$ { }, go to Step 4.
Step 8. $iter2 \leftarrow iter2 + 1$. Go to Step 1. }
While ($iter2 < iter\_max2$)
Go to *MCSS location perturbation algorithm* with chosen best admissible move.

Sub-problems ND and MND: *Iterative Simulation Modeling and Analysis Algorithm*

Input: Previous sub-problem configurations (φ_{DF} and φ_{RA} for sub-problem ND, φ_{DF}, φ_{RA}, φ_{MDF} and φ_{ND} for sub-problem MND); *Control*: ITER (total number of iterations).
Set $t = 1$.
Simulation and Learning Model:
1. Identify input parameters for respective sub-problem to be solved. Perform pilot simulations to determine significant parameters for detailed investigation (X_1 through X_p). Parameter identified as significant if a unit change produces at least 10% change in an output measure.
2. Determine factorial simulation experimental design based on number of significant parameters.
3. Determine number of replications needed for a given precision and confidence level (precision of 0.25 for ND and 0.15 for MND here, 90% confidence level) (see Law and Kelton 2000)
4. Initialize and run replications for each design point with appropriate warm up time for system to reach steady state. Each replication is run for 86,400 s of simulation time. Tabulate objective (output) measures z_1 through z_n, where n is the number of measures.
5. Estimate general linear model for each measure z_1 through z_n. Determine significant parameters and related coefficients of main and interaction effects (at 90% confidence level).
6. Choose initial solution configuration: using the sets of output measures from all experimental design points, create a locally non-dominated solution pool and choose one solution from this solution pool.

Decision Support for System Configuration:
7. Use general linear model to perturb parameters and identify promising configurations for detailed simulation analysis.
8. Simulate each configuration with above determined number of replications. Refine general linear model using updated data from simulations.
9. Determine locally non-dominated solution pool for configurations simulated in step 8. Calculate percentage improvement of each output measure compared to initial solution identified in step 6 (or current solution if $t > 1$). Choose configuration that produces the highest positive additive percentage improvement. If no configuration produces a positive additive percentage improvement, **STOP**. If $t \geq$ ITER, **EXIT**.
10. Update $t \leftarrow t + 1$. Go to Step 7.

Output: Design configuration.