Energy efficient mobile M2M communications

Andrius Aucinas and Jon Crowcroft
University of Cambridge
Cambridge, UK
{Firstname.Lastname}@cl.cam.ac.uk

Pan Hui
Deutsche Telekom Labs
Berlin, Germany
pan.hui@telekom.de

ABSTRACT

Energy efficient communications are extremely important in challenging environments, where access to mains power is difficult and sporadic. We propose a new approach of Machine-to-Machine (M2M) communication: augmenting the traditional model with cloud computing capabilities in the communication network to save energy and improve system availability. We propose that clones of the physical machines should be put on the network to create the Clone-to-Clone (C2C) communication and computation model. C2C has the potential to reduce traffic between end-points, reduce device power consumption and improve network performance. In the paper we present the architecture, focusing on performance improvement of a highly heterogeneous network, analyse the benefits and discuss potential drawbacks.

1. INTRODUCTION

A challenging environment is not necessarily distant or without cellular network coverage. It might simply mean that access to mains power is difficult and sporadic, requiring to save power. It could also mean that the environment is highly dynamic, with thousands of nodes potentially joining or leaving a network at any time. Or we might simply care about better node availability when connectivity is poor.

The current trends of computing are becoming inherently mobile - portable computers, smartphones, sensor and vehicular networks, etc. Furthermore, increasingly more of the systems are backed by large datacenters, such as Amazon's EC2, Google services or even Amazon's Silk browser, where browser's operations are offloaded to the cloud. LTE networks enable real-time machine-to-machine communication networks to be created [9], making cellular network based solutions more attractive.

To keep going in this direction, we need to find solutions to the limitations inherent in large-scale mobile computing: energy consumption, very limited battery power, relatively little computing resources and the load of the existing communication networks. Bi-directional communication is often made difficult by NATs and firewalls and information exchange needs to be done via central, globally accessible servers.

A straightforward example of a system we are concerned with is multiple mobile sensor systems that happen to be co-located and hence sensing and processing the same data (e.g. detecting speakers in the same room) while they could instead coordinate their efforts: total amount of data transmitted is smaller if the nodes are aware that there is somebody else already doing what they need and total amount of computation is reduced by not repeating redundant actions, reducing power consumption.

We augment the communication network with cloud computing capabilities to place virtual clones of the machines within the network, forming the Clone-to-Clone (C2C) model. In the C2C method communication between machines is replaced with communication between their clones within the network, wherever possible. It is achieved by using the clones to do all functions that do not require immediate interaction with the physical world as well as filtering and aggregating. The model uses computation offloading from the physical devices to augment their capabilities and reduce the amount of traffic between the network and the central data analysis nodes.

There are essentially only 3 basic ways of doing offloading. *Moving all or most of an application to the cloud* and accessing it using a thin client requires little CPU power on the device, but needs a lot of network resources. Dimorphic computing [5] is one such system, which dynamically switches between thin and thick client modes.

Moving most of an application to the mobile device and only synchronising occasionally, e.g. when the device is connected to mains power, is extremely light on the network, but has little effect on reducing CPU or storage requirements of a mobile device. The simplest example is the iPhone synchronising music library when connected to a computer or downloading large applications only when connected to a WiFi network. Another example is the Paranoid Android [10] smartphone vulnerability protection system, which records and replays system processes on a device clone. The synchronisation can be extremely lose, only synchronising when the device is recharging.

The third way, the one that has recently received a lot of interest, is *distributing the application into local and remote parts in a smart way*, so that offloading decision is optimally made based on network characteristics, power and energy consumption. Example of such systems include MAUI [2], which enables energy-aware offload of mobile code to in-

ExtremeCom '12, March 10-14, 2012, Zürich, Switzerland.

frastructure by comparing energy consumed by transmitting data and executing code locally, CloneCloud [1], performing thread migration between the device and it's clone. The cost in this case is more complexity in the management of these devices. Furthermore, due to relatively high energy cost of sending data, these approaches are useful for CPU-bound tasks and has limited benefits for data-intensive ones.

C2C can do all of these: it dynamically decides when it is worth to offload computation to the clones in the cloud, but has most of the application running in the cloud and works by synchronising computational state or data during execution as needed. However, to make C2C more suitable for data-intensive tasks than its predecessors, most of the data is contained in the clones and transmission to physical devices is minimised. It happens through only transmitting data that is necessary for the physical device at that time and sending higher-level information rather than raw data that has been processed within the clone network. In some sense, it could be viewed as reverse offloading: most operations happen in the clones and state is only synchronised with the mobile devices occasionally, as necessary.

Overlaying all communications onto the cloud would be a big performance hit when direct communication is possible between colocated devices, for example through ad-hoc WiFi or ZigBee networks. However, in many cases such close proximity communication is not possible, requiring complex solutions to circumvent NATs and firewalls. In the C2C network each node has a corresponding accessible virtual node, which then is able to communicate with all others directly, avoiding the problems of scaling a central server or requiring to trust an entity to store all network data. In an extreme case, for very low latency networking, clones of devices could be located on each other, rather than the cloud, so that most communication between devices never uses a radio interface. It is, however, CPU and storage intensive and energy saving provided would be very limited. Therefore, we do not analyse it further.

Importantly, the node owner gets to choose what cloud computing platform it uses to run its clones on. It means that the owner can maximize his own utility for his criteria, whether it is speed, reliability, cost or anything else. Running clones in the communication network allows using dynamic resource scaling when performing computations on behalf of the physical machines, virtually unlimited energy supply and significantly more efficient connectivity among the clones while still keeping all entities logically separate and ensuring data protection.

In this paper we take a deeper look at a few applications of a C2C network and present it's architecture, focusing on the challenging parts of making it universal for different platforms and automatic management of the network. We evaluate it using randomized traffic patterns, which shows the potential for more efficient energy consumption, especially when more application-level information can be known.

2. MOTIVATION

The core idea is depicted in Fig. 1, illustrating the main difference between a traditional cellular M2M network and the C2C network, where clones of devices reside within the communication network.

To justify the increased complexity of the communication network and the additional cost of computational resources it is necessary to first understand what benefits the system brings. Since our primary focus is on mobile devices, we also analyze the benefits most relevant to the system. Goals of our system include:

- Optimized wireless device communications - C2C is capable of changing the communication model from *many-to-many* to *one-to-one* from the device point of view. It replaces the radio many-to-many communication with communication between clones.
- Improved battery life of mobile devices - through reducing amount of communication between physical devices and the amount of computing they do.
- Increased computational power of participating devices - through offloading some of the computations to the network.
- Simpler overall system management - through abstracting away from the physical platforms via virtualization.
- Improved privacy control - through decentralized data storage and management.
- Improved system reliability - through fault tolerance within the C2C network.

To demonstrate in what cases and how it is possible to achieve the above goals, we present two scenarios in detail. We believe that these scenarios are representative of the use cases of M2M networks and therefore shows the potential of our C2C architecture well.

Intelligent Transport System. The scenario includes real-time traffic management and possibly vehicle collision detection and avoidance. It is based on devices equipped with sensors embedded in cars and surrounding environment and used in traffic light scheduling or automatic driving systems. The M2M devices (cars, highway cameras, traffic lights, inductive-loop detectors, etc.) communicate with each other to track traffic information. The receiving devices then take actions based on the collected data. For example, collision avoidance system can take control of the car to activate brakes, change driving direction, alert passengers, etc. Intelligent traffic lights, on the other hand, could adjust their cycles based on traffic flows in the entire city or switch to green light for approaching emergency vehicles or for individual cars when there are no other cars (e.g. in suburbs at night) to help save fuel.

Let's take the traffic light scheduling as a sub-system and understand it's requirements and characteristics. Traffic light scheduling is important both for congestion control and for improving vehicle fuel efficiency [4]. Making local decisions based only on an estimated number of cars going in a particular direction might lead to sub-optimal schedule due to possible congestion further on a route. Furthermore, adjusting schedule for different types of approaching traffic (higher priority for public transport and top priority for emergency vehicles) requires vehicle identification using radio communication. Communication between cars and traffic light schedulers as well as other cars can also help drivers and in-car systems[1] to be more fuel-efficient by helping to predict traffic flow in the near future.

The important properties of this scenario are:

- Bi-directional communication is needed between various entities to get updates and push their own information.
- Information from a single node has little meaning without others - nodes would therefore collect and process relevant

[1]For example, BMW's Start-Stop function, which stops engine when it's idle

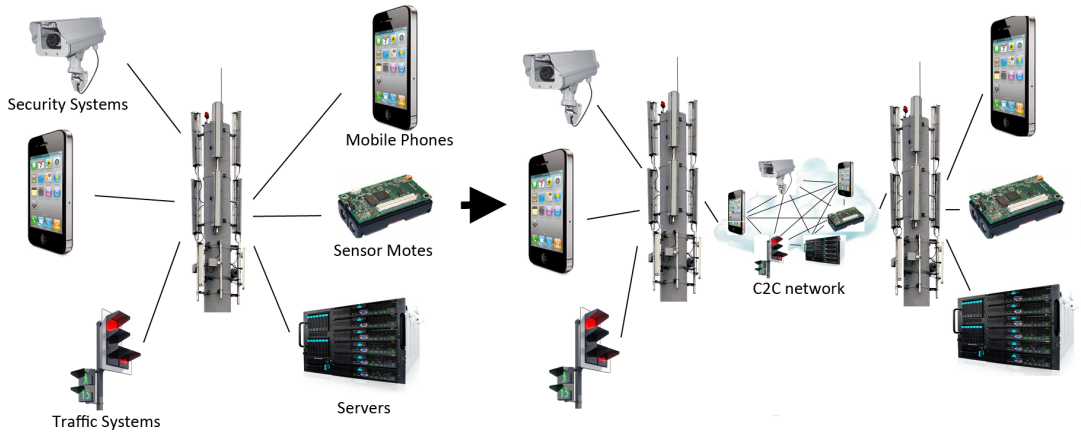


Figure 1: Difference between a cellular M2M and a C2C network

data in the cloud, disseminating the results to listeners from there.

- There is a large volume of communications. C2C helps to reduce them through the overlay network as well as by processing as much as possible within the network.
- Data obtained from vehicles (position, destination, driving habits) is important for the overall performance, but also highly privacy-sensitive - only a small, context-related subset of it would be exchanged at any point.
- Traffic load is highly dynamic throughout the day, making dynamic, commodity-like priced resource scaling a desirable feature.
- Environment is highly heterogeneous, with devices ranging from embedded controllers in traffic lights to almost full-fledged in-car computers.
- There is a large number of institutions and individuals managing a number of nodes in the network, willing to have control over them and the data collected. It makes C2C a much more feasible alternative to the centralized services.

The C2C platform therefore provides scalable computation and communication resources, serves as a set of private data containers for privacy-sensitive data and permits low-latency communication between nodes using different traffic priorities. Although M2M communications over a cellular network increase the network's load, handling large proportion of data within the C2C network reduces the effect. Integrating C2C closely with the cellular network by putting computing resources close to end-users [12] could allow to further mediate transmissions according to local network loads.

It may seem like the scenario needs real-time network and overlaying communications to the cloud unnecessarily adds latency. Nevertheless, direct radio communication even over moderately large distance is impractical. It requires multi-hop transmission or a fixed infrastructure, adding to the overall latency anyway. Furthermore, as we show later, communication latency within the cloud can be made very low compared to the other components. Although latency could be reduced by placing computing resources close to users, extremely low latency is not always necessary. Assuming that a car moves on average at 10km/h in heavy traffic (optimistic) and that we want an update from it every 10 meters, we need a signal every 3.6 seconds. Even if we consider a car moving at 50km/h, it would probably need to signal a traffic light from a 100 meters distance (unlikely there are other cars in such fast urban traffic), leaving 7.2 seconds before it reaches it. Both of these are easily achievable in today's Internet, hence the extra latency does not cause problems.

This scenario illustrates usefulness of a C2C network very well, as it is easy to change it to e.g. *Smart Home*, doing tasks such as helping with housekeeping (turning on a robotic vacuum cleaner when there is nobody in the house, etc.) or aiding disabled people - detecting if a person has fallen down and informing a care institution or taking snapshots of the daily life of a person suffering from Alzheimer's for later review as well as tracking his location in case she gets lost. There is always a large number of sensors and actuators tightly interconnected. Mobile phones or notebooks can further integrate with the environment by using it as a storage vault for daily events, or connecting to the home entertainment system.

Mobile Mutiplayer Online games are highly interactive and require crisp response from both application server and other players, otherwise leading to glitches and frustration. A broad genre of such games is MMORPG[2]. We will take *virtual bike race* as an example - race using real bicycles, where the opponents are possibly on different locations and race along corresponding tracks and sensor data is exchanged to track each other's progress.

Characteristics of this scenario include:

- Very low communication latency between players.
- Possibly large number of players, requiring scalable processing and communication.
- Limited battery power of mobile devices - sensor data processing, graphics and communications drain the battery.
- Traffic pattern is periodic and relatively high frequency, depending on the environment, number of players and their interaction patterns.
- It is important not to miss users' actions and to recover from glitches quickly to avoid state inconsistencies.
- Privacy can be important - in a pseudonym-based game it might be less crucial, but e.g. disclosing physical location of users causes more problems.

Latency can be an issue in this case and we are currently looking into ways of reducing it. Nevertheless, users in geographically distant locations will not feel a significant difference because the C2C network itself adds very little latency,

[2]Massively multiplayer online role-playing games

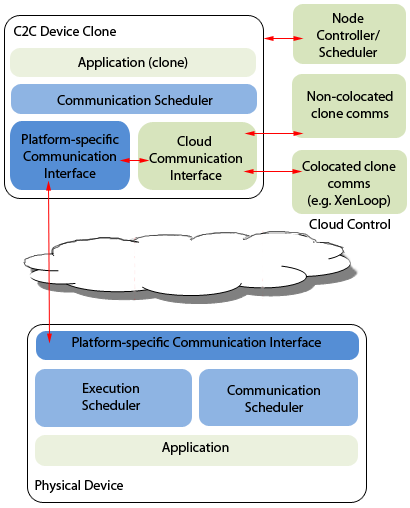


Figure 2: The C2C architecture - physical device, its clone and the cloud management entities

as we show later.

Like above, when there is a large number of simultaneous players, communication scalability issues arise: sending every GPS update to each of the other participants (tens or hundreds for large games) needs a lot of bandwidth and low delays between each participant. With C2C, instead of a participant sending the data to all other nodes, they would just send it to the virtual clone which would overlay it onto the virtual track and send it to other virtual nodes, making the radio transmission more energy-efficient and updates faster to propagate to all nodes.

Many games have another important characteristic: players' actions require computationally expensive computations. It includes complex AI engines (non-human players), computer graphics, physics engines, changing world state, etc. Therefore, it may be beneficial to use C2C computational capacity to do some of these operations to vastly improve user experience.

3. SYSTEM ARCHITECTURE

To accomplish the outlined goals it is necessary to create an efficient architecture for the C2C model. Below we present the core components of such network, making clear how C2C approach is superior to traditional M2M networks. The architecture is depicted in Fig. 2, which we describe in this section: we discuss where the C2C network is actually located, how we deal with heterogeneity of general M2M networks and how we improve their efficiency.

3.1 Building network

Network is crucial for an M2M application and its performance can often be the limiting factor of what is achievable. Communication network in C2C has two parts: the cellular data network and the inter-cloud network. The former is responsible for communication between the physical devices and the C2C cloud and the latter - for providing the direct clone-to-clone communication.

The main mechanism of improving mobile devices' battery life and improving network's scalability is through virtualizing them on a cloud computing facility. Since we rely on improving this part of the M2M communication path, it is crucial to make the cloud network efficient - latency has

Data path	Latency (μs)	Bandwidth (Mbps)
Inter Machine	101	941
Intra Machine (Xen Netfront/Netback)	140	2656
Intra Machine (XenLoop)	28	4143
Inter Device (4G)[3]	20000	300

Table 1: Average latency and bandwidth comparison

significant effect on applications, but can be reduced [11]. The clones form the C2C network with vastly higher computational capabilities, higher bandwidth and significantly lower achievable latencies, especially when using advanced techniques such as XenLoop (Table 1) and a thinner software stack. For this reason we have the Node Controller and separate out communication of colocated and non-colocated clone VMs into separate cases (Fig. 2).

Seeing that inter-VM communication overheads are almost negligible compared to cellular network latencies, it is obvious how C2C can improve performance of *many-to-many* communication model of M2M even when all messages are delivered immediately to recipient physical nodes. Instead of each node talking to every other node it wants to send data to, it just sends it to its virtual counterpart, data is distributed in the virtual network and the recipient virtual nodes forward the data to recipient physical nodes. From the physical nodes' point of view and network latencies it essentially becomes *one-to-one* communication with two RTTs (one to send the message to the cloud and the second for it to forward it to the relevant physical node) and amount of data for one message only.

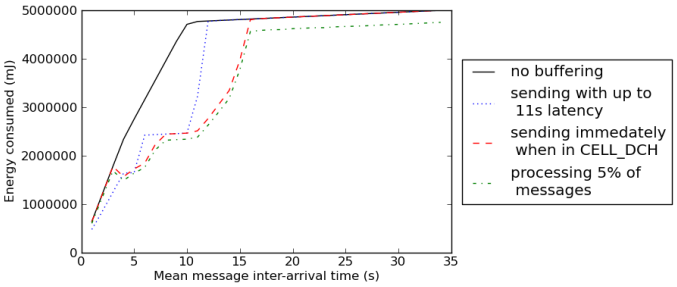
Finally, C2C architecture automatically allows for better data traffic control. Since all communications in C2C have associated tolerance to delays and delivery requirements, it is possible to schedule traffic in the cellular network to reduce peak load by smoothening traffic and not hurt performance at the same time by providing extra information to the communication scheduler (Fig. 2) of a clone. As it has been pointed out, networks are often suffering from the signaling traffic more than the actual bandwidth problems and it may soon become the main bottleneck [6]. Integrating C2C closely with the cellular network is a difficult problem, and we do not analyze it further in this work.

3.2 Managing heterogeneity

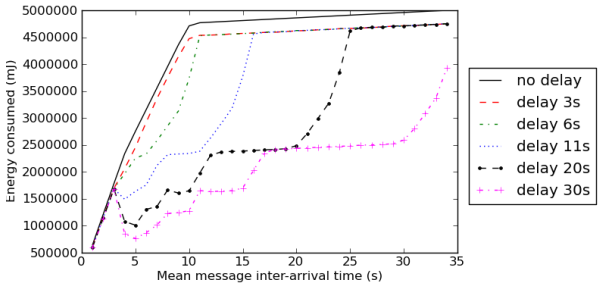
In M2M networks it is often desirable to have sensors at one end of the communication channel, transmitting any recorded data to the other end (telemetry). This, however, can turn out to be rather wasteful in terms of bandwidth consumed and not at all necessary. An important observation here is that very often it is not the low-level sensor data that is required, but higher-level information extracted from the raw data. Processing the data, however, costs resources - CPU power, energy, etc. and it may be worthwhile to offload such computations from the devices [3], even though radio transmission also uses a significant amount of data.

Many of the devices might be connected to an M2M network using proprietary protocols, or have very different interfaces. This complicates matters. In a traditional network it would be necessary to either modify software of the devices or have some centralized service for data exchange that is able to deal with the heterogeneity of the network.

C2C approach, allows to provide homogeneous interfaces between all devices by abstracting the physical nodes through



(a) Energy consumption when doing message aggregation and possibly processing part of them, using different methods



(b) Energy consumption using the same method but different delay bounds

Figure 3: Energy consumption of device-to-clone communication

virtualization: since the clone of a device is only loosely equivalent to the device itself (i.e. capable of locally performing the same functions as the physical one, as long as they do not require interaction with the physical world) it can have one interface for communicating with the physical device, and another, standardized, to communicate with the rest of the network, as depicted in Fig. 2 by the two communication interfaces in a clone. This characteristic requires our clones to be only loosely related to the physical devices. Nevertheless, they remain equivalent in terms of functions they perform, only the communication interfaces differ.

Current commercial virtualization solutions such as Amazon's "Elastic Computing" are not suitable in this environment either as the software running on many of the devices that would use M2M communication are neither running Linux, nor Windows and the standard software stack is simply too thick for virtualizing embedded systems. Furthermore, virtualization normally encapsulates an entire operating system and emulates it in a virtual environment, introducing yet another layer to the already bloated modern software stack, leading to an efficiency disaster [7].

We therefore turn to more specialized architectures, such as Mirage [8]. The key principle behind Mirage is to treat cloud virtual hardware as a compiler target, and convert high-level language source code directly into kernels that run on it. Such lightweight architecture gives many advantages - ability to instantiate and destroy Virtual Machines with low latency, reduce overheads and significantly reduce image sizes (e.g. the authors report 600KB image size for the Mirage images used for testing, compared to Linux distributions that are difficult to squeeze below 16MB).

Virtualization by itself also helps with the other goals of the system. It helps to manage the whole network by abstracting away from the hardware the virtual machines are running on and allowing dynamic relocation in the case of failures or when vertical scaling of a single clone is required. In addition to that, provided that the cloud hosting environment itself is considered trusted, virtualization provides isolation between VMs and therefore offers better security and data privacy guarantees.

Overall, with recent advances in cloud computing software, we see great opportunities in using the cloud efficiently for really scalable M2M communication, connecting any type of device together.

3.3 Managing C2C network

There is a number of interesting aspects of managing the C2C network that we still have not discussed. These include scheduling operations between physical devices and their clones and controlling communications. The main components supporting the network on the physical device are as shown in Fig. 2:

- Execution scheduler - deciding whether operations should be executed locally or on the cloud.
- Communication scheduler - filtering, aggregating and delivering messages between devices and clones.

As all the recent works related to mobile code offloading have shown, not all computations are worth performing remotely, since communication costs energy and causes delays. Therefore, for C2C applications it is crucial to devise efficient mechanisms to make the decisions of when it is worthwhile to offload certain operations. Multiple works have already addressed the task [2, 1], however there is a big difference between them and C2C: instead of sending all related data and computation requests and receiving results for every operation considered for offloading, we avoid transferring the control back to the physical device once offloading is done.

As part of our effort, we have implemented ThinkAir [3]. The system currently does code offloading code from mobile Android devices to the cloud. It chooses which pieces of program code should be executed remotely, optimizing execution time and energy consumption. It deals with resource allocation and parallel code execution on the cloud, and does so by dynamically creating and scaling clones of mobile devices that execute the offloaded code. However, extending it further to allow communication between these devices gives combined benefits of faster computations, optimized communications and distributed knowledge in the network.

Clones do not only act as servers for the devices operating in thin client mode, however, but rather are part of the communication network, therefore receiving data and requests from the other nodes and adjusting their behaviour based on them. Furthermore, as we have pointed out earlier, not every message sent between two machines necessarily has to reach the other end-point as soon as possible and dealing with them in the clones (aggregating received data, performing requests whenever physical environment is not involved, bundling messages for later delivery) saves traffic between the physical devices and allows to save energy and improve performance.

Tasks such as message aggregation and filtering should be done at application layer, as their timing guarantees are

application-specific and rules for data aggregation are specific to an application as well as current working conditions. For example, as shown in the scenarios above, most messages under normal network working conditions are simply periodic messages with minimal need to be delivered immediately, however there are cases when they must be forwarded to the end-point immediately, with minimal interaction with the C2C network to minimise overheads (e.g. warnings in collision-avoidance systems).

We demonstrate the potential benefits of message filtering/aggregation in figure 3. In the tests, we sent 1000 1KB messages, varying their average transmission interval from 1s to 35s (from Poisson distribution) and used energy consumption model based on PowerTutor [14]. We compared 4 ways of sending messages: a) sending them as they are generated, without b) delaying the first message in the queue for its maximum delay, while accumulating others in the meanwhile and sending the whole batch together, emptying the queue c) same as b, however if a message arrives into an empty queue while the radio is still in high power state (CELL_DCH), send immediately d) same as c, but randomly choosing a percentage of messages to be filtered/processed in the clone and therefore not sent at all.

Simply delaying message transmission is neither sufficient, neither is our goal. Application layer information is necessary for good results, but even in the simple experiment we can see huge gains: for average message inter-arrival time of 5s and up to 5s delay when aggregating messages or processing data into higher-level information, the savings are around 22.2% and if we increase the maximum delay to 11s it increases to 41.5% and goes as high as 47.3% if message inter-arrival time increases. C2C does not commit to always using higher delay traffic for an application though - application itself can decide whether it needs near real-time action or needs to focus more on energy savings.

In the case of a large clone network performance and energy savings are even more obvious: when no aggregation is done, instead of having total transmission time proportional to the number of nodes (i.e. at least n RTTs for n nodes) it only takes 2 RTTs (to reach the C2C network and to send message from a clone to the recipient). For example, if the data transmitted is sensor readings, further gains are achieved in C2C by processing the data in the clone and only sending higher level information (e.g. text instead of voice recording), further saving bandwidth and overall computing power required.

A difficulty is, with current virtualization technology timing guarantees cannot be meaningfully provided for interactions between co-located VMs, as the hypervisor (e.g. Xen) lacks knowledge of timing requirements of applications within each VM [13]. The problem can be partially solved by reducing latency of communication between co-located VMs, but to assure timely execution for real-time processes within a VM, the hypervisor's CPU scheduler needs to communicate with the VM's CPU scheduler to communicate the real-time requirements. We have yet to address this issue in future work.

In summary, communication scheduling as well as reducing the amount of it between the physical devices therefore plays an important role in improving performance of the network and reducing the devices' power consumption, but has more potential when combined with other performance-improving techniques, such as code offloading.

4. CONCLUSION

In conclusion, C2C is a new approach of building more powerful and efficient M2M communication networks for a wide range of applications that can be adapted similarly to the ones presented. It aims at saving mobile device battery power through reducing their radio traffic, but without sacrificing system's responsiveness. Furthermore, it allows to use more powerful end terminals without increasing power consumption or using more sophisticated (and expensive) hardware.

C2C is not a way to replace telemetry applications. Instead, we feel that M2M communication is much more powerful than that, and C2C is an approach helping to solve the issues hindering development of the next-generation networked applications.

5. REFERENCES

[1] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In *EuroSys*, pages 301–314, 2011.

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. Thinkair: Dynamic resource allocation and parallel execution in cloud for mobile code offloading. In *INFOCOM, 2012 Proceedings IEEE*, march 2012.

[4] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. MobiSys '11, pages 127–140, New York, NY, USA, 2011. ACM.

[5] H. A. Lagar-Cavilla, N. Tolia, R. Balan, E. de Lara, M. Satyanarayanan, and et al. Dimorphic computing. Technical report, 2006.

[6] H. A. Lagar-Cavilla, N. Tolia, R. Balan, E. de Lara, M. Satyanarayanan, and et al. Smartphones and a 3g network. Technical report, Signals Research Group, May 2010.

[7] A. Madhavapeddy, R. Mortier, J. Crowcroft, and S. Hand. Multiscale not multicore: efficient heterogeneous cloud computing. ACM-BCS '10, pages 6:1–6:12, Swinton, UK, UK, 2010. British Computer Society.

[8] A. Madhavapeddy, R. Mortier, R. Sohan, T. Gazagnaire, S. H, T. Deegan, D. Mcauley, and J. Crowcroft. Turning down the lamp: Software specialisation for the cloud.

[9] N. Nikaein and S. Krea. Latency for real-time machine-to-machine communication in lte-based system architecture. *Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless), 11th European*, pages 1 –6, april 2011.

[10] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos. Paranoid android: versatile protection for smartphones. ACSAC '10, pages 347–356, New York, NY, USA, 2010. ACM.

[11] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout. It's time for low latency. HotOS'13, pages 11–11, Berkeley, CA, USA, 2011. USENIX Association.

[12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14 –23, oct.-dec. 2009.

[13] J. Wang. Survey of state-of-the-art in inter-vm communication mechanisms, 2009.

[14] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. CODES/ISSS '10, pages 105–114, New York, NY, USA, 2010. ACM.