Lilliput meets Brobdingnagian: Data Center Systems Management through Mobile Devices

Saurabh Bagchi[1,2], Fahad Arshad[1]
[1] *Purdue University*
Jan Rellermeyer[2], Thomas Osiecki[2], Michael Kistler[2], Ahmed Gheith[2]
[2] *IBM Research Austin*

Abstract

In this paper, we put forward the notion that systems management for large masses of virtual machines in data centers is going to be done differently in the short to medium term future—through smart phones and through controlled crowdsourcing to a variety of experts within an organization, rather than dedicated system administrators alone. We lay out the research and practitioner challenges this model raises and give some preliminary solution directions that are being developed, here at IBM and elsewhere.

I. Introduction

Systems management of virtual machines and the virtualization infrastructure (hypervisor, virtual LAN, etc.) is a crucial element of running data centers. Such systems management has two broad elements within it—monitoring for detecting any anomalies, be they security failures, correctness failures, or performance problems; and, then taking corrective action if any anomalous pattern is detected. The monitoring is done at various levels of the stack in the virtualized environment, from the chassis-level (using an advanced management module on the server), the hypervisor level, the networking level, and all the way up to the application level. A plethora of such monitoring tools exist, which together generate a wealth of data about the target system. Then, the monitored data is aggregated and visualized (through infrastructure such as Splunk [1]), and system administrators inspect this to determine if mitigation action needs to be taken. There is an urgency in acting on monitored information *if* it indicates that a failure is imminent or has already happened and hence, control mechanisms have been developed, most rather application-specific.

In this paper, we hypothesize that a significant change is happening in the way such systems management is done for virtualized environments and that the pace of change will accelerate in the immediate future. The change is in two dimensions — the first dimension is the platform that is being used for the management and the second dimension, enabled by the first, is the number of people involved in the systems management activity. Regarding the first dimension, we see that it will become increasingly convenient to perform these actions from mobile devices. This trend is being driven by the continuing increase in the penetration of smart phones in the market, the increase in their compute power and visual rendering capability, the mobility of the workforce (such as, increasing prevalence of telecommuting), and the desire to access the systems management information from geographically anywhere. Regarding the second dimension, we believe that virtualization brings in an unprecedented level of interaction among the vertical layers of the software stack as well as among the horizontal layers. Thus, as an example, cache usage pattern of one VM can affect the performance of application running in a different VM on the same physical machine [2]. With this comes the need to tap into a diverse knowledge base because one system administrator cannot be expected to be an expert in performance tuning of a web server (application level) to security to guard against resource leaks among VMs (infrastructure level). Rather, it will become necessary for multiple people within an IT organization to cooperate in a loosely structured manner to determine a mitigation action.

The first dimension raises three important research and practitioner challenges:

1. We need to ingest large amounts of data on small form factor and resource constrained (relative to a traditional desktop-class machine) devices. An important constrained resource is the wireless bandwidth between the servers and the mobile clients and another is the amount of energy available to the device.
2. There is a mismatch in terms of the dependability characteristics of the servers and the mobile clients. The servers are often very secure while the mobile clients are not — security vulnerabilities and dependability flaws in the software architecture of these systems are being discovered routinely [4], the clients roam outside the comforting periphery of the company firewalls, and the clients are *not* always-available, always-on devices.
3. There is a need for sophisticated visualization techniques for displaying the large amounts of data from the large number of managed servers on the small form factor mobile clients. The display must be actionable so that the administrators can initiate timely mitigation actions in case they are needed.

The second dimension (larger number of people ingesting the systems management information and a subset of these acting on them) will introduce two significant research and practitioner challenges: 1. Current procedures

978-1-4799-0181-4/13/$31.00 ©2013 IEEE

for handling failures often focus diagnostic efforts on one layer or system component at a time, with transfers to other layers/components handled in a serial fashion. This increases the time to resolve the problem. We believe a better approach is to enable coordinated, simultaneous investigation of problems by multiple administrators or even domain experts within the organization, that are not official sysadmins. Mobile systems management applications can play a key role in this mode of operations by reducing communication time among administrators and helping in exchange of problem context among the involved people, which is needed for the problem resolution.

2. Can we leverage the overlap between the interests of multiple mobile end points and their geographical proximity to efficiently use the restricted cellular bandwidth available going out to the mobile devices. So it may be possible to reduce this contention by using near field communication (Bluetooth, Wi-Fi, etc.) between multiple close-by mobile devices that share interest or using an aggregator server that serves multiple mobile clients. This has to be done in a manner that is cognizant of the mobility of the clients, and security implications.

We believe that the challenges raised above are not solvable with a gradual evolution of today's solutions. Today's solutions for systems management for dependability rely on the information being consumed by resource rich platforms, and in tightly controlled settings — privileged user accessing information from well provisioned, and perfectly well known, execution environment. What we anticipate will negate both of these assumptions. Further, systems management work with mobile devices so far has looked at the opposite picture to what we are describing here. Existing work has shown how to manage a large number of mobile devices from servers, such as, how to push out a software patch to multiple smart phones, possibly using peer-to-peer dissemination among close-by phones.

II. Management from Resource-Constrained Mobile Devices

A. Optimizing for Communication

Carrol and Heiser have identified communication to be one of the dominating factors of power consumption on smartphones (for applications that require network connectivity) [5], which is consistent with the experience from earlier battery-powered embedded systems like sensor network nodes (e.g., [6]). Mobile systems management is inherently communication-intensive since the mobile client requires periodical status updates from the monitored machines and the fidelity of the systems management application is a function of the sampling rate. One of the key challenges is therefore to minimize the number of messages that need to be received to still reliably detect failures and receive other relevant information at the mobile device. In this regard, communication patterns like publish-subscribe (using a push paradigm) are clearly preferable to periodic polling. Unfortunately, most hardware management modules (Advanced Management Module (AMM) [7], Baseboard Management Controller (BMC) [8]) only support protocols that inherently require polling, a problem that we hope to see addressed by the hardware vendors when mobile systems management gains traction. Clearly, more research is needed to develop alternative systems management protocols which allow the mobile client to express sophisticated filters to the hardware management module. Alternatively, a proxy in the cloud could act as a middle box, perform the constant polling of the managed machines, and then serve as a publish-subscribe server for multiple clients, an option that we further elaborate on in Section III.

The problem of systems management can in broad terms be seen as a specific instance of the rare event detection problem: it can be expected that non-events (e.g., system parameters that show no abnormal patterns) occur with a much higher frequency than events of interest (e.g., system failures). In akin problem domains, much effort has been put into avoiding high sampling rate while still detecting the rare event with high likelihood or within predictable time bounds (e.g., [9], [10]). Many of these approaches, however, require a model of the distribution of the events or a precise understanding of their correlations—little of this is known currently in the managed systems. Finally, given that a certain number of messages per time period cannot be avoided to keep the application up to date, system-level optimization like traffic shaping [11] can help to minimize the power consumption to transmit and receive these messages.

B. Mismatch in Dependability Characteristics

Platforms for systems management have traditionally been considered as mission-critical as the managed systems themselves since ultimately failures on their side could mask failures of the managed system or could even compromise their security. However, mobile devices are much harder to physically control since they can easily leave access-controlled zones, can be stolen, or lost. Once a mobile device is in the hands of an adversary, OS-level protection can usually be easily circumvented since the majority of the smartphones available today have known vulnerabilities and tools exist to *root* these devices. In times where employees bringing their own mobile devices and using them for business work is an increasingly common pattern, it is not even clear how an enterprise can enforce policies on employee-owned devices. Therefore, mobile devices need to be considered unsafe environments and application-level security needs to compensate for the lack of enforceable physical access control and operating system protection.

Another problem is the reliability of the mobile device. Even when not considering the problem of network

connectivity and battery lifetime, mobile clients are often perceived (and justifiably so) to be more unreliable than well-managed Linux-based desktop-class machines. Reasons could be the lower maturity of the platforms or the high pressure on the manufacturers due to shorter development time. Little research beyond specific problems has been done so far to shed light on the question how reliable mobile phones actually are in demanding tasks such as systems management. We have done some preliminary work in this regard for the Android and the Symbian OS platforms [4] and more in-depth for interprocess communication in Android [12]. Our preliminary results have shown that some simple flaws in software components reused over and over again (such as, the web toolkit) contribute to a large fraction of user-visible failures. Second, the lack of input validation for messages received from external source (a user) or from other software components leads to a significant source of vulnerability.

C. Visualizing a Needle in a Haystack

Most systems management suites provide some means for administrators to request and receive notifications or alerts ("the needle") of outages, failures, or behavior that may be indicative of an imminent failure. With the introduction of mobile systems management applications, there is also the option of delivering alerts through "push notifications" presented directly by the application. This delivery method has a key advantage over the traditional "out of band" channels, such as, text or email messages, in that the mobile application can use information from the alert to guide the administrator to the appropriate dialogs within the application for diagnosing and correcting the problem. However, the subsequent processing for root cause analysis is best done on server-class platforms. Some of the analysis is computationally heavy, incorporating for example, pattern matching and machine learning algorithms. Limited screen real estate makes it challenging to display large amounts of data in a comprehensible way, which is, however, a key aspect of traditional systems management. As indicated before, the majority of the data that can be collected from systems management modules is informative, not essential, most of the time, unless there is a condition which requires action. Another design principle should be that alert data needs to be shown in a summary view for all machines in the data center as the central element of the application. Starting from this view, the mobile application can then selectively present more detailed data as well as actionable items to the user in a drill-down fashion. While this approach has been proven successful in our ongoing work with IBM Remote, a fundamental problem remains: relevance is a highly subjective classifier and there remains the need to do much work in user-driven mobile user interface design.

III. Crowdsourcing System Management

In this second dimension, we posit that crowdsourcing of systems management is a desirable outcome and this is enabled by systems management through mobile devices. The term "crowd" should not be overconstrued to mean just about anybody in the enterprise; rather, this will refer to someone who is a domain expert in one of the vertical layers in the system, such as, facilities expert (HVAC, etc.), networking expert, hypervisor expert, virtualization security expert, application expert, etc. The diversification will help by allowing a parallel workflow whereby multiple experts, each from their mobile device, can examine the problem symptoms to triangulate the problem. The different skills, independent thinking, access to different resources, skills in multiple layers of the stack, etc. are all expected to aid in timely systems management. The mobile application, unified across all these experts' mobile devices, can bring them all together into this kind of integrated system administration. The geographic diversity that may result from the mobility of the involved people can help in some investigations, such as, troubleshooting network bottlenecks or firewall rules.

We do not wish to be Pollyannaish in laying out the above vision. There are many reasons for *not having* crowdsourcing of the systems management tasks, such as, possibility of conflicting updates and security breaches. Also, we are aware that a lot of systems management is automated, but we still have significant human involvement for problem diagnosis, more so than problem detection. First, we believe that the increasing need for coordinated administrative actions will drive systems management applications to appropriate features of social media sites, such as "friending" of related systems, applications, and administrators, "feeds" of activities by systems or administrators, and "circles" to allow coordination and collaboration in problem solving. By adopting features and patterns of group communication similar to social media applications, systems management applications will enable rapid adoption of these features without special education or documentation. What comes with the territory is a loose level of synchronization rather than something on the lines of commit protocols. We would argue that current practice is even looser levels of synchronization. Second, we believe lessons from role-based access control (RBAC) can be adapted to handle the security implications. Fine-grained roles can be assigned, say for experts in different layers of the stack, and thanks to significant progress in RBAC, these roles can be handled in a sophisticated manner allowing hierarchies, overlaps, and transient existence.

With respect to the second aspect of this thrust, we believe there will be commonalities of interests among multiple mobile devices. Some of these commonalities will be driven by proximal geographic location, such

as, two administrators in the lab, based on an initial problem alert, may drill down requiring further data near concurrently in time. Considering that direct data connection between the mobile client and the managed server is likely to be expensive (consumes more energy, cellular data connection has contention, etc.), there is scope for a middle tier that provides aggregation services and feeds multiple mobile clients. There is some prior work with this model [13], [14]. The unsolved challenges in this are how timely can the updates be, what are the security implications since different persons have different access privileges to different pieces of information.

IV. Case Study

IBM Mobile Systems Remote (short IBM Remote) is an iPhone and Android application developed by IBM Research for managing IBM server systems, with a software architecture that enables it to be generalized to other server platforms. It has a View-Cache-Engine (VCE) Architecture [15] in which the mobile application fetches certain data items of interest to system management from the servers, such as temperature of the chassis and speeds of the different fans. *Views* express interest in certain data items to a *Cache*, which in turn is updated asynchronously by a communication *Engine* whenever it receives an update. The cache in turn updates the views that have subscribed to the new piece of data. How often a data item gets updated is determined by the *Freshness* of the data item, which is an intuitive way for the administrator to specify how timely does a data item have to be. We find that the freshness criterion varies from the relatively static data items, such as the machine hostname and MAC address, to slowly-varying dynamic information, such as up/down status, to fairly dynamic information, such as fan speed.

In Figure 1 we show two screenshots of IBM Remote. The left image shows the main view of the application where each cell is a different management endpoint. In this case they are IBM BladeCenters managed by an Advanced Management Module (AMM) — an AMM is a hot-swap BladeCenter module that is used to configure and manage all installed BladeCenter components [16]. The application talks a proprietary TCP-based protocol to get data from the AMM. Each cell contains prioritized information to give a high level overview of the machine, such as name, label, and machine type, and its health status — a cross indicates a critical error, an exclamation mark some non-critical problem, and a check that everything is fine with the machine. A timestamp for each data item update is kept in the cache and the time of the last update is displayed next to the connectivity symbol to give the user an idea of the freshness of the data.

If a user clicks on an individual machine cell on the main page they are taken to the front view of said BladeCenter as shown in the right image. We see each of the fourteen individual blades in the chassis and can click on any one to get further information. Below each blade is a power button which shows the current powered state and which the user can click to power each blade on or off, an example of the control part of system management.

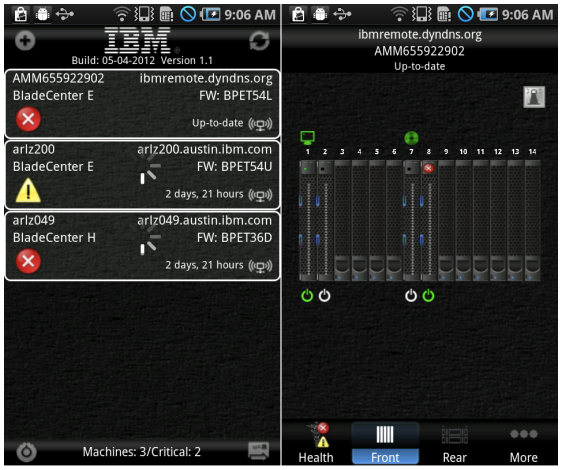


Fig. 1. **The main view on the left and front view of a BladeCenter chassis on the right**

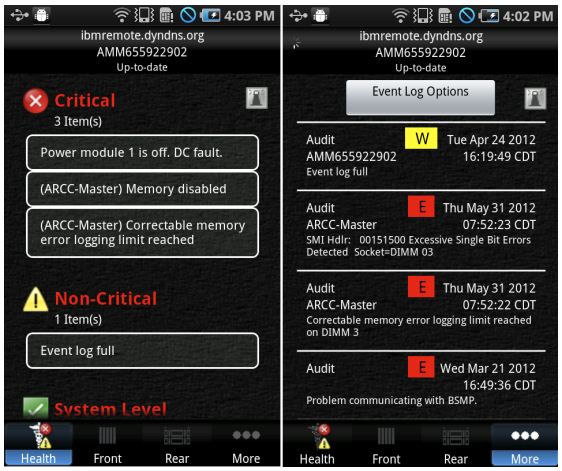


Fig. 2. **The health view on the left and event log on the right for a BladeCenter chassis**

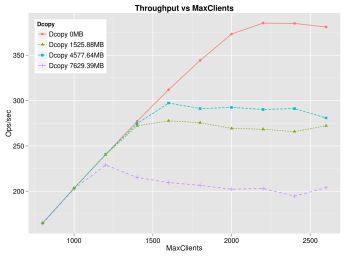
Figure 2 shows the health view and the event log view. The health view is broken down into critical, non-critical, and system level health messages giving a comprehensive look into the system. Due to the verbosity of event logs, the right image is filtered only to show warnings and errors. This is an example of the necessary level of filtering before visualizing on a mobile device so that a system administrator can take timely action.

Empirically, we find that our concept of *Freshness* improved battery performance [15], since certain data items now did not have to be refreshed as often, which reduced communication, the greatest component of the

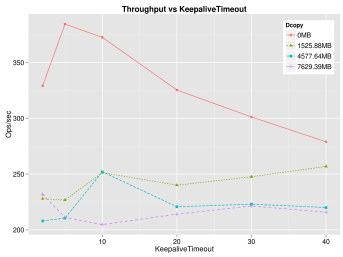
energy consumption. There was a 17% relative energy saving compared to the case where no freshness criterion is specified and the mobile device updates all data items every 30 seconds. Notably, the freshness optimization does not affect the time to detect a relevant event by the administrator. We chose a publish-subscribe based model, with the data cache being the publisher and the view being the subscriber. This reduces network traffic because several views have interest in overlapping sets of data items.

A class of bugs that highlights the challenges of systems management in virtualized environments is configuration bugs. Consider that for a production data center running KVM virtual machines we run an application — a Web 2.0 benchmarl called Olio [17] and two interfering workloads on two separate VMs. Olio is a benchmark mimicking a social network application and has been used widely for evaluating PHP web-based applications and even made a part of other benchmark suites. Olio runs on the Apache HTTP server (version 2.4, in multi-threaded i.e., *worker* mode) and uses a PHP accelerator for dynamic content called PHP Fastcgi Process Manager. In Figure 3, we show the impact of three configuration parameters on the application throughput, the first two being Apache configuration parameters and the third a PHP accelerator parameter. We find that the choice of best configuration parameter for Apache not only depends on workload intensity, but also on the amount of interference arising from co-located VMs (the DCopy size quantifies interference). Further, the configuration parameters have non-linear dependencies among themselves and this dependency also changes with interference (not shown here). This result tells us that there will arise the need for systems management tasks (such as, tuning these parameters) due to unpredictable events—consider for example, the complex relationships between performance and interference. Also, proper mitigation action will require the administrators to ingest fairly complex data, necessitating effective applications on mobile devices.

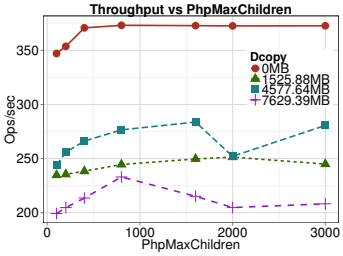
Design challenges and how we solved them. We faced several design challenges in building IBM Remote and the configuration engine and we list the three most important ones. First, the real estate on smart phones is really precious and therefore we found, through extensive trial with system administrators, that it was best to provide a summarized view of the health of a machine highlighting error conditions and the ability to drill down, as desired by the admin, to investigate the cause of the problem. Second, we found that the configuration parameters have dependencies, even non-linear dependencies, among them. Therefore, we solved this partially through having a table lookup of sets of parameter values for different operational regions, such as, high number of users and high amount of database load, high number of users and lots of short requests. This can be improved



(a) Throughput vs. `MaxClients`



(b) Throughput vs. `KeepAliveTimeout`



(c) Throughput vs. `PhpMaxChildren`

Fig. 3. Choice of optimal parameter values with varying Dcopy intensity. For all experiments, workload size is 2,000, chosen default values are `MaxClients`=2,000, `KeepAliveTimeout`=10, and `PHPMaxChildren`=2,000. In each experiment, one of the parameters is varied while others are kept constant at their default values.

through solving it in a rigorous optimization manner. Third, for the mobile client, we needed it to work with a variety of communication mechanisms available on different platforms, such as, pull-based or push-based (the latter is not efficiently supported in some platforms). Our compromise is to provide a push-based interface through *Views* and pull-based through the *Cache*.

V. Related Work

Prior work on system management in the mobile context has focused on managing mobile devices, which is the opposite direction to the control flow that we are proposing here. The limited amount of work in our direction has focused on mobile agents for monitoring servers [18], [19]. The sophistication in this work lies in having a dynamic set of agents whose monitoring policies can be changed flexibly and remotely. Some work has focused on adapting the agent-based monitoring to the security policies of enterprises [20]. UCSand [21] is the first Android app with a GUI for the Cisco Unified Computing

Systems (USCs) monitoring and control. The app utilizes the XML-API of the UCS-Manager but does not update periodically like IBMRemote, the user must close the app and open it again to refresh. PCMonitor [22] is a commercial product designed by MMSOFT Design Ltd. for the purpose of monitoring PCs running Windows and major Linux distribution from a mobile device, including the iPhone, Android, and Windows 7 mobile phones. PCMonitor relies on middleware that monitors on behalf of the mobile device, while IBMRemote does the actual communication. VMWare vCenter Mobile Access [23] (VCMA) is a fully configured and ready to run virtual appliance that is required on the server side to manage a datacenter from mobile devices. Administrators can perform various activities in their VMware environments using a mobile browser or the iPad application. The application lacks a native application, unlike IBMRemote, which causes it to not have as fluid of a user experience. Hewlett-Packard recently announced a mobile application for configuration, monitoring, and management for their new HP Proliant-8 family of systems [24]. This announcement shows that some vendors are already working to address the points we have raised in this paper.

VI. Conclusion

In this paper, we have laid out a vision that we believe is close at hand — mobile clients, such as smart phones, being used to manage large masses of physical and virtual servers. This turns some long-held practices and principles of systems manegement on their head. We identify two fundamental ones. The first is that management will now be done through multiple resource-constrained mobile devices, which have a different dependability characteristic than the managed devices. The second is that systems management will take on a flavor of crowdsourcing, albeit one where "crowd" is narrowly defined to be experts in one (or more) domains of the managed systems and the management function is still loosely controlled. For these two fundamental changes to be beneficial rather than deleterious to the vision, we outline solution directions and lay out a concrete case of a mobile application for systems management called IBM Remote, developed at IBM Research and being used in limited engagements. We highlight some of the requirements for systems management through mobile devices while troubleshooting configuration-related bugs.

References

[1] Splunk Inc., "Splunk for Application Management," http://www.splunk.com/web_assets/pdfs/secure/Splunk_for_Application_Management.pdf.

[2] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift, "Resource-freeing attacks: improve your cloud performance (at your neighbor's expense)," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 281–292.

[3] D. Williams and B. Newton, "Dell-BMC Industry Insights: Data Center Automation Your Path to the Cloud," http://www.bmc.com/products/documents/54/69/215469/215469.pdf, 2011.

[4] A. Kumar Maji, K. Hao, S. Sultana, and S. Bagchi, "Characterizing failures in mobile OSes: A case study with Android and Symbian," in *IEEE 21st International Symposium on Software Reliability Engineering (ISSRE)*, 2010, pp. 249–258.

[5] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *USENIX ATC*, 2010.

[6] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *SenSys*, 2004.

[7] IBM, "Advanced Management Module," http://publib.boulder.ibm.com/infocenter/bladectr/documentation/index.jsp?topic=/com.ibm.bladecenter.8886.doc/dw1fs_c_advanced_management_module.html.

[8] Dell Inc., "Remote Management with the Baseboard Management Controller in Eighth-Generation Dell PowerEdge Servers," www.dell.com/downloads/global/power/ps4q04-20040110-Zhuo.pdf.

[9] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in *At the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[10] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, 2005.

[11] M. Hoque, M. Siekkinen, and J. Nurminen, "On the energy efficiency of proxy-based traffic shaping for mobile audio streaming," in *Consumer Communications and Networking Conference (CCNC)*, 2011.

[12] A. K. Maji, F. A. Arshad, S. Bagchi, and J. S. Rellermeyer, "An empirical study of the robustness of inter-component communication in android," in *DSN*, 2012.

[13] M. Pitkanen, T. Karkkainen, and J. Ott, "Opportunistic Web Access via WLAN Hotspots," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2010, pp. 20–30.

[14] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan, "Mobile Data Offloading through Opportunistic Communications and Social Participation," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 821 –834, may 2012.

[15] J. S. Rellermeyer, T. H. Osiecki, E. A. Holloway, P. J. Bohrer, and M. Kistler, "System management with ibm mobile systems remote - a question of power and scale," in *13th International Conference on Mobile Data Management (MDM)*, 2012, pp. 1–6.

[16] IBM, "IBM BladeCenter blade server," http://www.ibm.com/systems/bladecenter/index.html.

[17] Olio, "Olio: Web 2.0 application," http://incubator.apache.org/olio, 2013.

[18] A. Tripathi, T. Ahmed, S. Pathak, M. Carney, and P. Dokas, "Paradigms for mobile agent based active monitoring of network systems," in *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, 2002, pp. 65 – 78.

[19] A. R. Tripathi, D. Kulkarni, H. Talkad, M. Koka, S. Karanth, T. Ahmed, and I. Osipkov, "Autonomic configuration and recovery in a mobile agent-based distributed event monitoring system," *Software: Practice and Experience*, vol. 37, no. 5, pp. 493–522, 2007. [Online]. Available: http://dx.doi.org/10.1002/spe.777

[20] A. Koliousis and J. Sventek, "A trustworthy mobile agent infrastructure for network management," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 383 –390.

[21] W. V. Schaik, "UCSand," https://market.android.com/details?id=net.ecliptic.ucsand, Aug 27, 2011.

[22] MMSOFT Design Ltd., "PCMonitor," http://mobilepcmonitor.com/, 2011.

[23] V. Labs, "VMWare vCenter Mobile Access," http://labs.vmware.com/flings/vcma, 2011.

[24] A. Shah, "HP to Release Server Management Apps for iOS, Android," February 2012.