Chapter XV
Context-Aware Task Redistribution for Enhanced M-Health Application Performance

Hailiang Mei
University of Twente, The Netherlands

Bert-Jan van Beijnum
University of Twente, The Netherlands

Ing Widya
University of Twente, The Netherlands

Val Jones
University of Twente, The Netherlands

Hermie Hermens
University of Twente, The Netherlands

ABSTRACT

Building context-aware mobile healthcare systems have become increasingly important with the emergence of new medical sensor technologies, the fast adoption of advanced mobile systems, and improved quality of care required by today's patients. A unique feature of our mobile healthcare system is a distributed processing paradigm whereby a set of bio-signal processing tasks is spread across a heterogeneous network. As well as applying the traditional adaptation methods such as protocol adaptation and data

prioritization, the authors investigate the possibility of adaptation based on dynamic task redistribution. In this chapter, the authors propose an adaptation middleware that consists of a task assignment decision mechanism and a task redistribution infrastructure. The decision mechanism represents task assignment as a graph mapping problem and searches for the optimal assignment given the latest context information. Once a new assignment is identified, the member tasks are distributed accordingly by the distribution infrastructure. A prototype implementation based on the OSGi framework is reported to validate the task redistribution infrastructure.

INTRODUCTION

Telemedicine has been receiving more and more attention due to its potential amongst others to tackle the resource challenges to the healthcare system posed by the aging society, by improving the quality of diagnosis and treatment and by reducing the costs of delivering healthcare (TelemedicineAlliance, 2004). Being part of telemedicine, mobile healthcare (m-health) is emerging along with the fast adoption of advanced mobile technology into daily life. Several m-health systems have been developed for mobile network environments (Halteren, Bults, Wac, Konstantas, & Widya, 2004; Hung & Yuan-Ting, 2003; Rasid & Woodward, 2005; Yuan-Hsiang et al., 2004), in which an m-health platform is introduced comprising a patient Body Area Network and some back-end healthcare service facilities linked by wireless communications links. On top of the platform, multiple applications, such as tele-monitoring and tele-treatment services, can be operated to provide continuous (24/7) mobile services to patients.

However, like other applications in a mobile environment, the performance of m-health systems can be seriously affected by context changes and scarcity of the platform resources, e.g. network bandwidth, battery power and computational power of handhelds (Halteren, Bults, Wac, Konstantas, & Widya, 2004; Jones, Incardona, Tristram, Virtuoso, & Lymberis, 2006). From a technical point of view, to solve this mismatch between application demand and resources, an appropriate context-aware adaptation mechanism should be embedded into the system. Satyanarayanan (Satyanarayanan, 2001) identifies three approaches to building such adaptation mechanisms: (1) task adjustment - this is to automatically change task behavior to use less of a scarce resource, e.g. scalable video transmission over wireless network; (2) resource reservation - this is to ask the environment to guarantee a certain level of a resource, e.g. QoS (Quality of Service) management and reservation techniques; (3) user notification - this is to suggest a corrective action to the user. The second approach assumes that it is possible to reserve sufficient resources for the task, which is sometimes unrealistic, e.g. the drop of network bandwidth could be so significant that the required data transmission quality just cannot be met. The third approach could avoid the mismatch by giving the patient suggestions or warnings, e.g. "Please stay near to a charging point to reduce the risk caused by draining battery power". However, restricting users' mobility in this way is a far from satisfactory solution. Therefore, we focus on the first "adjusting task demand" approach to tackle the problem in m-health of mismatch between demand and resources.

Previously, adjusting tasks was often performed within an isolated device, e.g. by a local application-specific adaptor (Badrinath et al., 2000). Methods applied in the past include data compression, discarding less important informa-

tion and handover to a better network connection. The fundamental model common to remote monitoring systems consists of a set of bio-signal data processing tasks distributed across a set of networked devices. Therefore, one possible adaptation scenario is to exploit the distributed processing paradigm and adjust the assignment of tasks across available devices at run-time. The rationale is that, at a particular moment, if one device cannot support a task in terms of computation or data communication demands, some other devices with richer resources may be available to take over this task. The advantage over traditional methods is that the user requirements are less likely to be compromised and distributed resources can be better utilized.

To realize this task redistribution based adaptation in m-health systems, two major topics are to be addressed: (1) how to determine a suitable task assignment and (2) how to reconfigure the tasks across the devices according to this assignment with minimum service disruption. Targeting m-health systems specifically, we would like to address the following questions in this chapter:

1. What are the requirements for task redistribution in m-health systems?
2. What are the QoS performance measures relevant for the m-health application and which performance characteristics can be improved by a "more suitable task assignment"?
3. How can an optimal task assignment be computed given one particular performance measure?
4. What is the impact of the redistribution of tasks on the continuity of the services and what are the potential techniques to minimize these disruptions?

This chapter is organized as follows. Section 2 motivates our research by presenting a context-aware m-health system and its application scenario, which allows us to further study the requirements on task-redistribution and relevant performance measures. In Section 3, we formulate the problem of task assignment in m-health in order to support task redistribution. Section 4 classifies the task assignment problem into several groups based on the model setting and presents the solutions. Section 5 presents the design and implementation of a component-based infrastructure to support the task redistribution. Section 6 provides some further discussions relate to both technical outcomes and non-technical outcomes. Section 7 concludes this chapter.

TASK-REDISTRIBUTION BASED ADAPTATION IN CONTEXT-AWARE M-HEALTH SYSTEM

The M-Health System

The m-health system under study consists of a body-worn set of devices (Body Area Network or BAN) communicating with a remote healthcare provider via an m-health server. This distributed system incorporates the following devices in the mobile and fixed networks: one or more body-worn sensors connected to a sensor front end (a "sensorbox"), a handheld (in this case a PDA) acting as the Mobile Base Unit (MBU) of the BAN, a back-end server and an end-terminal (Figure 1). This m-health system was developed over the course of a number of European and Dutch projects (IST MobiHealth[a], eTEN HealthService24[b] and FREEBAND AWARENESS[c]). The epileptic seizure detection application presented in the next section was developed during the AWARENESS project. The data acquisition system (sensors and sensorbox) collects bio-signals of the patient or other data such as location or activity and sends the data to the MBU over a short-range wireless connection, e.g. a Bluetooth connection. The MBU acts as a processor and communications gateway and thus can process

the data and send it to the back-end server of the healthcare portal over a wireless link (currently WiFi, GPRS or UMTS). Thereafter, the data can be streamed to (or stored for future access by) a healthcare professional using his end-terminal, e.g. his laptop or desktop PC.

In this distributed and mobile environment, application context including user situation, device capacity and network connection changes from time to time. M-health applications need to adapt to these context changes in order to provide timely and adequate services. Such context awareness is illustrated through a scenario in the next section.

Context-Aware Epileptic Seizure Detection

Epilepsy is a serious chronic neurological condition characterized by recurrent unprovoked seizures. If detection or even prediction of seizures by a few seconds were possible this would give the patient a chance to prepare and for appropriate medical assistance and/or advice to be given. Since seizures may happen anywhere and at any time, providing a mobile monitoring service for epilepsy patients can be very beneficial. The scenario below illustrates the AWARENESS vision of the context-aware mobile monitoring services for epileptic seizure detection to be provided by the m-health system and shows why the system needs to adapt to changing contextual factors.

John is an epilepsy patient who had been seizure-free for several years. He wears a mobile monitoring system which monitors his health state and can give him a few seconds' advance warning of an upcoming seizure. The m-health monitoring service constantly runs an epileptic seizure detection algorithm (Figure 2) which analyses the ECG signals in combination with context information about John's current activity levels derived from the activity sensor.

When John is at home, a broadband network is available to transfer his raw ECG and activity information to the remote monitoring centre, e.g. the back-end server in (Figure 1). In this case, all tasks in the detection algorithm are deployed on the back-end server and the doctor can be warned if a seizure is likely to occur. One afternoon, John is out jogging, following his usual route through the forest. Since there is no broadband network available in the forest, John's bio-signals cannot be transmitted due to insufficient network bandwidth. Therefore, some processing tasks are reassigned from the remote server to his MBU and his bio-signals are processed locally. During his run, the

Figure 1. The M-health system enables a remote health professional to view processed bio-signals and take appropriate action for the patient

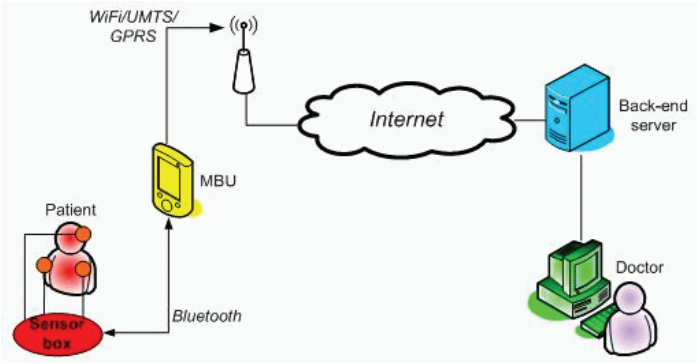
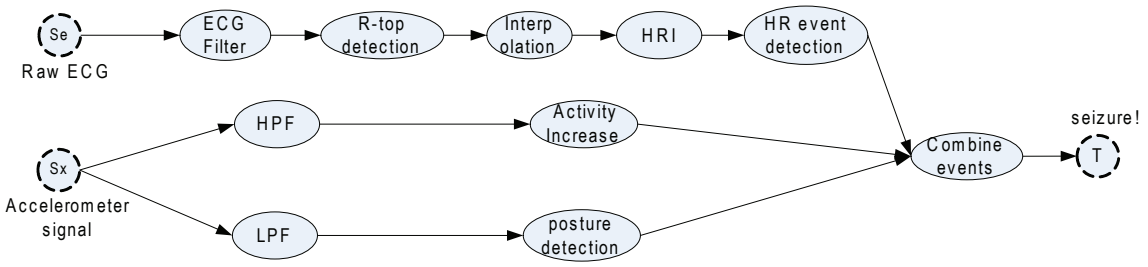


Figure 2. Distributed bio-signal processing tasks in a epileptic seizure detection algorithm studied in the AWARENESS project (Tönis, Hermens, & Vollenbroek-Hutten, 2006): Raw ECG and movement information (obtained by accelerometers) are processed in distinguished steps to estimate the chance of a seizure attack



signal processing algorithm detects a possible imminent epileptic seizure. John is immediately warned by his BAN, and stops running. At the same time, an alarm and John's GPS position are sent to the monitoring centre via a narrow band connection, e.g. GPRS or GSM. The alarm triggers the monitoring service to take appropriate action; for instance, depending on the circumstances, the emergency services or an informal caregiver can be dispatched to the exact location where John is to render emergency assistance.

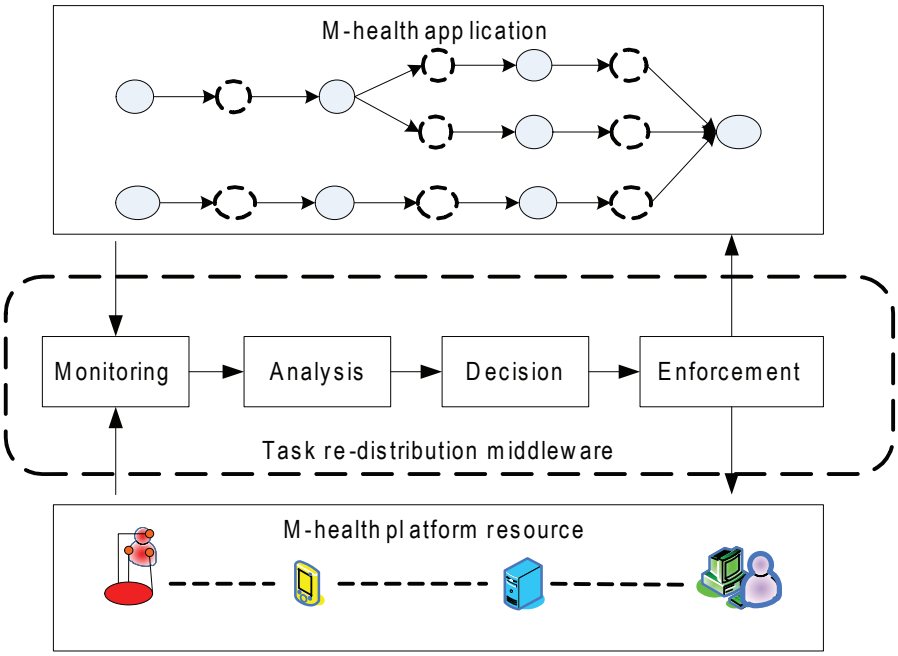
MADE – A Middleware Layer Supporting Task Redistribution Based Adaptation

As illustrated in the scenario, it is desirable that the m-health system can respond to context changes by properly redistributing the bio-signal processing tasks at runtime. To achieve this, we propose a so-called *MADE* middleware consisting of four parts: Monitoring, Analysis, Decision and Enforcement shown in Figure 3. The monitoring part includes registration of the m-health application, device discovery, resource monitoring, and context discovery/registration. The analysis part takes this information on the m-health application and system as input and runs a task assignment algorithm to determine the assignment with optimal system QoS performance under these circumstances. The task assignment algorithm can be executed based on a predetermined schedule or triggered by a "significant" context change, e.g. John moves out of an area of WLAN coverage into a GPRS only area. The decision part compares the computed optimal assignment with the current system configuration to determine the actual cost of reconfiguration. If the reconfiguration cost can be covered by the enhanced performance of the new configuration, the new assignment plan will be executed. The Enforcement part controls the m-health system to adjust its configuration according to the new assignment.

Performance Enhancement by Task Redistribution

Different assignments of application tasks to system devices result in different system configurations and each configuration exhibits different QoS performance characteristics. The main goal of the *MADE* middleware layer is to identify a system configuration with optimal QoS performance and reconfigure the m-health application/system by means of task redistribution. From the perspective of the end user, e.g. the doctor in Figure 1, we focus on the follow-

Figure 3. MADE – a task redistribution based adaptation middleware that can dynamically redistribute the application tasks across platform resources



ing QoS characteristics that are critical to the success of m-health mission. There are certainly other critical performance measures not covered here, in particular data security and privacy. We investigate here only the QoS performance measures that can be improved based on the task redistribution approach.

1. "End-to-end delay" is defined as the elapsed time between the m-health system receiving a frame of patient's bio-signal information and sending the corresponding processed result of this particular bio-signal frame to the decision point. This parameter indicates how quickly the bio-signal and processed result can be delivered by the m-health system. The faster the real-time bio-signal information is delivered, the higher the chance that the patient's emergency situation can be dealt with in time, e.g. in the time critical epileptic seizure detection application.
2. "M-health application availability level". The availability measure we consider here is the steady state availability as defined in (Muppala, Fricks, & Trivedi, 2000), that is the application mean uptime divided by the sum of the mean uptime and mean downtime. Failures may potentially occur during either data processing or communication.
3. "System battery lifetime" is defined as the minimum battery lifetime of all the battery powered devices in the m-health system. Some devices in the m-health system are powered by batteries; if the remaining battery energy is lower than a certain level, the node cannot perform bio-signal processing operations anymore, and thus, the m-health system cannot provide the necessary information to the decision point. This parameter indicates the maximum operating time of the m-health system.

Task Redistribution Based Adaptation Requirements

Based on the scenario described above, we can derive two non-functional requirements on the task redistribution based adaptation in order to make it effective:

- The adaptation should be agile, that is the speed at which the system detects and adapts to context changes should be fast in order to minimize service interruption.
- The adaptation should be cost effective: system reconfiguration by task redistribution consumes additional platform resources, so if the aim of adaptation is to reduce the usage of a certain resource, e.g. battery energy, it is important that the reconfiguration does not cost more than what can be saved.

TASK ASSIGNMENT PROBLEM STATEMENT

The core of the *MADE* middleware layer is the "analysis" part where a task assignment algorithm is required to compute the optimal task assignment given the application/system context information. In this section, we discuss this task assignment problem in more detail and present the mathematical model of the problem.

Definitions

We define an m-health remote monitoring application as a partial order of bio-signal streaming tasks. In our model we distinguish two types of streaming tasks: stream processing tasks (c.f. Figure 2) and stream transmission tasks. Processing tasks typically perform some operation on the bio-signal stream such as filtering, transcoding, or other m-health relevant signal- or data processing operations. Each processing task consumes one or more data streams and produces one or more data streams. Transmission tasks are the glue between processing tasks and have two functions: firstly they allow us to easily characterize properties of the data stream (for instance the data rate of the stream); and secondly, as we will see later, transmission tasks can be mapped onto a communication path (such a path may be a stream pipe within a device, or it may be a networked path between different devices).

An m-health application consisting of distributed tasks can be defined as a tuple of (P, T, A_t, L_P, L_T), where P is a set of stream processing tasks $\{p_1,p_2,\ldots\}$, T is a set of transmission tasks $\{t_1,t_2,\ldots\}$, A_t is a set of precedence relations between tasks, such that $A_t \subseteq P \times T \cup T \times P$, L_P is a set of labels over each processing task, L_T is a set of labels over each transmission task. L_P and L_T indicate the resource demand of processing tasks and transmission tasks respectively, a detailed overview is given in Table 1. The structure (P, T, A_t) is a (bipartite) directed acyclic graph (DAG) termed a *task DAG.* An example of a task DAG is given in Figure 4.

Similarly, an m-health system is defined as a tuple of (D, C, A_r, L_D, L_C), where D is a set of device resources $\{d_1,d_2,\ldots\}$, C is a set of (communication) channel resources $\{c_1,c_2,\ldots\}$, A_r is a set of precedence relations between resources, such that $A_r \subseteq D \times C \cup C \times D$, L_D is a set of labels over each device resource, L_C is a set of labels over each channel resource. L_D and L_C model the resource supply of devices and channels respectively, further details of this context information are given in Table 1. The structure (D, C, A_r) is a directed acyclic graph termed a *resource DAG, an example of such a graph is shown in* Figure 4.

We assume that device resources in an m-health system can relay bio-signal data streams, therefore a transmission task may be assigned to a communication path. A communication path is

defined as follows. Given two device resources d_i and d_j, a communication path *cp* is a path in the resource graph starting at d_i and ending at d_j. In general there may exist multiple paths from d_i to d_j, the set $CP_{di,dj}$ denotes the set of all paths from d_i to d_j. Note that when $i=j$ the communication path is the empty path, denoted ε, and $CP_{di,dj}=\{\varepsilon\}$.

Both task DAGs and resource DAGs are bipartite graphs in which each type of vertex forms a disjoint set, i.e. there is no connection between two vertices of the same type. In a task DAG, each transmission task has exactly one predecessor processing task and one successor processing task, while a processing task may have multiple predecessor transmission tasks and multiple successor transmission tasks. A similar property holds for a resource DAG.

Regarding the semantics of the task DAG, we have one important assumption namely that whenever a processing task has multiple successor transmission tasks, the bio-signal stream transfers over each of these transmission tasks. Whenever a processing task has multiple predecessor transmission tasks, the bio-signals entering from all these transmission tasks are synchronized. In effect, we only allow bio-signal forking and joining with AND semantics (as known from workflow theory).

Based on the two graph models, a task assignment is a mapping of tasks onto resources such that: processing task is mapped to one device task and each transmission task is mapped to communication path. In general, many different task assignments exist, and the objective is to find the best task assignment based on a

Figure 4. Model of task assignment from an m-health application, i.e. task DAG (Directed Acyclic Graph), to an m-health system, i.e. resource DAG

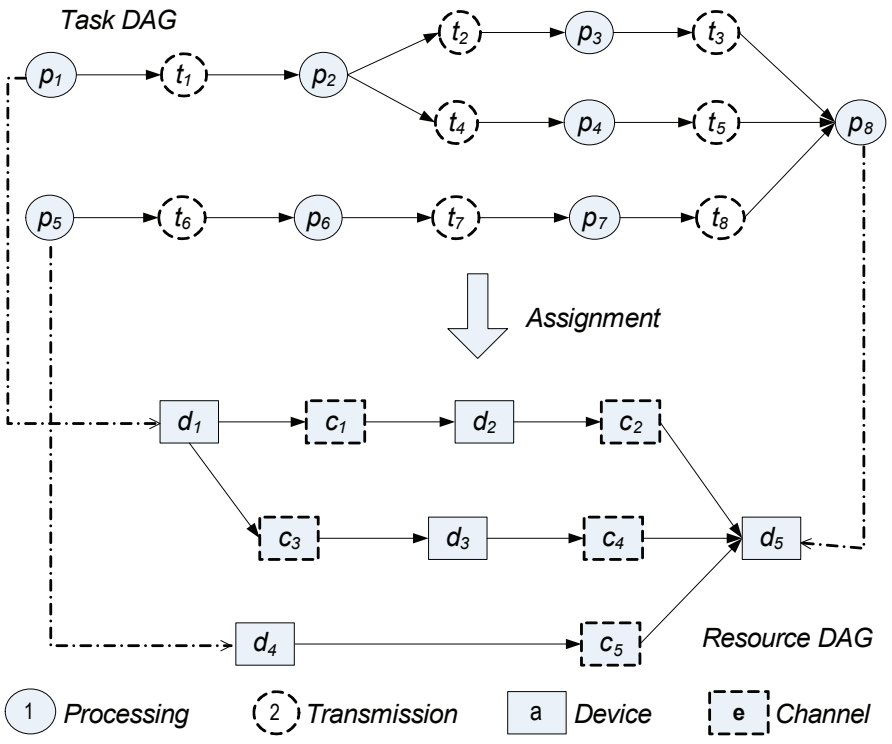


Table 1. Notations for labeling

Notation B	*elongs to*	*Meaning*
n_{pi}^{op}	L_P	the number of operations per second at processing task p_i
rr_{pi}	L_P	required resource for processing, e.g. minimum required CPU, minimum required memory, etc.
$av_{pi,di}$, $av_{ti,ci}$ or $av_{ti,di}$	L_P / L_T	availability measure of running a processing or transmission task at a particular resource[1].
n_{ti}^{tr}	L_T	the number of transmitted data units per second at transmission task t_i
e_{di}^{to}	L_D	total available battery energy at device d_i[2]
e_{di}^{hk}	L_D	energy consumption of device's "housekeeping" activities per second, e.g. CPU, display, powering network interface cards
e_{di}^{op}	L_D	energy consumption per operation on the device d_i
rs_{di}	L_D	available resource supply for processing at device d_i , e.g. CPU type, available memory, etc.
bw_{ci}	L_C	available bandwidth at channel c_i
lo_{ci}	L_C	Current load information at channel c_i
e_{ci}^{se}	L_C	energy consumption for sending one data unit through channel c_i
e_{ci}^{re}	L_C	energy consumption for receiving one data unit through channel c_i

Table 2. List of Symbols (exclude labeling symbols)

Notation	*Meaning*
D	a set of device resources
C	a set of (communication) channel resources
P	a set of processing tasks
T	a set of transmission tasks
CP	a set of communication paths in the resource DAG
A_t	a set of precedence relations between tasks
A_r	a set of precedence relations between resources
Ω^{de}	end-to-end delay
Ω^{av}	availability
Ω^{li}	battery lifetime
Ω	overall performance measure
$de_{pi,dj}^{pr}$	processing delay for processing task p_i to process one frame of bio-signal data at device d_j
$de_{ti,cpj}^{tr}$	transmission delay for transmission task t_i to transfer one frame of bio-signal data over the communication path cp_j
$de_{ti,dj}^{tr}$	transmission delay for transmission task t_i to transfer one frame of bio-signal data at device d_j
$de_{ti,cj}^{tr}$	transmission delay for transmission task t_i to transfer one frame of bio-signal data at channel c_j
e_{di}^{P}	energy consumption for processing at device d_i
e_{di}^{S}	energy consumption for sending data stream at device d_i
e_{di}^{R}	energy consumption for receiving data stream at device d_i

cost performance evaluation functions. A formal treatment of the task assignment problem is presented next.

Problem Formulation

We formulate the task assignment problem in m-health as follows:

- **Given:** (1) an m-health application (P, T, A_t, L_P, L_T); (2) an m-health system (D, C, A_r, L_D, L_C); (3) A QoS performance evaluation function Ω for task assignments.
- **Goal:** To find the optimal task assignment Ψ_{opt} among all possible task assignment $\{\Psi_1, \Psi_2, ...\}$ such that $\forall_{i\in[1,2,...]}\ \Omega(\Psi_{opt}) \geq \Omega(\Psi_i)$.
- **Subject to:** three assignment constraints namely *type constraint, local constraint* and *reachability constraint.*

The *type constraint* specifies that each processing task must be mapped to one device resource and each transmission task must be mapped to a communication path, hence:

$\forall p_i \in P$: $\Psi(p_i) \in D$, and $\forall t_i \in T$: $\Psi(t_i) \in CP$

The *local constraint* comprises two parts:

1. For every processing task p_i in P, its assigned device resource must provide the task's required resources, e.g. minimal CPU speed, library support, special hardware requirement, etc. That is, assuming we have a Boolean function *satisfy*(*rr*, *rs*) to evaluate whether a required resource, *rr*, can be satisfied by a resource supply, *rs*. Hence:

   $\forall p_i \in P$: $satisfy\ (rr_{pi}, rs_{\Psi(pi)})$

2. For each channel resource, the total assigned transmission tasks must not exceed its offered bandwidth:

   $$\forall_{c_i \in C} \sum_{c_i \in \Psi(ti)} n_{ti}^{tr} < bw_{ci}$$

The *reachability constraint* specifies that for each processing task p_i assigned to device resource $\Psi(p_i)$, its predecessor transmission task must be assigned to a communication path ending at $\Psi(p_i)$, and its successor transmission task must be assigned to communication path starting at $\Psi(p_i)$.

QoS Performance Evaluation for a Given Task Assignment

The purpose of the task assignment problem is to find (i.e. compute) the optimal assignment relative to a QoS performance evaluation function. In this section we detail this QoS performance evaluation function further. In general, optimization may be desired for either one specific QoS performance measure, or it may be some QoS performance metric composed from multiple QoS performance measures. In the following we define three different QoS performance measures, and define one QoS performance metric from these three. The definitions rely on the task labeling and resource labeling defined previously in Table 1.

End-to-End Delay

In general, given an m-health application, there may exist multiple paths between any two tasks in the m-health application. For a task DAG, there exist multiple task paths connecting a source task and a sink task. Every path is a sequence of processing tasks and transmission tasks denoted as $tp \subseteq \{P, T\}$ and the ith task in tp is denoted as $tp(i)$. Upon different task assignment Ψ, a task path can exhibit a different end-to-end delay, i.e. the summation of processing delay and transmission delay along the path:

$$\Omega^{de}(tp) = \sum_{tp(i)\in P} de^{pr}_{tp(i),\Psi(tp(i))} + \sum_{tp(i)\in T} de^{tr}_{tp(i),\Psi(tp(i))}$$

where $de^{pr}_{pi,dj}$ is defined as the processing delay for processing task p_i to process one frame of bio-signal data at device d_j, $de^{tr}_{ti,cpj}$ is defined as the transmission delay of transmission task t_i to transfer one frame of bio-signal data over a communication path cp_j. $de^{tr}_{ti,cpj}$ can be computed based on the transmission delay occurred at device d_j, $de^{tr}_{ti,dj}$, and at channel c_j as $de^{tr}_{ti,cj}$. $de^{pr}_{pi,dj}$, $de^{tr}_{ti,dj}$ and $de^{tr}_{ti,cj}$ can be estimated from the profiling information, e.g. n^{op}_{pi}, n^{tr}_{ti}, bw_{ci} and lo_{ci} based on the knowledge from real measurements, e.g. (Catalan et al., 2005; Xiao, Rosdahl, Center, Linear, & Draper, 2002).

The end-to-end delay of a given assignment Ψ, $\Omega^{de}(\Psi)$, is then defined as the maximum of all task paths' end-to-end delays:

$$\Omega^{de}(\Psi)=\max\left(\left\{\Omega^{de}(tp)\right\}\right)$$

Application Availability Level

Since we assume only "AND" semantics in task DAG, the availability of the whole application depends on all the included tasks: Only when all tasks perform successfully can the application perform successfully. Therefore, the application availability level for a given assignment, $\Omega^{av}(\Psi)$, can be computed as:

$$\Omega^{av}(\Psi)=\prod_{pi\in P} av_{pi,\Psi(pi)} \prod_{ti\in T,\alpha\in\Psi(pi)} av_{ti,\alpha}$$

Battery Lifetime

Based on the power consumption model of a mobile device (Rahmati & Zhong, 2007), we estimate the battery life time $\Omega^{li}(di)$ for a specific (mobile) device d_i given a particular task assignment Ψ as follows:

$$\Omega^{li}(di)=\frac{e^{to}_{di}}{e^{hk}_{di}+e^{P}_{di}+e^{R}_{di}+e^{S}_{di}}$$

Where e^{to}_{di} is the total available battery energy at device, e^{hk}_{di} is energy consumption of device's "housekeeping" activities per time unit, e.g. CPU, display, powering network interface cards, e^{P}_{di} is the energy consumed by local data processing; e^{R}_{di} is the energy consumption for receiving data stream; e^{S}_{di} is the energy consumption for sending data stream. Based on the aforementioned profiling information, these can be calculated per device given Ψ:

$$e^{P}_{di}=e^{op}_{di}\sum_{\Psi(pj)=di} n^{op}_{pj}$$

$$e^{S}_{di}=\sum_{(di,cj)\in A^r}(e^{se}_{cj}\sum_{cj\in\Psi(tk)} n^{tr}_{tk})$$

$$e^{R}_{di}=\sum_{(cj,di)\in A^r}(e^{re}_{cj}\sum_{cj\in\Psi(tk)} n^{tr}_{tk})$$

Once the battery lifetime of all devices are estimated, the minimum of all device's battery lifetime determines the overall system battery lifetime for a given assignment:

$$\Omega^{li}(\Psi)=\min\left(\left\{\Omega^{li}(di)\middle|di\in D\right\}\right)$$

Overall Performance Evaluation

Similar to our earlier work (Widya, Beijnum, & Salden, 2006), the overall QoS measure, Ω, for a particular assignment Ψ across these three dimensions is defined as:

$$\Omega(\Psi)=\sqrt{\frac{(w^{de})^2}{(\Omega^{de}(\Psi))^2}+(w^{av})^2\cdot(\Omega^{av}(\Psi))^2+(w^{li})^2\cdot(\Omega^{li}(\Psi))^2}$$

where w^{de}, w^{av} and w^{li} are the weighting factors among the three QoS measures and should be tuned based on specific applications. After adjusting the "direction" on end-to-end delay, we define the optimal assignment as the one that has highest score on Ω.

SOLUTIONS TO TASK ASSIGNMENT PROBLEM

Earlier work (Norman & Thanisch, 1993) has shown that finding the optimal task assignment is a NP-hard problem. Our problem belongs to this same category. Hence, we may expect only that restricted forms on the general problem can be solved in polynomial time. The model and problem formulation that we have developed in the previous sections differs from others as explained in the following.

- The underlying heterogeneous m-health system has to be considered as a general graph with asymmetric orientations while most earlier models consider a fully connected network with homogenous symmetric connections (Amini, Jain, Sehgal, Silber, & Verscheure, 2006; Ma, Chen, & Chung, 2004; Ucar, Aykanat, Kaya, & Ikinci, 2006). Therefore, many task assignments permitted in earlier work become invalid in our model subject to the new reachability constraint. The search for optimal task assignment is thus required to include an extra validity check on this new assignment constraint.
- The possibility of relaying data stream by devices further complicates the representation of a task assignment. In earlier work, relaying data stream is not really an issue since fully connected networks are often considered; therefore, the earlier assignment function is defined only between processing tasks and devices (Kafil & Ahmad, 1998; Lee & Shin, 1997).
- We model performance measures at a much lower abstraction level, i.e. measures related with energy consumption, delays, etc, while other work merely considers the more abstract measures of computation cost and communication cost (Lo, 1988; Norman & Thanisch, 1993; Ucar, Aykanat, Kaya, & Ikinci, 2006).

In the next subsections we go into further details on the computation of the optimal solution, and we discuss the computation of the optimal solution for two restricted forms of the general problem.

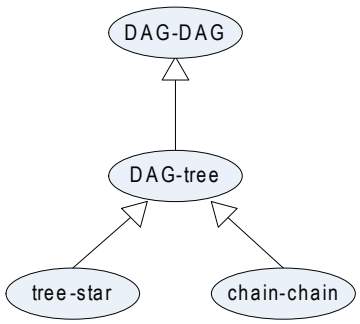
Towards a General Solution

A general approach to search for the optimal task assignment can be summarized as follows:

1. Find or construct a model to represent a task assignment $\Psi:\{P,T\}\rightarrow\{D,CP\}$.
2. Construct a search space consisting of all possible assignments, e.g. a search tree (Franken, 1996; Kafil & Ahmad, 1998).
3. Apply the QoS performance evaluation function for a task assignment found.
4. Identify a search algorithm (which could be brute force) to search for the optimal task assignment, e.g. A* algorithm (Kafil & Ahmad, 1998), branch and bound (Ma, Chen, & Chung, 2004) or graph partitioning.

For some m-health applications and m-health systems, the task assignment problem can be modeled with reduced complexity. In particular, we have studied three specializations of the general "DAG-DAG" task assignment problem: (1) "DAG-tree" where the m-health application is a directed acyclic graph and the m-health system has a tree structure; (2) "Chain-chain" where

Figure 5. Task assignment taxonomy



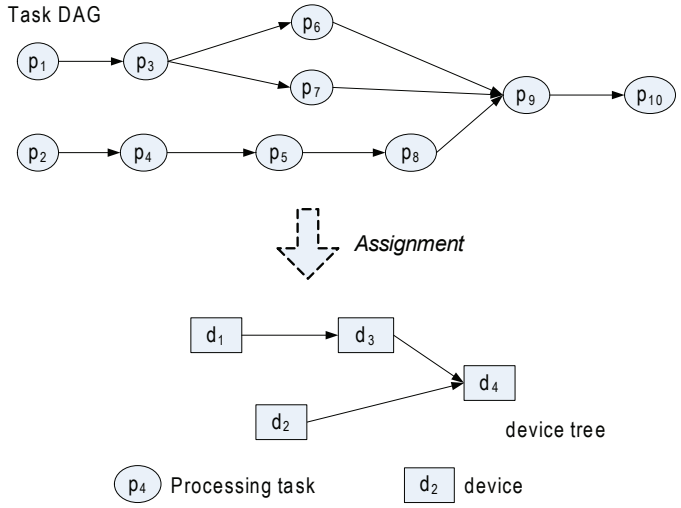
both the m-health application is a tree and the m-health system is a chain; (3) "Tree-star" where the m-health application has a tree structure and the m-health system has a star structure. Subtype-supertype relations exist among these four types as shown in Figure 5, e.g. "DAG-DAG" is a supertype of "DAG-tree". The solutions available for the supertype problem can be also applied to the subtype problem. However, due to the specialization of the subtype problem, there may exist more efficient algorithms to compute the solution for the subtype problem. We discuss the solutions for two out of these four types in the following sections: "DAG-tree" and "chain-chain". For the algorithm to find the solution of "tree-star" model, interested readers are referred to (Mei, Pawar, & Widya, 2007).

"DAG-Tree" Task Assignment

In our model of an m-health system we model communication between device resources explicitly by channel resources. This way, we can model alternative communication paths between two consecutive device nodes. Additional flexibility is achieved by allowing transmission tasks to be assigned to a communication path. For example, when considering the m-health system given in Figure 4, there are two paths connecting "d_1" and "d_5". Now if "p_1" is assigned on "d_1" and "p_2" is assigned on "d_5", modeling "t_1" explicitly and assigning it to one of the two paths allows us to distinguish these two different task assignments. However, if the model of some m-health systems has a tree structure, then there is only one unique path between any two vertices. Consequently, we can simplify the m-health application model and the m-health system model by eliminating transmission tasks and channel resources, as shown in Figure 6.

Based on this simplified model, a task assignment is a mapping of processing tasks to device resources. We can use an m by n matrix Φ, where m is the number of processing tasks and n is the number of devices, to represent one task assignment using the following interpretation: if "p_i" is assigned to device "d_p", then Φ_{ip} is "1"; otherwise Φ_{ip} is "0". An example task assignment for Figure 6 is shown in Figure 7.c. In the following sections, we present the algorithm to compute the optimal assignment matrix, Φ_{opt}.

Figure 6. Task assignment from an m-health application with a DAG structure to an m-health system with a tree structure



Formulation

We formulate the "DAG-tree" task assignment problem based on the adjacency matrix graph representation method as follows:

- **Given:** (1) A task connection matrix is a m by m matric Θ captures the topology of a task DAG and represents the amount of data units transferred per time unit between connected processing tasks as shown in Figure 7(a): If there is a directed arc from p_i to p_j, then Θ_{ij} in the matrix contains the data units sent from p_i to p_j every time unit. If there is no directed arc from p_i to p_j, then Θ_{ij} is "0". (2) A n by n device link matrix K captures the topology of a device tree as shown in Figure 7(b): If there is a directed arc from d_p to d_q, then K_{pq} is "1"; otherwise, it is "0". (3) A performance measure Ω is a measure for a particular task assignment corresponds to the assignment matrix Φ.
- **Goal:** To find the optimal task assignment matrix Φ_{opt} among all possible task assign-

Figure 7. Relative to the task assignment problem given in Figure 6: (a) A data throughput matrix Θ, (b) a device link matrix K, and (c) an assignment matrix Φ

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
p_1	0	0	40	0	0	0	0	0	0	0
p_2	0	0	0	50	0	0	0	0	0	0
p_3	0	0	0	0	0	20	20	0	0	0
p_4	0	0	0	0	30	0	0	0	0	0
p_5	0	0	0	0	0	0	0	20	0	0
p_6	0	0	0	0	0	0	0	0	15	0
p_7	0	0	0	0	0	0	0	0	10	0
p_8	0	0	0	0	0	0	0	0	20	0
p_9	0	0	0	0	0	0	0	0	0	5
p_{10}	0	0	0	0	0	0	0	0	0	0

(a)

	d_1	d_2	d_3	d_4
d_1	1	0	1	0
d_2	0	1	0	1
d_3	0	0	1	1
d_4	0	0	0	1

(b)

	d_1	d_2	d_3	d_4
p_1	1	0	0	0
p_2	0	1	0	0
p_3	1	0	0	0
p_4	0	1	0	0
p_5	0	0	0	1
p_6	0	0	1	0
p_7	0	0	0	1
p_8	0	0	0	1
p_9	0	0	0	1
p_{10}	0	0	0	1

(c)

ments $\{\Phi_1, \Phi_2, ...\}$ such that $\forall_{i\in[1,2,...]}\ \Omega(\Phi_{opt}) \geq \Omega(\Phi_i)$.

- **Subject to:** *Local constraint* and *reachability constraint* (c.f. Section 3.2).

Validity of Assignment

A valid assignment Φ must satisfy both local and reachability constraints. To verify the validity of an assignment Φ according to the local constraint is relatively simple: we can go through every processing task to check whether the assigned device can support it or not. For example, due to the local constraint, p_2 has to be assigned to d_2, therefore all the assignment matrices that violate this are considered non-valid.

To verify the validity of an assignment according to the reachability constraint, we need to ascertain that the assignment is such that the data transmission between two device resources can be realized by a communication path between the respective device resources. This can be computed as follows. Given an assignment Φ, we calculate the amount of data unit transmitted between all pairs of devices by calculating $\Phi^T\Theta\Phi$. We name the resulting node data transmission matrix Λ, where Λ_{pq} indicates the number of data units sent from d_p to d_q per time unit as shown in Figure 8(a).

The reachability of devices can be represented by the transitive closure of K, denoted K^*, as shown in Figure 8(b). If there is a direct path from d_p to d_q, then K^*_{pq} is "1"; otherwise, K^*_{pq} is "0". Given an assignment Φ, if the following rule holds true, then Φ is valid according to the reachability constraint:

$$\forall_{p,\,q\,\in\,\{1,n\}}\ \Lambda_{pq} \neq 0 \Rightarrow K^*_{pq} = 1$$

The rational behind this is that if there is a required data transmission from device d_p to device d_q as indicated in Λ, then d_q has to be reachable from d_p in order to make Φ as a valid assignment matrix.

A Sketch of the Algorithm to Search for the Optimal Assignment

The DAG-tree assignment problem in its general form is NP-hard. So, for small problems, small in terms of the number of tasks and number of devices resources, an exhaustive search may still be viable. Such an exhaustive search comprises the following steps:

1. Generate a candidate Φ (for instance by permutation).
2. Check if Φ is valid based on the validity verification method presented in Section 4.2.2.
3. If Φ is valid, calculate $\Omega(\Phi)$; otherwise, go back to step 1.
4. If $\Omega(\Phi)$ is better than previous best one, record this new $\Omega(\Phi)$ and the corresponding Φ as temporary Φ_{opt}.

Figure 8. (a) the node data transmission matrix Λ, (b) the transitive closure K. Both are derived from the matrices shown in Figure 7*

	d_1	d_2	d_3	d_4
d_1	40	0	20	20
d_2	0	50	0	30
d_3	0	0	0	15
d_4	0	0	0	55

(a)

	d_1	d_2	d_3	d_4
d_1	1	0	1	1
d_2	0	1	0	1
d_3	0	0	1	1
d_4	0	0	0	1

(b)

5. Go back to step 1 until no further candidate Φ.

Although this exhaustive approach is able to find the optimal assignment in the "DAG-tree" task assignment problem, it is far from efficient since every task assignment needs to be checked. Nevertheless, we argue its advantage in the low implementation complexity: the matrix representations of task DAG and device tree allow a structural way to represent the application and platform information; the proposed verification procedures for local constraints and reachability constraints can be supported well by a number of matrix computation tools or libraries.

"Chain-Chain" Task Assignment

When both the m-health application and system form a chain as illustrated in Figure 9 and the goal of the task assignment is to find an optimal assignment with minimal end-to-end delay (or more generally the performance evaluation function is monotonic increasing), the optimal task assignment can be found in polynomial-time. This section explains an algorithm achieving this.

Problem Formulation

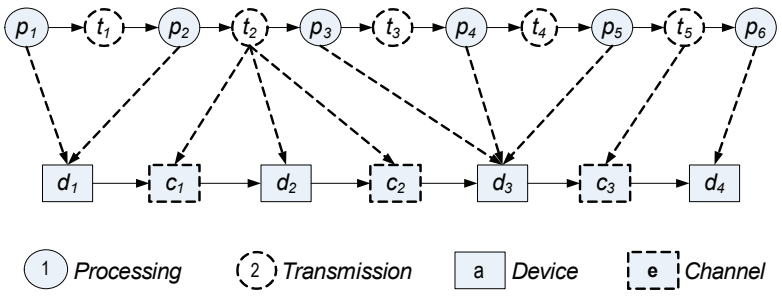
Starting with the general model, an example of a "Chain-chain" task assignment is given in Figure 9. Once two adjacent processing tasks are assigned to device resources, the transmission task in between can only be assigned to the communication path connecting the two host devices (this may potentially be the empty path). Given the task assignment problem with 6 processing tasks and 4 devices as shown in Figure 9 and reusing the definitions given earlier in Table 2, the end-to-end delay can be computed as follows

$$\Omega^{de} = de^{pr}_{p1,d1} + de^{pr}_{p2,d1} + de^{pr}_{p3,d3} + de^{pr}_{p4,d3} + de^{pr}_{p5,d3} + de^{pr}_{p6,d4} + de^{tr}_{t1,d1} + de^{tr}_{t2,c1} + de^{tr}_{t2,d2} + de^{tr}_{t2,c2} + de^{tr}_{t3,d3} + de^{tr}_{t4,d3} + de^{tr}_{t5,c3}$$

However, similar to the "DAG-tree" assignment problem, it is sufficient to model the assignment of processing tasks onto device resources: We formulate the problem of optimally assigning a processing task chain onto a device chain as:

- **Given:** A directed task chain containing m processing tasks, $\{p_i, p_2, \ldots, p_m\}$ connecting in a chain fashion (with m-1 transmission tasks). A directed device resource chain containing n device resources, $\{d_j, d_2, \ldots, d_n\}$ (with n-1 channels). A performance measure on end-to-end delay of a particular assignment of task across devices, Ω^{de}.
- **Goal:** To find an assignment function Ψ_α among all possible assignments Ψ: $\{p_i\} \rightarrow \{d_j\}$ that $\Omega^{de}(\Psi_\alpha)$ is minimized. To find the

Figure 9. An example of assigning a directed task chain containing 6 tasks onto a directed resource chain containing 4 devices



task assignment with minimal end-to-end delay Ψ_{opt} among all possible task assignments $\{\Psi_1, \Psi_2, ...\}$, hence: $\forall_{i\in[1,2,...]}\ \Omega(\Psi_{opt}) \leq \Omega(\Psi_i)$.

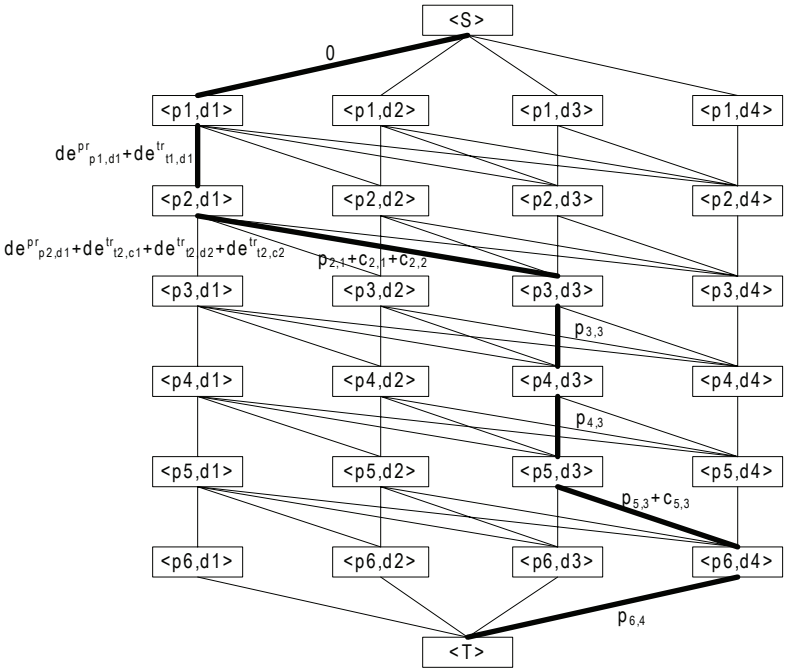
- **Subject to:** The local constraint, type constraint and reachability constraint.

Solution

We first build a layered assignment graph consisting *mn+2* nodes (Figure 10). In this graph, each row (excluding nodes <S> and <T>) corresponds to a processing task and each column corresponds to a device. The label $<p_i,d_j>$ on each node corresponds to a possible (i.e. satisfying the local constraint) assignment of processing task "p_i" to device "d_j". A node labeled $<p_i,d_j>$ is connected by arcs to all nodes $<p_{i+1},d_j>$, $<p_{i+1},d_{j+1}>$ … $<p_{i+1},d_n>$ in the layer below. All nodes in the first (last, respectively) layer are connected to the node <S> (<T>, respectively). Therefore, any directed path connecting nodes <S> to <T> corresponds to an assignment of processing tasks to devices fulfilling the type constraint and reachability constraint (e.g. the thick path in Figure 10 which corresponds to the assignment shown in Figure 9).

Each arc of this layered assignment graph is then labeled with a weight representing the sum of processing delays and transmission delays such that: (1) the arcs connecting node <S> to node $<p_1,d_1>$ till $<p_1,d_n>$ have weight *0*; (2) In layer *i* (except for the last layer), the arc connecting node $<p_i,d_j>$ to node <pi+1,dk> has a weight equal to the sum of processing delay (the delay for "p_i" to process one frame at device resource "d_j", i.e $de^{pr}_{pi,dj}$) and the transmission delay (the delay caused the transmission over the channel connecting di and dk);. (3) For the last layer, each arc connecting node $<p_m,d_j>$ to node <T> the weight equals the processing delay (for the delay for "p_m" to process one frame

Figure 10. The 6-task-4-device assignment graph. The thick path corresponds to the assignment illustrated in Figure 9



at device "d_j", hence $de^{pr}_{pm,dj}$). As an illustration, the weights associated with the thick edges are shown in Figure 10.

In the last step, by applying a shortest-path search algorithm (e.g. Dijkstra's algorithm), we can identify a path connecting <S> and <T> in this weighted graph that corresponds to the optimal assignment Ψ_{opt}. The space complexity (defined as the number of nodes in the assignment graph) of this method is $O(mn)$. Thus the time complexity of shortest-path search step is $O(m^2n^2)$ if the Dijkstra algorithm is used. If a labeling method as proposed by Bokhari (Bokhari, 1988) is used for search, the time complexity can be reduced to $O(mn)$.

Performance Analysis

We implemented a Java prototype of the proposed assignment. We recorded the CPU time of two steps in this method: the assignment graph construction (includes weighting) and shortest-path search. For the "search" step, we used an open source library[f] which implemented Dijkstra algorithm. The program was tested on a Windows XP machine with Intel Pentium 2.4G CPU and 1.5G RAM. 10 different pairs of m and n, e.g. (20,10), (40,10), … (100,20), were tested and the number of assignment graph nodes ranges from 200 to 2000.

TASK REDISTRIBUTION INFRASTRUCTURE

As presented earlier (Mei et al., 2007), we design the bio-signal processing tasks as software components, named BSPUs (Bio-Signal Processing Units), whose execution location may vary during the life-time of the m-health application. A

Figure 11. The CPU time of the proposed optimal assignment algorithm for task-chain to device-chain assignment

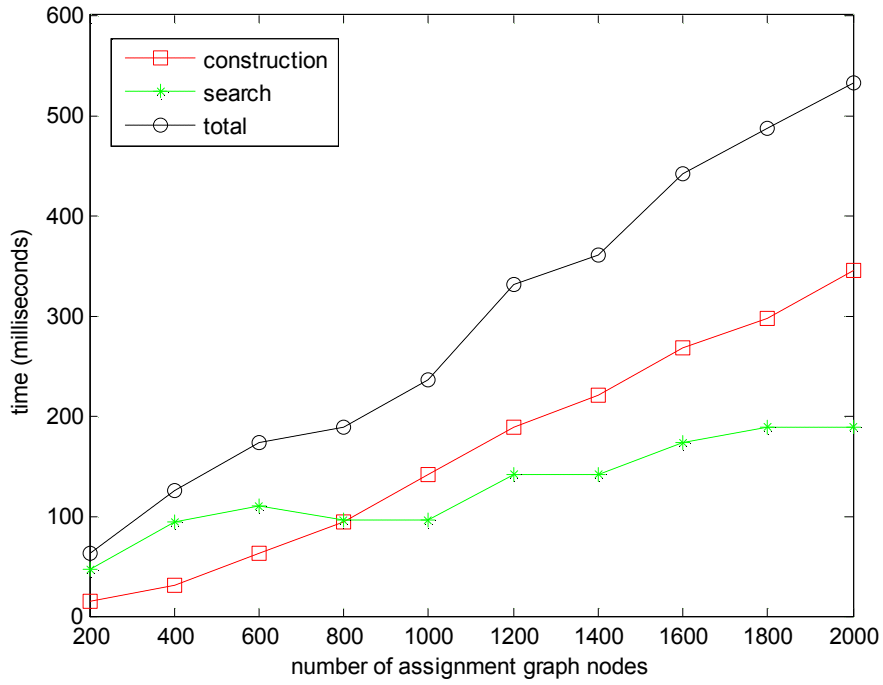
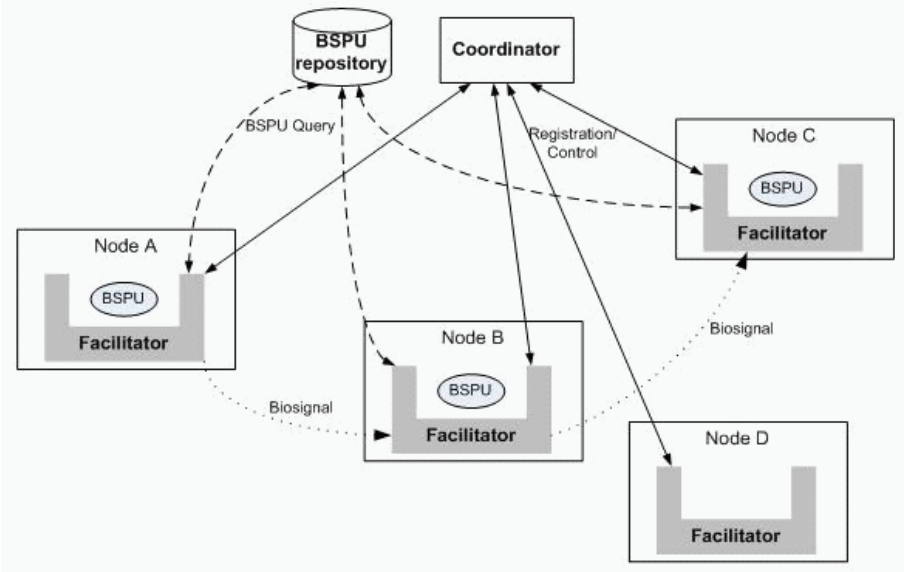


Figure 12. Architectural components of BSPU redistribution infrastructure



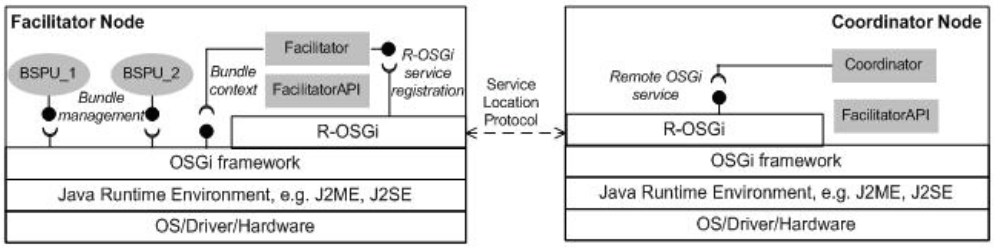
possible architecture of our proposed MADE middleware consists of a Coordinator, a BSPU Repository and a set of Facilitators (Figure 12). The Coordinator (which hosts the assignment algorithm), can make decisions to reconfigure the m-health system, e.g. redistribute some BSPUs across the Facilitators. The reconfiguration can be either stateful (transferring state information) or stateless (discarding state information), depending on application specific requirements. Each node has a Facilitator to manage the resident BSPUs. It can be instructed by the Coordinator to execute the BSPU configuration commands. The BSPU Repository stores BSPU implementations that can be provided on request.

To realize the Facilitator, we have adopted the OSGi technology[g], which is a service oriented, component-based software container that, as apposed to other container platforms such as J2EE, standardizes the lifecycle management of components and services. In addition, OSGi technology is lightweight and can run on for instance mobile devices such as Smart Phones and PDAs. For networked communication between OSGi containers we have used the so called R-OSGi[h] service.

Each BSPU is implemented as a single OSGi bundle, which we name a BSPU bundle. This means that we can benefit from the life-cycle management facilities in the OSGi framework: the bundled BSPU can be installed, started, stopped and uninstalled by OSGi bundle management commands. Both the Facilitator and the Coordinator are also implemented as OSGi bundles. As shown in Figure 13, the Coordinator bundle can connect to various remote Facilitators using the "R-OSGi" service. At the moment, we adopted a "break-before-make" strategy in the implementation: the current configuration is terminated first and then new BSPUs are distributed and formed into a new configuration. In addition, the reconfiguration is assumed stateless.

To test the performance of the task redistribution infrastructure, we built an experimental system containing 4 computers (PC1, PC2, PC3 and PC4) within the same LAN. Each of PC1 and PC2 hosts one Facilitator, PC3 is the Coordinator node and PC4 is the BSPU reposi-

Figure 13. Coordinator node connecting Facilitator node through R-OSGi service interface



tory. The measured application interruption time resulted by the "break-before-make" BSPU redistribution ranges from tens to hundreds of milliseconds depending on the code size of the BSPU and network conditions. Further experiments are planned on mobile systems with the aim to analyze the adaptation's agility according to the targeted scenario.

DISCUSSION

With the development of mobile and high capacity personal computing devices, miniature wearable sensors and ever improving wireless communication infrastructures, m-health is becoming a realistic prospect from the technical point of view. However, challenges remain for providing timely and adequate services in real world m-health applications (Jones, Incardona, Tristram, Virtuoso, & Lymberis, 2006). We argue in this chapter that context awareness should be an essential feature in the design and implementation of m-health systems. Typical scenarios that we investigate address the remote monitoring and treatment of patients. In particular, we have addressed an epileptic seizure detection application based on m-health systems. Patient monitoring sessions are often long lasting, in extreme cases it may be 24/7. To cope with the dynamics of the ICT infrastructure under consideration, we proposed a task redistribution based middleware, MADE, and investigated several assignment algorithms to obtain the optimal system configuration.

Our research on task assignment algorithm benefited from earlier research in the area of parallel and distributed computing (Norman & Thanisch, 1993). In particular, we have adopted graph-based techniques to model the assignment problem. We have identified three different categories of constraints (type, local and reachability constraints) that allow us to build realistic models of the assignment problem. Because of the generic nature of our model, it may be very well applicable to distributed stream processing in general, for instance (Amini, Jain, Sehgal, Silber, & Verscheure, 2006; Gu & Nahrstedt, 2006; Pietzuch et al., 2006; Xing, Hwang, Çetintemel, & Zdonik, 2006).

CONCLUSION

In this chapter, we propose a task redistribution based adaptation middleware (MADE). It consists of a task assignment decision mechanism and a task redistribution infrastructure. The task assignment decision mechanism models the task assignment as a graph mapping problem and searches for the optimal assignment given the latest context information. In particular, we identify three assignment constraints in the mobile healthcare system: the type constraint distinguishing between computation and communication, the local constraint capturing the

computation heterogeneity and the reachability constraint capturing the communication heterogeneity. Although the problem in its general form is NP-hard, some cases with specific topologies can be tackled efficiently in polynomial time. We illustrate a general approach to solving "DAG-tree" problems and a polynomial-time algorithm for "chain-chain" problems. Once a new assignment is found, the member tasks are distributed accordingly by the redistribution infrastructure. A prototype implementation based on the OSGi framework, which has been used to validate the task redistribution infrastructure, is also reported.

The future research agenda for improving the MADE middleware may consist of: (1) applying various techniques, e.g. A* and branch and bound, to design more efficient task assignment algorithms, (2) proposing performance metrics (including reconfiguration time, power consumed by reconfiguration) on the redistribution infrastructure to evaluate the reconfiguration, and (3) experimenting with the MADE middleware in a real world mobile healthcare system to evaluate the feasibility of task redistribution based adaptation.

REFERENCES

Amini, L., Jain, N., Sehgal, A., Silber, J., & Verscheure, O. (2006). *Adaptive Control of Extreme-scale Stream Processing Systems*. Paper presented at the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06).

Badrinath, B., Fox, A., Kleinrock, L., Popek, G., Reiher, P., & Satyanarayanan, M. (2000). A Conceptual Framework for Network and Client Adaptation. *Mobile Networks and Applications (MONET)*.

Bokhari, S. H. (1988). Partitioning problems in parallel, pipelined, and distributed computing. *IEEE Transactions on Computers, 37*(1), 48-57.

Catalan, M., Gomez, C., Viamonte, D., Paradells, J., Calveras, A., & Barcelo, F. (2005). TCP/IP analysis and optimization over a precommercial live UMTS network. *Wireless Communications and Networking Conference, 2005 IEEE, 3*.

Franken, L. (1996). *Quality of Service Management: a Model-Based Approach*.

Gu, X., & Nahrstedt, K. (2006). On Composing Stream Applications in Peer-to-Peer Environments. *Parallel and Distributed Systems, IEEE Transactions on, 17*(8), 824-837.

Halteren, A. v., Bults, R., Wac, K., Konstantas, D., & Widya, I. (2004). Mobile Patient Monitoring: The MobiHealth System. *The Journal of Information Technology in Healthcare, 2*(5).

Hung, K., & Yuan-Ting, Z. (2003). Implementation of a WAP-based telemedicine system for patient monitoring. *Information Technology in Biomedicine, IEEE Transactions on, 7*(2), 101.

Jones, V., Incardona, F., Tristram, C., Virtuoso, S., & Lymberis, A. (2006). Future challenges and recommendations. In R. S. H. Istepanian, S. Laxminarayan & C. S. Pattichis (Eds.), *M-health Emerging Mobile Health Systems* (pp. 267-270): Springer.

Kafil, M., & Ahmad, I. (1998). Optimal task assignment in heterogeneous distributed computing systems. *Concurrency, IEEE [see also IEEE Parallel & Distributed Technology], 6*(3), 42.

Lee, C.-H., & Shin, K. G. (1997). Optimal task assignment in homogeneous networks. *Parallel and Distributed Systems, IEEE Transactions on, 8*(2), 119-129.

Lo, V. M. (1988). Heuristic algorithms for task assignment in distributed systems. *Computers, IEEE Transactions on, 37*(11), 1384.

Ma, Y.-C., Chen, T.-F., & Chung, C.-P. (2004). Branch-and-bound task allocation with task clustering-based pruning. *Journal of Parallel and Distributed Computing, 64*(11), 1223 - 1240.

Mei, H., Pawar, P., & Widya, I. (2007, March 26 2007). *Optimal Assignment of a Tree-Structured Context Reasoning Procedure onto a Host-Satellites System*. Paper presented at the 16th Heterogeneity in Computing Workshop (HCW 2007), Long Beach, California.

Mei, H., Widya, I., Broens, T., Pawar, P., Halteren, A. v., Shishkov, B., et al. (2007). *A Framework for Smart Distribution of Bio-signal Processing Units in M-health*. Paper presented at the 2nd International Conference on Software and Data Technologies (ICSOFT 2007), Barcelona, Spain.

Muppala, J. K., Fricks, R. M., & Trivedi, K. S. (2000). Techniques for Dependability Evaluation. In W. Grasman (Ed.), *Computational Probability* (pp. 445-480): Kluwer Academic Publishers.

Norman, M. G., & Thanisch, P. (1993). Models of machines and computation for mapping in multicomputers. *ACM Computing Surveys, 25*(3), 263-302.

Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., & Seltzer, M. (2006). *Network-Aware Operator Placement for Stream-Processing Systems*. Paper presented at the 22nd International Conference on Data Engineering (ICDE'06).

Rahmati, A., & Zhong, L. (2007). *Context-for-Wireless: Context-Sensitive Energy-Efficient Wireless Data Transfer*. Paper presented at the 5th international conference on Mobile systems, applications and services (Mobisys).

Rasid, M. F. A., & Woodward, B. (2005). Bluetooth telemedicine Processor for multichannel biomedical signal transmission via mobile cellular networks. *Information Technology in Biomedicine, IEEE Transactions on, 9*(1), 35.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *IEEE Personal Communications, 8*(4), 10-17.

TelemedicineAlliance. (2004). Telemedicine 2010: Visions for a Personal Medical Network. from http://www.esa.int/esapub/br/br229/br229.pdf

Tönis, T., Hermens, H. J., & Vollenbroek-Hutten, M. (2006). *Context aware algorithm for discriminating stress and physical activity versus epilepsy*: AWARENESS deliverables (D4.18).

Ucar, B., Aykanat, C., Kaya, K., & Ikinci, M. (2006). Task assignment in heterogeneous computing systems. *Journal of Parallel and Distributed Computing, 66*(1).

Widya, I., Beijnum, B.-J. v., & Salden, A. (2006). *QoC-based Optimization of End-to-End M-Health Data Delivery Services*. Paper presented at the 14th IEEE International Workshop on Quality of Service (IWQoS).

Xiao, Y., Rosdahl, J., Center, S. L. D., Linear, M., & Draper, U. T. (2002). Throughput and delay limits of IEEE 802.11. *Communications Letters, IEEE, 6*(8), 355-357.

Xing, Y., Hwang, J.-H., Çetintemel, U., & Zdonik, S. B. (2006). *Providing Resiliency to Load Variations in Distributed Stream Processing*. Paper presented at the 32nd International Conference on Very Large Data Bases.

Yuan-Hsiang, L., Jan, I. C., Ko, P. C. I., Yen-Yu, C., Jau-Min, W., & Gwo-Jen, J. (2004). A wireless PDA-based physiological monitoring system for patient transport. *Information Technology in Biomedicine, IEEE Transactions on, 8*(4), 439.

END NOTES

1 http://www.mobihealth.org/

2 http://www.healthservice24.com/

3 http://awareness.freeband.nl/

4 The value can be zero, i.e. meaning it is not possible to run a task on certain resource due to a violation of a hard QoS requirement,

e.g. device CPU type or channel's bandwidth cannot support the to-be-assigned task.

5 If a device has a fixed line power supply, then its e^{to} has a value of $+\infty$.

6 http://rollerjm.free.fr/pro/graphs.html

7 http://www.osgi.org

8 http://r-osgi.sourceforge.net/