Performance Evaluation of Distributed Application Virtualization Services Using the UMTS Mobility Model

Chung-Ping Hung* and Paul S. Min†
Department of Electrical and Systems Engineering, Washington University in St. Louis
One Brookings Drive, St. Louis, MO 63130, USA
Email: *chung23@wustl.edu †psm@wustl.edu

***Abstract*—In this paper, we first introduce how virtualization technologies can mitigate mobile application software publishing problems due to platform diversity and fragmentation. In our previous work, we proposed a distributed server arrangement and the corresponding hand-off protocol to provide better user experience for application virtualization on mobile devices and evaluated the performance using the modified UMTS outdoor to indoor pedestrian mobility model. We continue our previous work on evaluating performance of the proposed service architecture using the UMTS rural vehicular mobility model with similar modifications. In this paper, combined with our previous work, we complete the establishment of quantitative relations between the performance improvement or impact and the infrastructure related parameters in the typical mobility model.**

***Index Terms*—telecommunication and wireless networks, computer networks, information technology, UMTS mobility model.**

I. Introduction and Related Work

Advanced wireless communication and low-power semiconductor technologies have enabled mobile computing widely available to consumers and dramatically expanded our imagination on personal computing. Mobile computing devices, however, are hardly considered as technical extension of general purpose computers. Mobile computing devices are limited in computational resources, communication-oriented, and highly compact. Therefore, mobile computing devices are "advanced interactive embedded real-time systems" rather than "reduced PCs".

The nature of mobile computing devices inevitably makes the software engineering on them tightly managed by platform vendors. Although centrally managed software publishing helps software developers securing their revenue, this paradigm also enables platform vendors to take much more control on SDKs, design guidelines, and whether a 3rd party software product can be shipped or not. Furthermore, various mobile operating systems are still competing with each other and none is expected to dominate the market in 2 or 3 years. Software developers have to deal with multiple dictating bureaucracies, instead of one, to make their products available to most customers and maximize their revenue. Therefore, developing cross-platform software for mobile computing devices is a costly work using conventional frameworks.

Fortunately, virtualization technologies can work around the difficulties of deploying cross-platform mobile application software. Roughly speaking, application virtualization technologies can be categorized into two major paradigms: one is creating a compatible runtime platform, i.e., virtual machine, on each client's device and publishing well managed code packages running on top of it [1][2], and the other is executing application software on a well managed server while each client's device only handles user inputs and server outputs [3][4][5]. We generally refer to the later paradigm as the *browser-based approach* since web browsers provide a very ideal framework for it.

Despite being technically feasible, deploying a virtual machine running on top of a mobile operating system to execute downloaded common codes circumvents the official software publication platform and generally is considered a violation of the "Non-Compete" policy [7][8] by marketplace operators.[1] Therefore, the browser-based approach becomes the remaining legitimate way to provide application virtualization services on mobile computing devices for general 3rd party developers without special privilege, unless marketplace operators enforce the "Non-Compete" policy against interactive web contents.

The conventional web-based application virtualization relies on a centralized server to provide the service through established Internet infrastructure. Although this configuration can be built with low cost, the long response latency could significantly reduce the user experience since every input must travel through a series of routers and bridges to the colocation center and the corresponding update has to traverse backward through the nodes. Each node along the route induces processing delay, queuing delay, and transmission delay, and each link comprises the route induces propagation delay. Generally speaking, network delay is highly related to the geographical distance between two end points given similar network infrastructure technologies.

[1] VMware's Mobile Virtualization Platform (MVP) [6], which implements this paradigm, is not available in Android Marketplace. To install MVP on an Android phone requires sideloading, and only Android platform leaves this loophole to install apps outside the marketplace, which is at the mercy of Google and wireless service providers. In fact, some wireless providers do block sideloading on some Android phones. Furthermore, among the major mobile device players, only Android is supported by MVP. Therefore, even VMware starts their own app store for MVP, it doesn't help cross-platform software deployment anyway.

 ISBN: 978-1-61208-164-9

In our previous work [11], we have proposed an alternative configuration to address this issue that geographically partitions the service area into multiple smaller service areas and deploys a smaller server for each one to provide the service locally. The proposed configurations should significantly reduce propagation delay in most case due to the shorter average communication distance. The proposed configuration, however, has to handle hand-off cases, i.e., mobile stations in use moving from one service area to another. Therefore, we also proposed a hand-off protocol offering seamless user experience.

Handling hand-offs induces longer response latency and thus the overall performance depends on the hand-off behavior model. Therefore, we used an empirical and simplified approach to evaluate the performance as a result of infrastructure arrangement and application software's properties in our previous work [11]. We also used one of the UMTS mobility models to empirically establish the correlations between the performance and the size of each local service area and the capabilities of the network infrastructure in [12]. In this paper, we complete the performance simulation of the proposed architecture with the UMTS rural vehicular mobility model. In the UMTS rural vehicular mobility model, base stations (BSs) are sparsely but optimally placed, mobile stations (MSs) move faster and more freely, and the hand-off behavior among base stations is different as well. Consequently, the simulation program in the UMTS rural vehicular model is significantly different from the one presented in our previous work though sharing the same concept. The simulation algorithm and results based on the UMTS rural vehicular mobility model are represented in this paper.

There are several papers proposed to optimize service migration though for different applications. Bienkowski et al. proposed competitive analysis for service migration in optimizing the server allocation in VNets in [9]. Arora et al. proposed some strategies for flexible server allocation in [10] following the previous work [9]. Although these works were not specifically for mobile application virtualization, they provide a precious insight on the performance evaluation for dynamic service allocation considering both user experience and operational cost. However, the analytical approach used in these works is topological and does not focus on the user mobility and interaction models. In our approach, we simulate the user mobility geographically based on the UMTS mobility models which provides an alternative performance preview in resource migration. Furthermore, the authors of [9] and [10] allow services being temporarily interrupted during migrations, which is not feasible for application virtualization services. In the proposed configuration, application services are available to users with reduced performance during hand-offs.

The rest of this paper is organized as follows. In Section II, we describe the proposed configuration aim to improve the user experience of application virtualization services. In Section III, we propose a VM-level hand-off protocol to handle the additional information exchange brought by the proposed server configuration. We specify our experiment design, settings, and cost metrics in Section IV. Then the simulation results given different parameter adjustments are presented in Section V. Finally, we conclude our work and outline some future work we expect to do in Section VI.

II. Proposed Configuration

Running application software on a remote server while creating an illusion that the client has full control of the software in hand is conceptually similar to the usage model of time-sharing mainframe computers in the 1960s [13]. Although the communication bandwidth between terminals and mainframe servers at that time was very low by modern standards, it did not affect the user experience thanks to the text-only display and short traverse distance. However, in recent application virtualization technologies which follow the same concept, such as Virtual Desktop Infrastructure (VDI) proposed by VMware [14], much more versatile and bloated content must be exchanged over much longer distances between clients and servers than their predecessors.

An infrastructure ready to offer mobile users application virtualization services includes base stations covering the whole service area, a core network connecting base stations and servers together, and a server hosting the services. A command sent by a mobile station has to travel over the wireless channel to the BS, go through the backhaul network to the server, and then make some changes on the server. Should any update corresponding to the command be sent to the MS, the information has to travel all the way backward. In order to reduce the network delay generated by long transmission distances among the backhaul network, we geographically deploy multiple servers among a wide area to serve their nearby MSs in the proposed configuration, instead of setting up one centralized server serving all MSs.

In the proposed configuration, each server connects to several nearby BSs to form a *local service group* (LSG). The area covered by the BSs of the same LSG is defined as the *local service area* (LSA). Every BS should belong to one LSG in order to provide the service all over the wireless network's coverage area. When a user demands a virtual application program, the server of the LSG, based on VDI [14] paradigm, starts a virtual machine (VM) dedicated to the user and launches the application software on top of it. The MS only handles inputs and outputs that interact with the VM at the server.

As long as the MS stays in the same LSA, the user can enjoy using application software with low response latency. If the MS moves from the original LSA to a nearby one, a hand-off at the VM level, which transfers the runtime environment to the server of the next LSG, is triggered. Therefore, a protocol to deal with the hand-off condition is required.

III. Hand-off Protocol

The purpose of the proposed hand-off protocol is to transfer minimum information required to recreate the runtime environment on a remote server, i.e., the *snapshot*, without

 ISBN: 978-1-61208-164-9

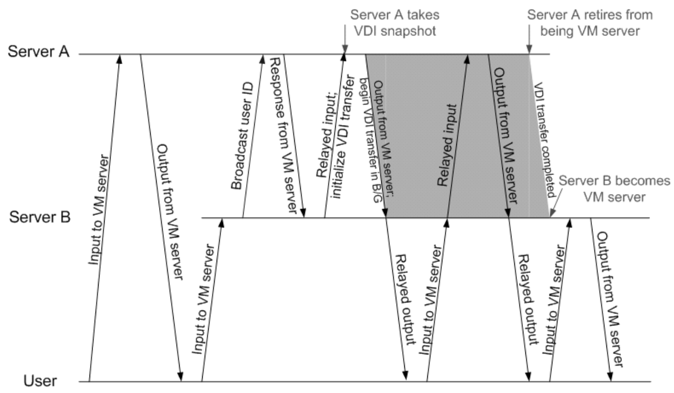

Fig. 1. Protocol timeline for an MS moving from Server A to Server B.

interrupting the service. To provide a seamless user experience during transmitting the snapshot, the next server has to record all inputs from the MS, relay all inputs to the previous server, and relay all output from the previous server to the MS, until the runtime environment resumes locally. The proposed hand-off protocol is described as below:

1) When an MS moves from Server A's to Server B's LSA and sends an input command, Server B notices a newcomer within its LSA.
2) Server B broadcasts the newcomer's identification to all geographically nearby servers.
3) Server A, which hosts the MS's runtime environment, i.e., its VM server, responds Server B's inquiry. Now Server B knows the newcomer's VM server is Server A.
4) Server B records and relays the user's input commands to Server A, signals Server A to transfer the runtime environment, and relays display updates from Server A to the newcomer.
5) Once Server A is signaled to transfer the runtime environment, it takes a snapshot.
6) Besides continually responding to the input commands relayed from Server B as the MS is still in its LSA, Server A also sends the snapshot to Server B in the background.
7) Once Server B receives the complete snapshot and recreates the runtime environment from the snapshot and base data, it internally feeds the input queue, which was recorded during the transition period, to the runtime environment. Therefore, the runtime environment state on Server B is synchronous with that on Server A after the snapshot was transferred.
8) Server A completely stops serving the MS, the MS's VM server is now Server B instead.

The timeline of the proposed hand-off protocol is illustrated in Fig. 1.

If the MS turned around and reentered Server A's LSA before the hand-off was completed, Server A can preempt the snapshot transmission and resume serving the MS as if the hand-off never happened. Since Server B relays all inputs to Server A while the MS is absent from Server A's LSA, aborting the hand-off procedure would not generate any noticeable glitch. This hand-off abortion mechanism can prevent unnecessary data transmission from moving VM servers back and forth if an MS were moving around the edge of an LSA.

On the other hand, if the MS moved to Server C's LSA before the hand-off was completed, Server C initializes another hand-off procedure with Server B. In addition to the snapshot, Server B has to transfer the input record before Server C joins the hand-off chain. We allow pipelining transmission to reduce hand-off periods and shorten subsequent hand-off chains in this scenario.

IV. Performance Evaluation

The proposed service architecture is designed to reduce interaction latency and thus provide more responsive user experience on remote controlled application virtualization services. However, due to the involvement of the hand-off protocol, the performance of the proposed service architecture depends highly on the probability of hand-offs, the geographical deployment of the BSs, and the configuration and capability of the backhaul network. The former two factors can be modeled by the test environments of existing communication systems, such as the well-published UMTS benchmarks [15]. On the other hand, the configuration and capability of the backhaul network can only be assumed based on reasonable technical and cost considerations.

A. UMTS Vehicular Mobility Model

The UMTS document [15] provides three different test environments, which are the Indoor Office, the Outdoor to Indoor and Pedestrian, and the Vehicular ones, for technology selection and evaluation. We simulated the hand-off behavior and evaluated performance of different infrastructure settings using the Outdoor to Indoor and Pedestrian mobility model in our previous work [12]. In this paper, we complete the performance evaluation by simulating the application virtualization services on the Vehicular test environment specified in the UMTS document.

As shown in Fig. 2, the UMTS rural vehicular test environment is a plain with no physical obstacle. Each MS's speed is fixed at 120 km/h. Each MS's moving direction is allowed to change up to 45° left or right every 20 meters with 20% chance. All MSs are initially uniformly distributed on the plain.

The BSs in the UMTS rural vehicular test environment are located at the dark grey dots in Fig. 2. Each BS has three directional antennae to serve tri-sectored cells. Each cell is assumed to be a hexagon and seamlessly tiles with each other. Each cell's radius R is either 2000 meters (for services up to 144kbit/s) or 500 meters (for services above 144kbit/s). Therefore, the minimum distance between two BSs can be 6 km or 1.5 km, respectively.

The original UMTS mobility model generates discontinuities on the boundaries of the test area. We consequently add some special traffic rules, known as *portals*, to eliminate the boundary discontinuities and allow the interaction among

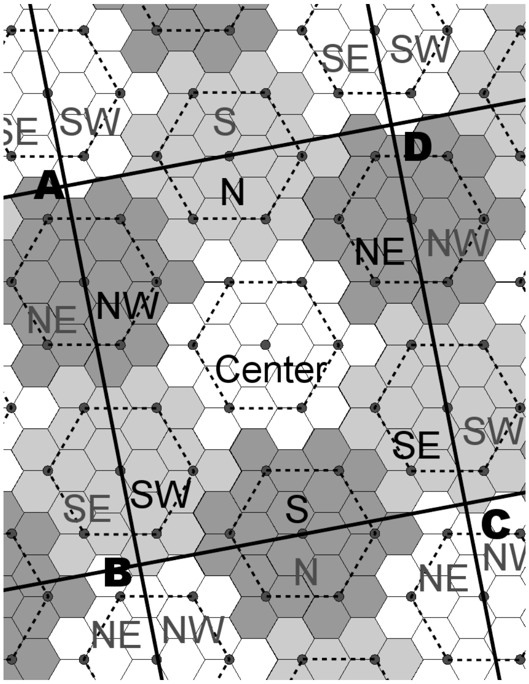

Fig. 2. The UMTS rural vehicular test environment with LSA arrangement.

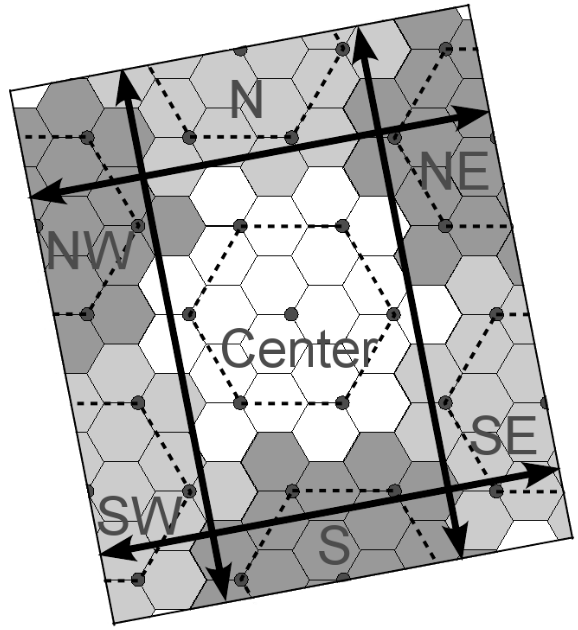

Fig. 3. Möbius County map with teleporting directions.

LSAs to be simulated and observed for an indefinite period of time. The characteristics of the portals will be detailed in the next section.

B. Möbius County

What interests us is the geographical relation between the service facilities and the MSs' moving space. As the method we conducted in our previous work [12], the first step is to define a sample area which can represent all the geographical characteristics of service infrastructure we need. We first group the BSs in Fig. 2 to form approximately hexagon-shaped LSAs which are optimized in both coverage and average transmission distance by deploying servers at the centers. As the urban counterpart, i.e., Möbius City, in our previous work [12], the sample area should include one complete LSA in the center and six neighboring halves. Given R and N, the number of the BS intervals per LSA's edge, if we align the origin to the server of an LSG, we define the Parallelogram ABCD surrounded by four straight lines, which are:

1) $\sqrt{3}x - 3(2N+1)y = -6\sqrt{3}R(3N^2+3N+1)$ on the north,
2) $\sqrt{3}x - 3(2N+1)y = 6\sqrt{3}R(3N^2+3N+1)$ on the south,
3) $\sqrt{3}(2N+1)x + y = -3\sqrt{3}R(3N^2+3N+1)$ on the west,
4) and $\sqrt{3}(2N+1)x + y = 3\sqrt{3}R(3N^2+3N+1)$ on the east.

as the sample area of our best interest. We can, therefore, crop out Parallelogram ABCD in Fig. 2 as our test area, where we call *Möbius County* as shown in Fig. 3, to represent every identical piece comprises the indefinite large test area.

Like Möbius City [12], assigning four logical LSGs in Möbius County is sufficient to figure out when, where, and how frequently an MS moves from one LSA to another. However, to apply the hand-off aborting mechanism, which was disabled in [12], we need to distinguish whether an MS is coming back to the LSA it just left or entering the LSA on the opposite side of the one it just crossed. Therefore, we have to assign an additional unique identification for each LSG.

The portals around Möbius County are also similar to those around Möbius City. Whenever an MS is about escaping from Möbius County, the portal teleports it to a proper location at the opposite side so that it reenters Möbius County. Therefore Möbius County can emulate a limitless test area. Since there is no street structure to align in Möbius County, the rules of the portals are much more simple and straightforward than of Möbius City:

1) For MSs about crossing the north boundary, teleport them to $\left(3R, -3\sqrt{3}R(2N+1)\right)$ from their current locations.
2) For MSs about crossing the south boundary, teleport them to $\left(-3R, 3\sqrt{3}R(2N+1)\right)$ from their current locations.
3) For MSs about crossing the west boundary, teleport them to $\left(-\frac{9R(2N+1)}{2}, -\frac{3\sqrt{3}R}{2}\right)$ from their current locations.
4) For MSs about crossing the east boundary, teleport them to $\left(\frac{9R(2N+1)}{2}, \frac{3\sqrt{3}R}{2}\right)$ from their current locations.

The teleport directions are shown in Fig. 3 as well.

The purpose of the portals is to eliminate all discontinuities except the MS's coordinates when it is moving out of the boundary: it keeps the same direction and speed, it associates with the same logical LSG, and preserves the geographical parameters relative to the service group's facilities. Thus, everything interests us is equivalent as the MS moving into an adjacent parallelogram area in a limitless test area.

C. Configuration of Backhaul Network

We assume a mesh-styled backhaul network as we did in [12]. Therefore, each BS only has direct links to its six neighboring BSs. In the mesh-styled backhaul network, network latency between a BS and the server depends on the number of nodes along the shortest path, the total length of the path, and the relay latency per node. The former two factors are related to the coordinates of the BS and the server, while the last one is varied to simulate different nodal transmission capabilities.

D. Performance Metric

We define the response time as the average time interval between when a user sends an input and gets an expected output update. The proposed server configuration is meant to improve the response time by reducing traverse delay along the communication route from each MS to the server which is hosting the service. Factors other than the traverse delay, such as computational capabilities provided by servers, would affect the user experience and the quality of our service. Most of them, however, either affect different configurations equally, or can be overcome with reasonable cost. The traverse delay is defined as:

$$T_{tv} = 2 \cdot \left\{ \frac{L_r}{V_r} + \frac{L_l}{V_l} + N_{rt} \cdot T_{rt} + N_{rl} \cdot T_{rl} \right\} \qquad (1)$$

where L_r is the distance of radio transmission, which is the distance between the MS and the BS covering it, V_r is the propagation speed of radio, which is the speed of light, L_l is the total length of wireline transmission in the mesh network, V_l is the propagation speed in wireline, which is approximately two thirds of V_r, N_{rt} is the number of nodes along the transmission path in the mesh network, T_{rt} is the average waiting time per node in the mesh network, which includes nodal processing delay, queuing delay, and transmission delay, N_{rl} is the number of servers which are receiving the snapshot and relaying data to/from the VM server, and T_{rl} is the processing and relay time per server in the hand-off chain. Obviously, all parameters, except V_r and V_l, depend on an MS's geographical location and hand-off state.

E. Hand-off Duration

Whenever a VM-level hand-off occurs, we set up an anticipated hand-off end time by adding hand-off duration to the current time. The hand-off duration is calculated by the following equation:

$$T_{ho} = T_x + \frac{L_s}{V_l} + N_s \cdot T_{rt} \qquad (2)$$

where T_x is the total time to deliver every bit of a snapshot to media, which is the summation of queuing delay, processing delay, and transmission delay of the snapshot, which is proportional to the size of the snapshot, L_s is the total transmission distance between the current and the next VM servers, and N_s is the number of nodes between two neighboring servers, which always equals to $2N + 1$ in this case.

F. Update Time Points and Cost Charging

Although we only calculate costs at position update points, updates actually take place when a hand-off is completed in addition to when an MS reaches an update position. At each update time point, T_{tv} and transaction counts are updated concurrently.

Whenever a position update comes at T_{now}, all hand-off end times registered in queue prior to T_{now} are update time points as well. The corresponding costs have to be calculated in retrospect according to the algorithm described below:

1) Define T_n as the n_{th} earliest hand-off end time in queue, L_{sn} as the total transmission distance between servers corresponding to the n_{th} earliest hand-off in queue, L_r, L_l, N_{rt}, and N_{rl} are the current cost parameters calculated by the MS's current position and hand-off status, and T_{last} as the previous update time.
2) If $T_{now} > T_0$, insert an update time point at T_0, calculate the transaction counts by the Poisson process given user input rate λ and time duration $(T_0 - T_{last})$, set $T_{last} = T_0$, subtract N_{rl} by one, subtract N_{rt} by $\{2N + 1\}$, subtract L_l by L_{s0}, update T_{tv} according to the new parameters, and remove T_0 and corresponding L_{s0} from the queues.
3) Redo step 2 until $T_{now} < T_0$ or the queue is emptied.
4) Calculate the transaction counts by the Poisson process given λ and time duration $(T_{now} - T_{last})$, update T_{tv} according to the new parameters, and set new $T_{last} = T_{now}$.

As specified in the UMTS rural vehicular mobility model, we update the MSs' positions every 20 meters. Since a hand-off may occur at the same time, we have to handle the extra cost brought by it as well. When a new hand-off occurs with a position update at current time T_{now} while the previous update time is T_{last}, and every hand-off end time earlier than T_{now} is already treated with the above algorithm, we use the following algorithm to update the cost parameters:

1) Register the new hand-off end time and the corresponding L_s in the queue.
2) Increment N_{rl} by one.
3) N_{rt} is recalculated by the MS's current position and added by $\{N_{rl} \cdot (2N + 1)\}$.
4) Let L_l equals to the summation of all L_s's in queue.
5) T_{tv} is then updated accordingly.
6) The transaction counts are calculated by the Poisson process given λ and time duration $(T_{now} - T_{last})$, and then set new $T_{last} = T_{now}$ for the next update.

Since the variation of the geographical parameters is negligible along the 20 meters (or less) long path, every transaction in an update interval is charged with identical T_{tv} to reduce the computational complexity. Note that T_{tv} updated at time T is applied to the transactions which occur *after* T, while the transaction counts calculated at T are placed in the time interval ended at T.

 ISBN: 978-1-61208-164-9

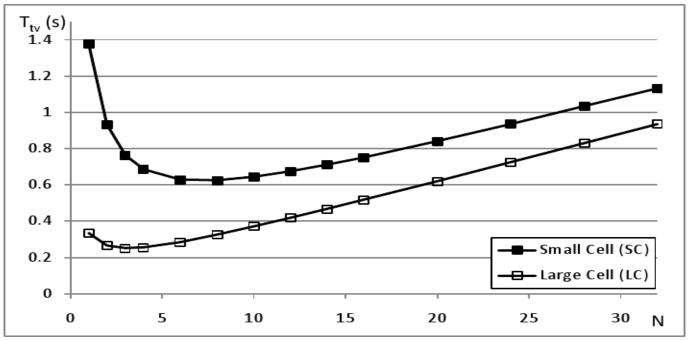

Fig. 4. Simulation results of different N of both cell configurations given $T_{rl} = 0.5s$ $T_{rt} = 20ms$, and $T_x = 600s$, $\lambda = 1.0$.

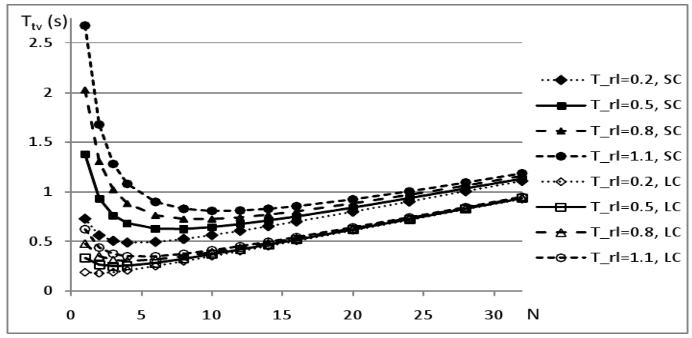

Fig. 5. Simulated T_{tv}'s of both cell configurations given $T_{rl} = 0.2s$, $0.5s$, $0.8s$, $1.1s$ and $T_{rt} = 20ms$, $T_x = 600s$, $\lambda = 1.0$.

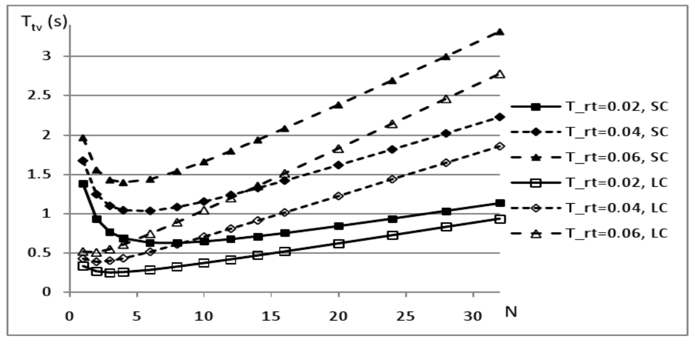

Fig. 6. Simulated T_{tv}'s of both cell configurations given $T_{rt} = 20ms$, $40ms$, $60ms$ and $T_{rl} = 500ms$, $T_x = 600s$, $\lambda = 1.0$.

G. Traverse Time Accounting

The average T_{tv} per transaction is calculated at the end of 100,000 independent simulations, each lasting 86400 seconds. The simulation results of variable N, T_{rt}, T_{rl}, T_x, and λ for both $R = 2000m$ or $500m$, are presented in the following section.

V. Simulation Results

We first simulate how the size of LSAs affects T_{tv} given *nominal* parameters, which are $T_{rt} = 20ms$, $T_{rl} = 500ms$, $T_x = 600s$, and $\lambda = 1.0$. The simulation results of both R settings are shown in Fig. 4.

As we can see in Fig. 4, both T_{tv}'s bear a strong resemblance in shape to the counterpart in [12] despite the significantly different mobility models. T_{tv}'s are high in small LSA configurations due to the higher hand-off occurrence rate. As N increases, T_{tv}'s first descend, level for several N's, and then linearly ascend. The descending for low N's is due to the reduction of hand-off occurrences. The smooth ascending for higher N's is caused by the higher average number of the nodes along the backhaul route and the longer average transmission distance while the hand-off occurrence rate is too low to matter. The flat bottom in between is the result of the two effects competing with each other.

Note although we compare two cell configurations, $R = 2000m$ and $R = 500m$, in the same figure, each LSA of the former one is in fact 4 times larger than of the latter one. Therefore, each MS encounters much fewer hand-offs in the large cell configuration than in the small cell one. We can also observe slightly steeper ascending for higher N's in the large cell configuration than in the small cell one due to the higher propagation delay brought by the longer wireline and wireless transmission distances.

We can conclude that in this case, setting $N = 4$ for the large cell configuration, and $N = 8$ for the small cell one, are optimal in reducing average T_{tv} and keeping the total number of the servers low, which also means lower deployment and maintenance cost.

Since the above quantitative conclusion is only applicable in this set of parameters, we adjust each parameter in the nominal set and compare the results to see how it affects T_{tv}'s as functions of N in the following subsections.

A. Effect of T_{rl}

T_{rl} only participates in hand-off conditions. In this simulation, we set T_{rl} to $200ms$, $800ms$, and $1100ms$, and see how it affects both T_{tv}'s. Both simulated T_{tv}'s in large and small cell configurations as functions of N and T_{rl} given $T_{rt} = 20ms$, $T_x = 600s$, $\lambda = 1.0$ are shown in Fig. 5.

As we can see in Fig. 5, higher T_{rl} significantly increases T_{tv}'s in small LSA configurations due to the higher occurrence rate of hand-offs. As N increases, T_{tv}'s in each cell configuration given different T_{rl}'s have a tendency to converge together since the hand-off occurrence rate is dramatically reduced and thus renders the effect of T_{rl} insignificant. In the large cell configuration, T_{tv}'s converge more significantly and earlier due to the extremely low hand-off occurrence rate.

B. Effect of T_{rt}

Higher T_{rt} amplifies the influence of transmission distance. The simulated T_{tv}'s in both cell configurations as functions of N and T_{rt} given $T_{rl} = 0.5s$, $T_x = 600s$, $\lambda = 1.0$ are shown in Fig. 6.

Fig. 6 shows the comparison of T_{tv}'s of both cell configurations as functions of N given $T_{rt} = 20ms$, $40ms$, and $60ms$. Besides the resemblance in shape to the counterpart in [12], we can also notice that T_{rt} is a more decisive factor for the large cell configuration's performance due to the low hand-off

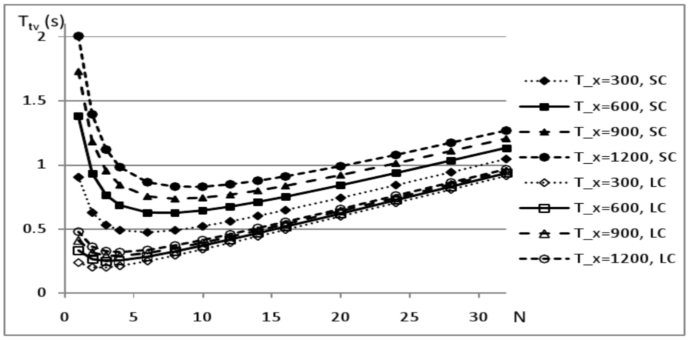

Fig. 7. Simulated T_{tv} of both cell configurations given $T_x = 300s$, $600s$, $900s$, $1200s$ and $T_{rt} = 20ms$, $T_{rl} = 0.5s$, $\lambda = 1.0$.

occurrence rate and the long average communication distance in each LSA. Even $N = 1$ can be preferable if T_{rt} is greater than 60ms in the large cell configuration.

C. Effect of T_x

T_x only affects the cost brought by hand-offs. A higher T_x may mean a larger snapshot file, a longer hand-off initialization time, or a longer queuing delay. How T_x affects T_{tv} is represented in Fig. 7.

Similar to the counterpart in [12], T_{tv}'s of each cell configuration as functions of N given different T_x's are virtually parallel for high N to each other and show very little tendency to converge as N increases. However, slightly higher optimal N brought by higher T_x in both configurations is still observable.

D. Effect of λ

Although we have shown that user input rate λ was not a relevant parameter in [12], we still simulate T_{tv}'s as functions of N given different user input rates λ in Möbius County. We again confirm that the property doesn't change in the UMTS rural vehicular mobility model.

However, we should keep in mind that the user experience depends more on the interactivity of the application software than on the absolute response latency.

VI. Conclusion And Future Work

In this paper, we complete the performance evaluation of the proposed application virtualization configuration and the corresponding hand-off protocol by applying the UMTS rural vehicular mobility model. In addition to the model Möbius City which we proposed in our previous work [12], we propose Möbius County using a similar concept to enable MSs to move in the test environment based on the UMTS rural vehicular mobility model indefinitely without dealing with any boundary condition. We simulate the network delay as a result of MSs' movements and the occurrences of VM-level hand-offs in Möbius County given variable sizes of LSAs, server relay latencies, nodal relay costs, and snapshot transmission durations.

Möbius County, combined with Möbius City, can provide a performance preview of network infrastructures aimed at improving mobile application virtualization services in large scale unknown environments. We can also design a benchmark framework specific for distributed application virtualization services based on the proposed simulation environments.

In this paper, we employ deterministic infrastructure delay parameters and a simple usage model to evaluate the performance. We will introduce more sophisticated usage and infrastructure delay model to facilitate more precise mobile application virtualization service simulations. Furthermore, besides the benefit of lowering the average response latency, the distributed application virtualization service configuration and the hand-off protocol can also be applied to load balancing and fault tolerance for better resource management and service robustness. We will investigate these potential applications in the future as well.

References

[1] Sunwook Kim et al., "On-demand software streaming system for embedded system", *WiCOM 2006 International Conference on Wireless Communications, Networking and Mobile Computing*, 22-24 Sept. 2006, pp. 1-4.

[2] EMA Report: "AppStream: Transforming On-premise Software for SaaS Delivery - without reengineering"

[3] Joeng Kim; Baratto, R.A.; Nieh, J., "An application streaming service for mobile handheld devices", *SCC'06 IEEE International Conference on Services Computing*, Sept. 2006, pp. 323-326.

[4] VMware Inc., "VMware ThinApp: Agentless Application Virtualization Overview".

[5] Ana Fernandez Vilas et al., "Providing web services over DVB-H: mobile web services", *IEEE Transactions on Consumer Electronics*, Vol. 53, No, 2, May 2007, pp. 644-652.

[6] VMware Inc., "VMware MVP (Mobile Virtualization Platform)", http://www.vmware.com/products/mobile/overview.html, Retreived 7 Aug. 2011.

[7] Apple Inc., "App Store Review Guidelines for iOS Apps", 2.7 and 2.8, http://developer.apple.com/appstore/guidelines.html, Retrieved 9 Sep. 2010.

[8] Google Inc., "Android Market Developer Distribution Agreement", 4.5, http://www.android.com/us/developer-distribution-agreement.html, Retrieved 22 Feb. 2011.

[9] Marcin Bienkowski et al., "Competitive Analysis for Service Migration in VNets", in *Proc. 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, 2010, pp. 17-24.

[10] Dushyant Arora et al., "On the benefit of virtualization: strategies for flexible server allocation", *Hot-ICE'11 Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, 2011.

[11] Chung-Ping Hung and Paul S. Min, "Infrastructure arrangement for application virtualization services", *I2TS 2010 The 9th International Information and Telecommunication Technologies Symposium*, 13-15 Dec. 2010, Vol. 1, pp. 78-85.

[12] Chung-Ping Hung and Paul S. Min, "Service area optimization for application virtualization using UMTS mobility model", *ICOMP 2011 International Conference on Internet Computing*, 18-21 July 2011, pp. 128-134.

[13] L. P. Deutch and B. W. Lampson, "SDS 930 Time-sharing System Preliminary Reference Manual", Doc. 30.10.10, Project Genie, Univ. Cal. at Berkeley, April 1965.

[14] VMware Inc., "Virtual Desktop Infrastructure".

[15] ETSI. "Universal Mobile Telecommunications System (UMTS); selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03, version 3.2.0)". Technical report, European Telecommincation Standards Institute, Apr. 1998.