# Mobile Distributed Processing of Physiological Data

James Meneghello
School of Information Technology
Murdoch University
Perth, Western Australia
Email: J.Meneghello@murdoch.edu.au

Kevin Lee
School of Information Technology
Murdoch University
Perth, Western Australia 6150
Email: Kevin.Lee@murdoch.edu.au

Kiel Gilleade
School of Natural Sciences and Psychology
Liverpool John Moores University
Liverpool, UK
Email: K.M.Gilleade@ljmu.ac.uk

***Abstract*—Physiological monitoring can be useful in a number of scenarios to evaluate or diagnose the status of individuals or groups, for health or mental reasons. The devices used to collect this data have become increasingly portable, but deriving useful metrics from such data can often take significant processing - a commodity not always available in mobile environments. This paper presents and evaluates a system designed to easily process physiological signals in mobile environments by utilising commonly-available smartphone hardware to collaboratively transform collected data.**

## I. Introduction

Physiological monitoring is the practice of using sensors to read, store, process and interpret physiological data from organic beings, including biofeedback signals associated with heart, brain, muscle and other organ activity. Physiological monitoring can provide a plethora of useful health, fitness and other related data in real-time. Useful measures such as pulse, respiration rate and blood oxygen levels have been successfully used to diagnose and aid in the treatment of conditions such as sleep apnoea [1], as well as assisting in the monitoring of cardiac systems [2].

Some physiological monitoring devices collect data in a form that is inherently useful without any processing or analysis performed upon it, such as heart-rate or blood pressure data. However, there are a number of data points that can only be extracted through extensive processing. For example, respiration rate can be obtained through complex processing of a number of physiological signals. This kind of complex processing on a number of data streams in real-time can present problems to current physiological processing methods, as mobile environments often lack processing resources.

Complex physiological monitoring in real-time can already be performed using fixed processing resources such as servers or cloud platforms. However, a number of environments exist in which transmission to these resources is not viable. Poor or non-existent broadband communications coverage in remote areas effectively nullifies the use of remote processing to perform data transformations. In addition, traditional processing resources (such as workstations or servers) are not portable and require uninterrupted power supplies, which are generally not available in remote or highly mobile environments.

In order to provide an suitable source of processing resources for these environments, existing mobile devices (such as smartphones or tablets) can be used to process desired data transformations. Unfortunately, while the processing ability of mobile CPUs has increased dramatically over recent years, power drain and battery capacity remain bottlenecks. In order to extend the monitoring lifetime of these mobile devices, any data analysis required can be processed in a collaborative fashion to balance resource drain across contributing devices - preventing any single device from taking too much of the work and depleting itself. Mobile devices have the advantage of not requiring an uninterrupted power source, allowing them to be used as processors while running off battery power - and be charged using a portable generator or solar cells.

A system designed to facilitate the distributed processing of physiological data can be evaluated through a comparison to existing methods of mobile processing - namely, a single device handling all processing required. Successful scaling of processing across multiple mobile devices would allow for more complex processing to be performed upon physiological data, allowing for new opportunities in medical monitoring and evaluation in remote environments.

The remainder of this paper is structured as follows: Section II provides a background of physiological monitoring and justification for the work. Section III presents the design of a system for performing processing of physiological data on Android smartphones and PC hardware. Section IV evaluates the effectiveness and viability of using the system to monitor and process data in real-time. Finally, Section V concludes.

## II. Physiological Processing

Physiological monitoring generally consists of a few components; a monitoring device used to collect data, a processing resource used to transform data into useful information (if necessary), and a transmission device used to communicate the results to a centralised source for analysis by experts (such as a doctor or medical technician). Currently, situations that require complex data analysis on physiological signals utilise the communications link to transmit data to fixed-location processing resources, such as a server or Cloud computing cluster. In remote and mobile environments, this is often not an option as there are many places lacking in communications, electrical and processing infrastructure. By reducing the reliance of monitoring systems on external resources, physiological monitoring can be extended into remote communities that stand to benefit from increased medical responsiveness.

978-1-4673-4723-5/12/$31.00 

Initially, monitoring systems were largely confined to clinics and hospitals due to size and low mobility, but advancements in mobile monitoring systems [3]–[5] have provided greater flexibility in usage environments. By utilising readily-available sensor devices with low power requirements, long-term monitoring can enhance the quality of care for active patients, as well as providing numerous advances to research associating physiological markers with mental states, such as the measurement and analysis of stress levels [6].

As well as monitoring individuals, it is possible to use distributed physiological sensors to monitor groups. By analysing and aggregating physiological data from the group, conclusions can be drawn regarding the state of the entire group. Group monitoring has a range of possible usage scenarios, such as monitoring the life-signs of miners in underground tunnels in order to hasten awareness of emergency situations [7], or monitoring a population for symptoms of influenza to locate outbreaks [8]. Applications for this technique can also be found in monitoring psychophysiological status of participants in training exercises [9], and determination of traffic flow within city areas [10] for civic planning purposes. This wide range of uses could be further extended if devices were made more mobile and available, through the use of commodity hardware for implementation purposes.

Relatively simple sensors, such as pedometers or heart rate monitors, are available to monitor various high-level physiological data points for use by consumers, often integrating with exercise trackers or social networks. Because the data is generally collected in its final desired form, little transformation is required. For many users, simple data collection and transmission provides a satisfactory view of the desired result.

An example is illustrated in Figure 1, which represents a typical use case for a casual consumer using basic physiological monitoring devices. A Bluetooth-enabled heart rate monitor is worn during exercise, which transmits pulse rate (HR) data to the user's smartphone or logs it for later collection. This data is then transformed using a simple mathematical algorithm into R-R interval, a representation of peak distance between heartbeat waves. Both resulting variables can then be uploaded to activity trackers or social media sites, allowing users to compare exercise habits or encourage other activity.

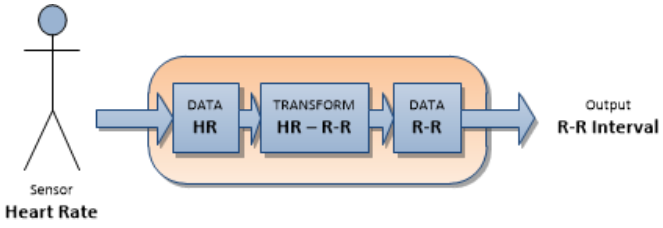


Fig. 1. Simple transformation from Heart Rate to R-R Interval

Other applications, particularly involving cardiac-related monitoring [2] or psychophysiological analysis [6], require more complex data transformations. A physiological transformation is the process of taking raw physiological measurements and applying algorithms that can extract other useful information, which is in turn used for activities such as disease diagnosis and measuring physical and mental stress levels.

In addition to simple physiological measures such as pulse and blood pressure, the human body can be monitored for electrical signals produced by specific parts of the body. In a healthy human heart, the sinoatrial node produces a bioelectric pulse designed to contract heart muscles and produce the standard human heart rate of 72 beats per minute. Electrical activity like this can be measured using surface electrodes, with position of electrode determining the object being monitored. This technique of monitoring the heart through collection of surface electrical activity is called electrocardiography [11], and is one of a number of electrical signals used in biofeedback and physiological monitoring.

There are many electrical signals that can be collected from the human body for purposes of disease diagnosis and health analysis. Electrocardiographs (ECGs) [11] are commonly used to assess and diagnose cardiac problems, but can also be analysed to produce other data such as pulse and respiratory rate [12]. Photoplethysmographs [13] are a signal produced most commonly by pulse oximeters, which measure blood oxygen saturation and are used for cardiac-related analysis and respiration detection [14].

Transforming signal data such as electrocardiography or photoplethysmography into useful metrics can involve advanced signal processing algorithms [15], which can require significant computing time to complete. Additionally, even for routine transformations it can be a multi-step process. To ensure result accuracy, data is often filtered and normalised prior to signal processing, which further adds to computation time.

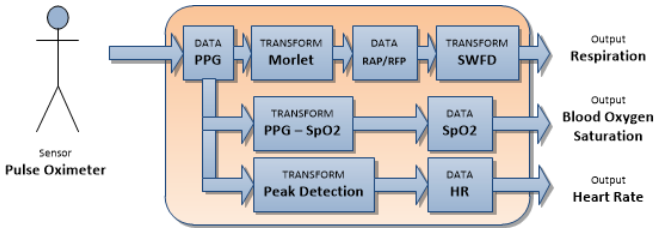


Fig. 2. Complex transformation chain for extracting several data points from a PPG

Figure 2 illustrates the derivation of several useful points of data from a photoplethysmograph (PPG), using a pulse oximeter attached to a subject. Obtaining the heart rate (HR) is a simple time-domain peak detection operation [16] and blood oxygen saturation (SpO2) is a simple calibration function applied to the pulse oximeter's output connections. By comparison, determining respiratory rate from a PPG is a computationally expensive process [14], requiring multiple wavelet transforms [17]. These types of signal processing algorithms take significant processing time to complete (depending on size of dataset), making it impractical to provide real-time results without dedicated computational resources.

While a simple transformation (such as Figure 1) could be handled by even a small embedded processor, continuous or

concurrent complex transformations require significant computational resources to complete in a timely fashion. Depending on scenario and environment, it is not always realistic to carry static processing resources, particularly if part of a highly-mobile group. To avoid the unwieldy scenario of transporting powerful workstations to process collected data, a cluster of heterogeneous mobile devices can instead be used to collaboratively process work. In the event that further resources are needed, on-demand Cloud or on-site resources can be utilised, at a price.

In order to alleviate the strain on external resources for this method of analysis, the devices collecting and transmitting the physiological data should be able to handle some or all of the processing occurring. While mobile devices have previously been used for similar purposes [18], it is noted that such devices are often resource-limited: particularly considering low battery capacities. For the system to cope with devices with different levels of ability or power resources processing should be distributed across all devices such that their weighted load provides a more efficient use of group resources [19]. This would allow the system to monitor groups for longer and with greater responsiveness than otherwise possible, which is an important consideration in situations involving health-critical monitoring subjects such as cardiac rehabilitation patients.

Enhancing the mobility of physiological processing systems could provide benefits in a wide range of areas. By tracking physiological data of groups of soldiers, remote assistance can be provided to assess casualties, as well as provide early notification of potential medical emergencies. In addition, it can be used for evaluation purposes - by establishing baseline heart-rate variability markers for participants, stress levels during training in mobile environments can be assessed [20]. This provides the potential for early discovery of stress-related problems in soldiers, which can have significant effects, both while participating in battle [21] and upon returning home [22].

Similarly, miners could be equipped with monitoring devices and small processing units could be built into existing safety equipment. This would allow the system to include other types of sensors looking for issues with air quality or poisonous gasses, able to be entirely processed by the mobile units. By collating data from these units, it would be possible to locate areas with poor habitability and alert workers to avoid certain areas. While mine workers are likely to be able to carry larger battery reserves for processing than individuals in other scenarios, efficient distribution of work processing would allow for more accurate monitoring, as more available processing could cater for lower intervals between sensor readings.

Available communications are an important part of most systems requiring significant amounts of processing resources, as it allows the system to offload excess tasks to more capable processing facilities, such as Cloud computing. Environments which are naturally unsuitable to mobile broadband communications networks such as heavy forest, mountainous or hilly terrain and areas suffering from heavy snowfall are difficult to operate these kinds of systems in, as communications are often disrupted or completely unavailable. Without the ability to offload some or all of the processing to these external resources via those mobile networks, the system may have to severely reduce functionality in order to remain even slightly useful - generally an undesirable scenario.

The issues highlighted in this section provide insight into some of the areas of physiological monitoring and processing that could be improved. Shifting to a more mobile paradigm would abate or entirely solve some of the issues. Mobile devices operate on portable power supplies, removing the requirement for traditional processing resources to be connected to fixed power supplies, and introducing the option of recharging mobile batteries through use of a portable generator or solar cells. The devices are also less heavily reliant on the presence of reliable mobile broadband communications networks, as they should be able to process all requisite physiological transformations in a collaborative fashion, without having to offload processing to fixed or Cloud computing resources. The devices are also completely portable and can be used while mobile, allowing the use of physiological monitoring and processing in a wider range of situations. Finally, mobile devices are almost ubiquitous, making implementation accessible due to a lack of special hardware requirements.

## III. Mobile Distributed Physiological Processing System

To provide a platform for distributed physiological monitoring and transformation processing, careful consideration is required while designing an architecture to suit potential usage environments, such as those described in Section II. In order to ensure compatibility with existing devices and interoperability with existing physiological monitoring systems, standard interfaces are designed through which data access and transformation requests may be made. Specific roles within the system are defined, with responsibilities allocated in such a way to provide a scalable architecture that is easily configurable in mobile environments.

### A. Architecture

The proposed system is a generic architecture for processing real-time physiological transformations that is designed to manage the packaging and distribution of processing to other devices. Any available device can be utilised for processing, including nearby smartphones, other mobile devices such as tablet computers, workstations or laptops and Cloud Computing resources.

In order to effectively manage groups of sensor nodes, a hybrid peer-to-peer network architecture is used to group monitoring devices for data collection and processing. Positioned in the center of all monitored networks, a supervisor is manually selected, based on a number of metrics such as remaining power resources, processing ability and strength of connection to both other nodes and upstream. The supervisor maintains connections to managers, receives data from monitoring devices and provides an interface for clients to

interact with. If stationary, high-powered devices are available (such as wireless base-stations), these static supervisors can more effectively replace the role of a manager in the system. A simple topological example is presented in Figure 3. In this example, the laptop takes the place of the supervisor and manager, while the smartphones act as both monitors and processors.


Fig. 3. Device topology in a mobile simple usage scenario

Individual processors perform tasks as directed by the manager, which would often be their own processing. If power or processing resources on other processors are depleted, the manager would re-allocate work to other nodes with abundant resources, and processing can continue unimpeded. If all processors have depleted resources, processing is restricted to collection and transmission - the manager would then send all results upstream to Cloud computing resources for processing. The benefits and disadvantages of processing at each of these layers is modelled in Figure 4. Importantly, it is noted that latency and difficulty of device access increases dramatically as processing is shifted further away from individual nodes, as fast and reliable network connections can be difficult to procure in some usage scenarios.

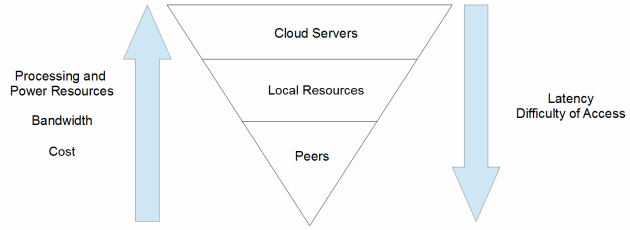


Fig. 4. Benefits and Disadvantages of Processing Layers

To allow the manager to balance load evenly across the network, nodes are also required to report their current state to the supervisor. This includes details such as current load, remaining power resources, current processing details and others. Because the stream registration system already exists for data collection and transmission purposes between client and server, the existing stream methods can be repurposed to report device state alongside any sensor data.

The manager balances load and resources by monitoring the state of clients, allowing it to distribute work to low-load devices with plentiful resources. The supervisor also keeps aggregate state details, such as an average power resource level of the group. In turn, the supervisor provides the manager with supplementary data, allowing it to decide if the network is unable to cope with current workload for a sustained period of time. In this event, the manager would trigger offloading of workload to Cloud Computing resources (if accessible).

As mentioned previously, the proposed system categorises resources into four roles; processors, monitors, managers and supervisors. Any resource can be used in any of these roles, though suggestions for optimal selection are noted below, where each role is described in detail.

*1) Supervisor:* The supervisor is the overarching controller of the entire system. Its primary role is as an interface to the system for all external entities, including monitors, managers and clients. The supervisor hosts the RESTful interface to the system, as well as collating all data received from monitoring nodes, passing transformation requests to the manager (along with required data streams) and tracks node state for load balancing purposes. As the supervisor has to deal with a substantial amount of incoming requests, it is recommended that a device with good processing ability and network throughput is utilised in this role.

*2) Manager:* The manager only directly interacts with the supervisor and processing nodes, and acts as a load balancer for all processing resources. It retrieves node state from the supervisor and uses either utility functions or a decision tree to determine where transformation tasks should be sent for processing. For smaller networks, the manager can share resources with the supervisor. However, due to the considerable workload assigned to the manager, a dedicated device is recommended for use in larger monitoring networks.

*3) Monitor:* A monitoring device has a single purpose; to receive raw sensor data and package it in a form suitable for transmission to the supervisor. The configuration of the monitor should be flexible enough to enable it to interact with any sensor that could be attached to the device. In most networks, these will be low-powered devices. If the device is required to poll multiple sensors for data with high responsiveness, the multi-threaded nature of this task will require substantially more powerful hardware.

*4) Processor:* Processing resources are the workhorse of the network. The manager distributes transformations to the processors to be completed in as little time as possible, but the nature of the networks allows significant scaling. In portable environments, even low-end smartphones can act as capable processors. If heavy transformational work is required, dedicated processing servers or even Cloud computing resources can be utilised in this role.

Each role must be fulfilled in some manner for the network to function properly, but in many instances a single device will fill many roles. As an example, most processors will also be monitors in order to report their state to the supervisor. In smaller networks, the supervisor and manager may share

resources. However, when monitoring large sensor networks each role can be filled by appropriate resources to ensure optimal network reliability and responsive transformation processing.

### B. Interface

Connections to the physiological monitoring and processing system will be made to servers through a RESTful [23] web API using XML, allowing connections by any software that implements the correct interface. Monitoring devices also register users, streams and transmit data using this API, which then serves data and transformation requests by external clients. Using a RESTful interface provides an accessible and compatible connection interface to physiological data and transformations, supporting a broad range of applications and existing sensor products. Presenting a standardised interface to users assists in the integration of existing systems, as well as the development of analysis suites. To ensure the interface is as accessible as possible, the XML-based communications can be utilised over HTTP and HTTPS connections.

While user and monitor data stream registration and authentication is reasonably simple, some XML messages can be relatively complex, particularly those concerning chained physiological transformation requests. A "chained" transformation is effectively a series of physiological analysis functions performed on a set of data, such as that depicted in Figure 2. Listing III-B depicts a sample XML message passed via HTTP POST to the supervisor. The message defines a two-stage transformation that processes one set of data based on an existing stream, then passes the results along with another stream to a second transformation.

```
<?xml version="1.0"?>
<Transformation>
        <Transform>
                <Name>ECG2IBI</Name>
                <Input>
                        <StreamID>1</StreamID>
                        <Format>ECG</Format>
                </Input>
                <Output>
                        <Name>someIBI</Name>
                        <Report>True</Report>
                </Output>
        </Transform>
        <Transform>
                <Name>MergeData</Name>
                <Input>
                        <StreamID>2</StreamID>
                        <Format>PPG</Format>
                </Input>
                <Input>
                        <Name>exampleIBI</Name>
                        <Format>IBI</Format>
                </Input>
                <Output>
                        <Name>SPC</Name>
                        <Report>True</Report>
                </Output>
        </Transform>
</Transformation>
```

Listing 1: Complex XML Transform Request

Upon submission of a correctly-created XML document conforming to the above structure, the supervisor will schedule work to be completed across any available processing resources. Transformations may not necessarily be completed on a single processor (as each part may be executed separately), so the complexity of the transformation may determine how long the data will take to return. The server conforms to standard RESTful HTTP rules, correctly returning HTTP status codes to signify success or failure.

### C. Implementation

To ensure ease of use and cross-platform compatibility, the system is developed as a series of applications, divided by role. The resulting software should execute on any cloud platform (such as Amazon EC2 [24] or Google App Engine [25]), PC-compatible system or Android [26] smart-phone, allowing a broad range of devices to be deployed to support the operation of the system.

All roles in the system are implemented as a standalone Python [27] scripts. The system takes advantage of the XML message syntax specified previously to serve user requests and stores monitoring data using a database engine. For configurability and accuracy's sake, the Mako [28] templating engine is used to generate XML messages.

*1) Supervisor:* Data received from monitoring agents (as well as the supervisor's own internal metadata) can be stored using any storage engine supported by SQLAlchemy [29]. SQLAlchemy provides a layer of abstraction above the database, and allows users to easily change the storage engine to suit. While the default SQLite [30] file-based database engine can provide for lightweight scenarios and small tests, it can run into concurrency issues with higher volumes of transactions. In this instance, it is beneficial to switch to a more concurrent engine, such as PostgreSQL [31] or MySQL [32].

To provide a high-performance interface to clients, CherryPy [33] is used to serve user requests. This has the advantage of providing a thread-pooled server, allowing more than one user request to be processed concurrently - an important consideration, with the potential amount of data being received.

*2) Manager:* The majority of the manager's functionality resides around communications with the supervisor and distribution of transformation processing to available processor nodes. To distribute the transformation processing, ParallelPython [34] has been utilised with modifications. Load balancing and task distribution has been modified for the purposes of evaluation, and to distinguish the difference in availability of local vs Cloud Computing resources.

The manager is not a particularly resource-heavy application; as such, it is possible to execute it on a device already being used in another role. This is only recommended for use in small-scale scenarios, and placing the manager on the same device as the supervisor may result in impeded performance.

*3) Monitor:* The monitoring agent is written in Python, with the primary aim of configurability and extensibility. The default agent uses a simple configuration file to initialise connections to servers, such as the supervisor, and to configure

outgoing data streams. As the configuration file is a Python script itself, the monitor should be able to port data from any monitoring device into a format appropriate for transmission to the supervisor, including Bluetooth and serial connections, assuming appropriate connectivity is available.

The monitor script itself is fully multi-threaded, providing the ability to collect and transmit data from many sources at once. This allows the monitor to transmit device state information along with physiological data, providing the supervisor with useful data on battery and load levels. Data collation and transmission rate is configurable, and monitor collection rate is configurable on a per-device basis. Monitoring devices themselves can be used as low-power transmitters, or paired with the processing script to become mobile transformation processors - particularly useful in mobile scenarios.

*4) Processor:* The processor is a relatively simple script used as part of the ParallelPython [34] distributed processing library, written in Python. As such, it runs on similar platforms to all other scripts. This is beneficial, as it means that any device chosen to use for testing can be used as a processing agent, and initialising Cloud Computing resources for processing is as simple as executing a single script.

All system scripts have been tested and execute successfully on desktop PCs (using Python 2.6 and 2.7), all Cloud computing resources (using both Unix-like operating systems and Microsoft Windows) and several mobile devices running the Android mobile operating system (using SL4A [35] with Python 2.6). It is recommended that Android devices use the SQLite engine, while any supervisor using a PC (such as a laptop) can use any other supported database management system.

## IV. Evaluation

In this section, the suitability of the system in particular situations is evaluated, primarily using performance (or system responsiveness) as a comparative metric. Latency is also evaluated as an additional factor that can inhibit performance, based on the architectural and physical location of processing resources. Performance of the system is heavily related to the processing power of the devices upon which it is implemented. As such, we evaluate implementation performance across a series of different architectures, including multiple mobile devices, a mix of mobile/local and mobile/Cloud. Implementation responsiveness is a good indicator of the system's ability to perform transformations in a timely fashion, providing insight into performance during real-world scenarios.

The equipment used in the evaluation is as follows; 3 devices running Android 4.0 (HTC One X, HTC Desire, Acer Iconia A500), a workstation running Windows 8 (Intel i5-2500K) and an Amazon Linux EC2 Instance (High-CPU Medium). This selection of devices allows us to evaluate each level of resources as defined in Figure 4, from peer-level (mobile devices) to local (workstation) and Cloud computing (Amazon EC2).

To evaluate the effectiveness of the system in a typical usage scenario, an appropriate amount of test data was generated to support the execution of three data transformations in a chain. This transformation chain consists of three steps. It performs a Discrete Fourier Transform upon a randomly-generated sample of data typical of Electrocardiogram output, then takes resulting data and performs some basic arithmetic upon the output of approximate complexity to transforming R-R interval to heart rate. It then combines results from the previous two transformations to evaluate the processing and transmission of sizable data sets. Finally, the results of the first and third transform are returned to the Supervisor (and from there, to the Client).

### A. Responsiveness

For the system to be considered useful in a wide range of potential usage environments, it must respond to user requests as quickly as possible, a term referred to as responsiveness. System responsiveness can be a key consideration when attempting to retrieve data related to health, as lack of responsiveness can have disastrous effects. Additionally, responsiveness provides a useful metric to compare transformation processing capability of a single device (as traditionally used) as opposed to a cluster of devices, such as this paper proposes.

This test compares the performance of the physiological monitoring and processing system on a single device (HTC Desire) compared to multiple (Desire, One X), as well as utilising locally-available resources (Workstation), which would not typically be available in mobile environments.


Fig. 5. Responsiveness, ordered by request time

As expected, the system in single-device mode (as would traditionally be used in a mobile environment) performs poorly. The rapid increase in request time is indicative of the device being unable to cope with the workload being allocated to it.

Using a second mobile device as an additional Processor proved more successful than anticipated - even outperforming the local resources available. This reduced the inevitable reduction in system responsiveness to a minimum, decreasing at a much lower rate compared to both the single-device and local resources setups. The two-device setup almost halved request time in comparison to the single-device, providing impressive levels of scalability.

Using three devices for processing still resulted in performance gains over two devices, though to a lesser degree. Figure 5 shows the response time of requests ordered by the time of request, showing us the performance of the system over the period of execution. The third device generally improves performance, but the low intensity level of transformations ensures that the overhead of managing three devices starts to detract from the gains of having more available processing power. In the event that transformations were of higher intensity, the benefit of having additional processing devices is likely to be more clear.

### B. Latency

Latency between monitoring devices and processing resources can have a large impact on the usefulness of data processed by the system. If real-time feedback is required (particularly in the event of health monitoring), the associated decrease in responsiveness due to physical distance to processing resources can be problematic. To evaluate the responsiveness of the system over multiple layers of resources, as described in Figure 4, transformations are requested from a mobile device. This mobile device, acting as both Supervisor and Manager, utilises itself as a Processor as well as available resources at individual layers. Latency from the Supervising mobile device to the Peer was relatively small ( 10ms), highly variable to the Workstation ( 30-90ms), and typical of international transit to the Amazon Cloud computing resources ( 200-260ms).

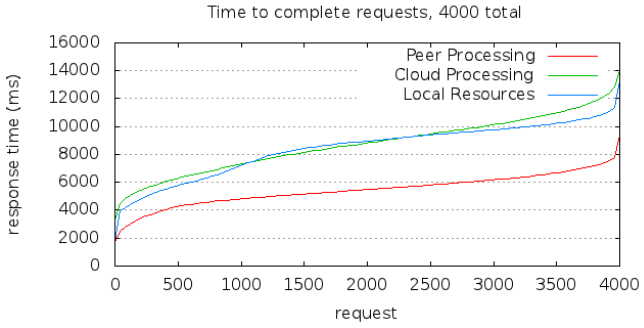


Fig. 6. System Responsiveness with Different Resource Levels

Figure 6 illustrates the total request time by each processing layer. The easily-accessible Peer layer provides the most responsive processing, with Local and Cloud computing resources taking significantly longer to respond to Client transformation requests. However, this is likely due to the low complexity of transformations being performed on the data - more complex transforms would encourage the higher processing capability of Local and Cloud resources to be more significant.

Figure 7 demonstrates the difference made if transformations are of higher intensity, with the benefits of low-latency connections displaced by the time taken to process each transformation. However, all available processing options are still preferable to solo-processing with a mobile device. Local resources become the most attractive option in this experiment when using high-intensity transformations, though allocating significantly higher levels of Cloud computing resources would alter this outcome.

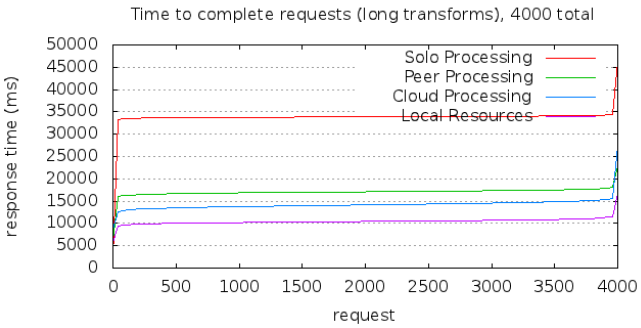


Fig. 7. System Responsiveness with Different Resource Levels - Long Transforms, including Solo Processing

This evaluation suggests that mobile devices collaboratively processing physiological transformations can, in most circumstances, provide an adequate level of system responsiveness. It outperforms local and Cloud computing resources in situations where latency to these resources is high, connectivity is unstable or transformations are reasonably low-intensity. The more powerful resources are recommended for use in situations where connectivity is stable or transformations are high-intensity, and using these resources has the additional benefit of reducing power consumption by the monitoring devices. However, particularly low-intensity transformations may consume less power to process on mobile devices than to transmit over communications with high power consumption, such as 3G/4G (commonly used mobile data networks).

## V. Conclusion

Physiological monitoring provides significant benefits in the areas of disease diagnosis, rehabilitation evaluation and assessing subjects for stress, amongst other uses. Existing physiological monitoring techniques generally monitor and store physiological data for later submission to centralised processing servers that can perform physiological analysis and transformations upon the collected data.

To enable real-time monitoring of physiological measures, devices rely either on the presence of fixed processing resources or a reliable communications network that it can use to relay data to processing servers for transformation. Several environments exist in which physiological monitoring would be beneficial for health or evaluation reasons, but are unsuitable for existing monitoring techniques. These environments tend to be mobile environments, where fixed power supplies (and therefore fixed processing resources) are unavailable, or where communications networks have little penetration, such as highly remote or heavily forested areas.

In order to reduce reliance on these two environmental variables and improve physiological monitoring by expanding the

range of potential usage environments, a shift to a more mobile paradigm is recommended. By encouraging the use of existing mobile devices as a collaborative processing framework to perform physiological data transformations, both problems are alleviated. Smartphones contain their own portable power supplies, that can be recharged periodically using a generator or photovoltaic cells. The devices are also extremely portable, having been specifically designed for use while mobile.

This paper presented an evaluation of the viability of using mobile devices to collaboratively process physiological transformations, which determined that such a technique is scalable and viable for use in real-world situations. In several environments, distributed mobile processing actually offers greater benefits than using traditional methods or Cloud Computing, as latency to local mobile devices is significantly lower than external resources.

## REFERENCES


[1] N. Oliver and F. Flores-Mangas, "HealthGear: a real-time wearable system for monitoring and analyzing physiological signals," in *International Workshop on Wearable and Implantable Body Sensor Networks, 2006. BSN 2006*. IEEE, Apr. 2006, pp. 4 pp.–64.

[2] G. G. Mendoza and B. Q. Tran, "In-home wireless monitoring of physiological data for heart failure patients," in *Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, vol. 3. IEEE, Oct. 2002, pp. 1849–1850 vol.3.

[3] Y. Lin, I. Jan, P. Ko, Y. Chen, J. Wong, and G. Jan, "A wireless PDA-based physiological monitoring system for patient transport," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 8, no. 4, pp. 439 –447, Dec. 2004.

[4] B. Lin, B. Lin, N. Chou, F. Chong, and S. Chen, "RTWPMS: a Real-Time wireless physiological monitoring system," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 4, pp. 647 –656, Oct. 2006.

[5] S. Dai and Y. Zhang, "A wireless physiological multi-parameter monitoring system based on mobile communication networks," in *Computer-Based Medical Systems, 2006. CBMS 2006. 19th IEEE International Symposium on*, 2006, pp. 473 –478.

[6] E. Jovanov, A. O'Donnell Lords, D. Raskovic, P. G. Cox, R. Adhami, and F. Andrasik, "Stress monitoring using a distributed wireless intelligent sensor system," *IEEE Engineering in Medicine and Biology Magazine*, vol. 22, no. 3, pp. 49– 55, Jun. 2003.

[7] Zephyr Technology, "CASE STUDY: zephyr provides physiological monitoring of chilean miners during san jose mine rescue operation," http://www.zephyr-technology.com/resources/case-studies, 2010. [Online]. Available: http://www.zephyr-technology.com/resources/case-studies

[8] J. D. Malone, R. Brigantic, G. A. Muller, A. Gadgil, W. Delp, B. H. McMahon, R. Lee, J. Kulesz, and F. M. Mihelic, "U.S. airport entry screening in response to pandemic influenza: Modeling and analysis," *Travel Medicine and Infectious Disease*, vol. 7, no. 4, pp. 181–191, Jul. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1477893909000313

[9] E. Jovanov, D. Raskovic, A. O. Lords, P. Cox, R. Adhami, and F. Andrasik, "Synchronized physiological monitoring using a distributed wireless intelligent sensor system," in *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2003*, vol. 2. IEEE, Sep. 2003, pp. 1368– 1371 Vol.2.

[10] P. Casari, A. P. Castellani, A. Cenedese, C. Lora, M. Rossi, L. Schenato, and M. Zorzi, "The "Wireless sensor networks for City-Wide ambient intelligence (WISE-WAI)" project," *Sensors*, vol. 9, no. 6, pp. 4056–4082, May 2009. [Online]. Available: http://www.mdpi.com/1424-8220/9/6/4056

[11] L. N. Katz and A. Pick, *Clinical electrocardiography*. Lea & Febiger, 1956.

[12] G. B. Moody, R. G. Mark, A. Zoccola, and S. Mantero, "Derivation of respiratory signals from multi-lead ECGs," *Computers in Cardiology*, vol. 12, p. 113–116, 1985.

[13] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *Physiological Measurement*, vol. 28, no. 3, pp. R1–R39, Mar. 2007. [Online]. Available: http://iopscience.iop.org/0967-3334/28/3/R01

[14] S. G. Fleming and L. Tarassenko, "A comparison of signal processing techniques for the extraction of breathing rate from the photoplethysmogram," *World Academy of Science, Engineering and Technology*, 2007.

[15] J. Healey and R. Picard, "Digital processing of affective signals," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 6, 1998, p. 3749–3752.

[16] B. Sangeeta and S. Laxmi, "A real time analysis of PPG signal for measurement of SpO2 and pulse rate," *International Journal of Computer Applications*, vol. 36, no. 11, p. 45–50, 2011.

[17] P. Addison and J. Watson, "Secondary wavelet feature decoupling (SWFD) and its use in detecting patient respiration from the photoplethysmogram," in *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, vol. 3, Sep. 2003, pp. 2602 – 2605 Vol.3.

[18] D. Chu and M. Humphrey, "Mobile OGSI.NET: grid computing on mobile devices," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, Nov. 2004, pp. 182 – 191.

[19] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Workshop on compilers and operating systems for low power*, vol. 180, 2001, p. 182–195.

[20] R. W. Hoyt, J. Reifman, T. S. Coster, and M. J. Buller, "Combat medical informatics: present and future." *Proceedings of the AMIA Symposium*, pp. 335–339, 2002, PMID: 12463842 PMCID: PMC2244161.

[21] Z. Solomon, M. Mikulincer, and S. E. Hobfoll, "Effects of social support and battle intensity on loneliness and breakdown during combat." *Journal of Personality and Social Psychology; Journal of Personality and Social Psychology*, vol. 51, no. 6, p. 1269, 1986.

[22] C. B. Lapierre, A. F. Schwegler, and B. J. LaBauve, "Posttraumatic stress and depression symptoms in soldiers returning from combat operations in iraq and afghanistan," *Journal of Traumatic Stress*, vol. 20, no. 6, pp. 933–943, Dec. 2007. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/jts.20278/abstract

[23] R. Fielding, "A little REST and relaxation," in *The International Conference on Java Technology (JAZOON07), Zurich, Switzerland*, 2007.

[24] Amazon Web Services, "Amazon elastic compute cloud (Amazon EC2)," http://aws.amazon.com/ec2/, Aug. 2006. [Online]. Available: http://aws.amazon.com/ec2/

[25] Google, "Google app engine," https://developers.google.com/appengine/, Apr. 2008. [Online]. Available: https://developers.google.com/appengine/

[26] ——, "Android," http://www.android.com/, Sep. 2008. [Online]. Available: http://www.android.com/

[27] Python Software Foundation, "Python programming language," http://www.python.org/, 1990. [Online]. Available: http://www.python.org/

[28] M. Bayer, "Mako templates," http://www.makotemplates.org/, 2012. [Online]. Available: http://www.makotemplates.org/

[29] ——, "SQLAlchemy - the database toolkit for python," http://www.sqlalchemy.org/, Jan. 2008. [Online]. Available: http://www.sqlalchemy.org/

[30] D. R. Hipp and D. KENNEDY, "SQLite," *Available fr om: http://www.sqlite. org*, 2007.

[31] PostgreSQL, "PostgreSQL database management system," http://www.postgresql.org/, 1996. [Online]. Available: http://www.postgresql.org/

[32] Oracle Corporation, "MySQL database management system," http://www.mysql.com/, 2012. [Online]. Available: http://www.mysql.com/

[33] CherryPy, "CherryPy - a minimalist python web framework," http://cherrypy.org/, 2001. [Online]. Available: http://cherrypy.org/

[34] V. Vanovschi, "Parallel python," http://www.parallelpython.com/, 2005. [Online]. Available: http://www.parallelpython.com/

[35] SL4A, "Scripting layer 4 android," http://code.google.com/p/android-scripting/, 2010. [Online]. Available: http://code.google.com/p/android-scripting/