Automatic Computing Resource Awareness in Resource Managers for Cognitive Radios

Ismael Gomez, Vuk Marojevic, Antoni Gelonch
Dept. of Signal Theory and Communications, Polytechnic University of Catalonia,
Av. Canal Olímpic s/n, 08860 Castelldefels, Spain
ismael.gomez@tsc.upc.edu
marojevic@tsc.upc.edu
antoni@tsc.upc.edu

***Abstract*— With all the potential flexibility of software-defined radios, the flexibility of SDR terminals is currently limited to design time flexibility. The capacity of the platform in terms of processing resources and internal bandwidths is dimensioned for the range of supported functionalities. In a platform-independent design scenario, resource managers play the role of matching waveform demands with platform capabilities. Predicting the task execution times has been studied in grid and distributed computing contexts with different objectives and assumptions. Given the dynamic nature of waveform demands as a function of the radio environment, an accurate characterization of the consumed resources can increase the efficiency of resource management strategies. In the SDR context, this efficiency translates to less energy consumption and higher resource utilization. Based on our experience acquired during the development of an SDR execution environment, this work presents the metrics that are needed by computing resource managers.**

I. Introduction

Software-defined radio (SDR) has been introduced as an alternative to proprietary, inflexible, and non-scalable radio nodes. It is widely accepted that SDR is a technology enabler for cognitive radio. SDR terminals are theoretically able to dynamically change transmitting signal as a function of present (and past) radio conditions. Mitola's cognitive cycle [1] assumes a continuous evolution of radio channel conditions, e.g. interference and noise levels, time, space and spectrum utilization. Practically, the radio terminals' functionalities or supported radio access technologies (RATs) are, however, defined at design time. This means that the computing capabilities are limited to a small set of RATs.

Following the desktop computing history, it is reasonable to consider future radio terminals, where the functionalities are designed without detailed knowledge about the computing platform. Another scenario envisages shared-resource base stations: infrastructure operators rent hardware resources (digital signal processors, converters, RF frontends, and so forth) to network operators in a pay-per-use fashion [2]. In such a scenario, abstraction layers, virtualization mechanisms and resource managers (RMs) play the role of matching the waveform demands to the platform capacities optimizing some metric.

In this paper we show how available resources can be discovered and their utilization monitored. The management algorithms and their performance evaluation are out of the scope of this work. We present our resource awareness proposal in a case study assuming our open-source middleware ALOE (abstraction layer and operating environment [3]). Finally, a novel middleware interface (API) for radio resource management algorithms is also presented. It can provide hardware resource occupations in real-time as a function of several variables.

II. Related Work

The execution times of processes are highly dynamic as they depend on many system parameters and circumstances. The execution time prediction of processes or algorithms is a topic which has been studied in the field of grid and distributed computing. Predictions can be *static* or *dynamic* [2].

Static methods use off-line profiling or code analysis techniques to obtain, given a known processor model, an estimation of the execution time. Conversely, dynamic methods characterize the behaviour of the algorithm in terms of statistical regression methods. During the execution of the process, measurements are taken of the execution profile. With these samples, the execution time under different circumstances (system loads, processor clocks, etc.) of the program is predicted.

To the best of our knowledge, all frameworks for task execution time prediction address grid or distributed computing applications. System conditions in these scenarios are different than in signal processing:

- arrival times are dynamic but periods are much longer,
- processors perform background and non real-time applications,
- networks are more complex and have longer latencies,
- energy consumption is not a constraint,
- management complexity is not such an important issue,
- application model is arbitrary and more complex, and so on [4].

Another important characteristic of SDR platforms is their heterogeneity. Grid computing prediction assumes this [5], but for some specific platform characteristics, such as processor clock, memory size, and system load. In SDR, platform heterogeneity implies different processor architectures, instruction-sets, levels of parallelism, and so forth.

III. Resource Awareness

A. Platform Resources Awareness

Typical SDR platforms assume a set of heterogeneous processing elements (general-purpose processors, digital signal processors, dynamic reconfigurable areas, etc), which are either integrated on a single silicon die or not. Devices are usually interconnected using dedicated interfaces or routing elements (Network On-Chip). Our computing resource manager model considers processing devices and their inter-processor communication bandwidths [6]. A vector of *N* elements represents each processor capabilities, whereas a matrix of *NxN* elements specifies the network connectivity and bandwidths. The RM automatically discovers vector *C* and matrix *B*.

Before measuring capacities or bandwidths, it is necessary to identify the network architecture, which involves (1) obtaining the number of processors, (2) uniquely identifying each element, and (3) identifying the interfaces between them.

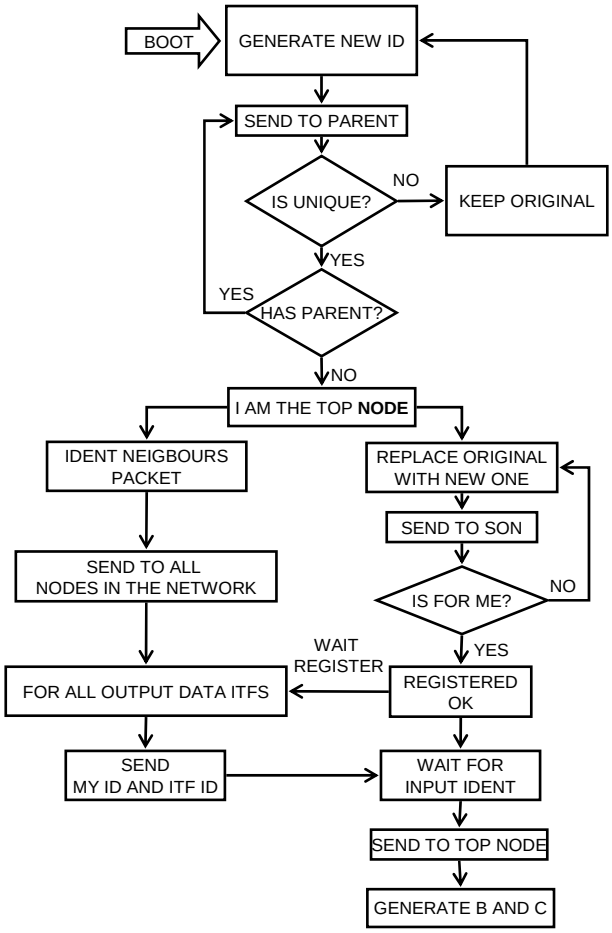


Fig. 1. Network identification algorithm

This procedure is similar to the problem of automatic addressing in computer networks for which several protocols have been studied. In computer networks, addressing and routing is managed by higher layer network protocols (e.g. IP, DHCP), which usually assume a shared channel. The interfaces of SDR platforms are heterogeneous and lack of a unique protocol for this purpose. What is more, such protocols usually introduce intolerable latencies. Simplifications arise if we constrain the network architecture of the control plane to be hierarchical. In this case, a single element (at the topmost level without a parent node) centralizes all the information of all elements in the network. A hierarchical control-plane does not constraint the data-plane network architecture (which uses different physical or logical interfaces) or the application performance.

Fig. 1 shows our algorithm proposal, which uniquely identifies a network of processors and their connectivity in a plug-and-play fashion. When a new processor is attached to the network, it obtains a unique id from the top parent node and starts a periodic report of its status information (not represented in the figure). The top node sends a packet to all the nodes requesting to send their unique id and interface id to their neighbours. Data is collected by the top node and the modelling matrices, *B* and *C*, are generated. The same procedure is initiated when a processor fails to periodically communicate its status (not represented in the figure).

B. Waveform Consumption Awareness

Before discussing the resource consumption, it is necessary to specify a general waveform model. A waveform is defined in terms of a graph of *M* nodes. Nodes indicate computations and arcs communications links. Nodes are executed periodically and synchronously (in the sense that their behaviour is data-independent). On each invocation, a set of computing resources are consumed, such as CPU time, energy, input/output communications and silicon area. For simplicity, we consider time and energy only. Then, invocation *k* of component *m* in processor P_n consumes a set of resources, i.e.:

$$r_m(P_n,k)=\begin{pmatrix} T_m(P_n,k) \\ E_m(P_n,k) \end{pmatrix} \qquad (1),$$

where *T* is resource “time” and *E* is resource “energy”.

Monitoring the processes’ CPU utilization is a challenging task. Reference [7] compares several intrusive and non-intrusive methods in PC-Linux platforms. Clearly, intrusive methods exhibit lower variance (e.g. *gettimeofday()* vs. *getrsuage()*) but need to modify application code. In a middleware context, intrusive methods are naturally included in the API (section IV.B).

Energy consumption measurements need specific device support. It is measured in Joules per invocation. It can be assumed that processing devices provide system calls or specific registers to obtain instantaneous energy dissipation. In the cases where energy cannot be obtained, it is estimated as mean power consumption times the execution time.

Consumed resources are deterministic. The time spent running a process, for example, is a function of the instructions

that are executed plus the cache or memory wait states. As the program complexity grows, also does the variability of the number of instructions or branches (e.g. turbo-decoding, ray-search, etc.). Moreover, under high processor loads, the underlying software (OS, middleware, or the like) exhibits higher variances in the execution time. In general, the higher degree of abstraction, the higher the variability in the execution time. Therefore, a more general model assumes random contributions [5], for example,

$$T_m(P_n,k) = T_m(P_n) + z_m(k) \quad , \quad (2)$$

where $z_m(k)$ is a zero-mean uncorrelated random process and $T_m(P_n)$ is deterministic. The random effect is observed in Fig. 2. It shows a histogram of $T_m(P_n,k)$ for UTRAN components, captured with the ALOE tools. The variance of $z_m(k)$ depends on the program (component) and the measurement method. CHSIM adds random Gaussian noise to the samples, implemented with *rand()* system calls (and others). These functions execute a different number of instructions at each call, which increases the variance. The INTERLEAVER, on the other hand, always executes the same number of instructions. Its randomness might come from cache misses, measurement and other system specific influences. The TURBODECODER histogram shows two peaks corresponding to the probability of doing 1 or 2 iterations (the decoder has an early-stopping mechanism; therefore the iterations are function of the amount of errors in the received codeword). In this case, the average of the execution time changes with the realization of the process, having two states with different probability. The same effect is observed in the CHSIM component although in this case caused by the operating system instead of the received data or signal quality.

Considering the aforementioned statistical nature of the resource consumption, the system has to decide which value represents the set of possible values. A very conservative strategy considers maximum resource consumptions. An example is worst-case execution time (WCET) analysis, which considerably overestimates the problem, implying poor resource utilization. On the opposite side, arithmetic mean causes too much missed deadlines. Missing a deadline means losing samples at the digital-to-analog-converter (DAC) input, which increases the bit-error rate (BER). As opposed to classical hard real-time systems, wireless communications already assume a certain BER; therefore, it is reasonable to choose a value so that the BER is below the desired threshold.

At iteration *K*, the system estimates resource consumption as a function of past measurements (component index *m* is omitted):

$$\hat{T}(P_n,K) = f(T(P_n,k)) \quad , \quad 0 \le k < K \qquad (3)$$

so that at any time, the probability to miss a deadline is upper-bounded by

$$P\left(T(P_n,k) > \hat{T}(P_n,K)\right) < \varepsilon \quad , \quad 0 < k < \infty \quad ,(4)$$

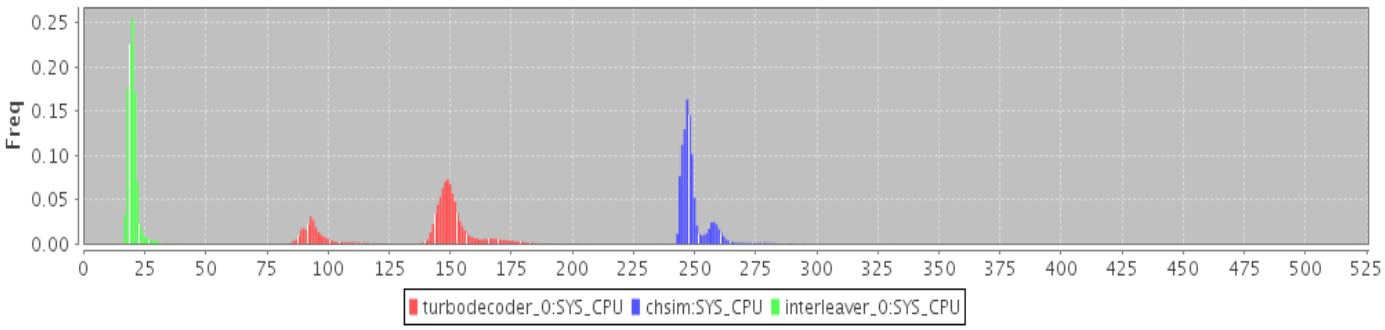
assuming a strictly stationary process and known probability density function (*pdf*).

Therefore, the larger the variance with respect to the mean, the more extra resources will be allocated for a given BER. In general, signal processing algorithms exhibit large means and low variances. Small components with low consumptions, however, make system function calls an equal number of times. Therefore, the variance/mean relation increases. When all consumptions are aggregated, small components are dominant. The level of granularity, therefore, influences the total effect of randomness, which reduces the mean resource utilization (or increases the BER).

This effect is mitigated if fewer system calls are used or if their behaviour is more deterministic (e.g., through real-time OS extensions). If each component is implemented as a separate process, the scheduler complexity must also be constant. Nevertheless, many complex algorithms may still show random or asynchronous behaviour. Moreover, flexible NoC based multiprocessor chips also suffer from random communication latencies under high network loads [8]; this reinforces the utility of an accurate statistical modelling of the resources consumed by waveforms.

C. Resource Manager Feedback

From the previous sections, we observe that the size of the parameter space is very large when the number of processors and tasks is high. Given all the captured data, it is impractical (from a computationally point of view) to exhaustively search for an optimum resource allocation. A more feasible strategy is "*divide and conquer*": several RMs coexist in the scenario, each addressing a reduced number of parameters. The struc-



ture can be layered, hierarchical, or cooperative. In this work, we consider a layered structure with two RMs: a lower-layer RM addressing computational efficiency issues (admission control and task mapping/scheduling) and a higher-layer RM addressing the interactions with the radio environment. This section focuses on the lower-layer RM.

At system start-up, instance number *k=0*, the available information about the component demands is based on prior, manually defined information. Static execution time prediction methods or cycle-accurate processor simulators can be used to improve the accuracy. Initial energy consumption can be estimated as the mean device power consumption times the execution time. The resource demands for a component to be executed by processor P_n at time 0 would then be:

$$r(P_n,0)=\begin{pmatrix} T^0 \\ E^0 \end{pmatrix} \quad , \quad 0 \le n < N-1 \ . \ (5)$$

After the first mapping of the given application to processor P_n, the first measurements are available and the resource utilization can be computed. Let the system run for *K*=100 cycles. At time stamp 100,

$$r(P_n,100)=\begin{pmatrix} \hat{T}(P_n,100) \\ \hat{E}(P_n,100) \end{pmatrix} , \ (6)$$

better approximates the real computing demands.

The more time the system runs, the more accurate will be the resource occupation estimation. Several regression methods can be used to predict the execution time of the same application module on another processor. The level of accuracy will usually incur in more or less predictor complexity. The prediction will be a function of the measurements performed at the other processors. For example, the execution time on processor with available real data for processor is a combination or function of real measurements and the characteristics of both processors:

$$\hat{r}(P_j,k;P_n)=f\left(r(P_n,k),C_n,C_j\right). \qquad (7)$$

The prediction accuracy increases with processor architecture homogeneity, which is generally low for SDR platforms. RM actions (mappings) with predicted data can be incorrect in several cases implying poor resource utilization or missed deadlines. Therefore, the more real data is available to the RM, the better will be its management actions.

IV. Case Study: The ALOE Middleware

The Abstraction Layer and Operating Environment (ALOE) is an open source SDR framework with real-time computing resource management capabilities [3]. It is specifically designed for signal processing applications, that is, data flow based processing. The middleware targets general-purpose processors (GPPs) and digital signal processors (DSPs) through a lightweight static-memory implementation in plain ANSI C. The GPP version requires an underlying POSIX OS, while the DSP version (for TI C6000 DSP) resides on top of the Texas Instruments DSP/BIOS RTOS. Hardware devices or single-threaded processors are also supported through a small VHDL version of some middleware services and APIs, currently targeted for Xilinx Virtex-5 FPGA devices.

A. Cognitive Services Architecture

ALOE distributes its functionalities and services in isolated components. Figure 3 illustrates the concept: the RM represents a group of managers, which are instantiated once for the entire platform. Another set, the sensors, are instantiated on each processing device. The RMs collect the measurements of the different sensors, which can also perform some actions. Higher-level radio resource management (RRM) or combined computing and radio resource management entities obtain data from the system through the CMDMAN component (section IV.C).

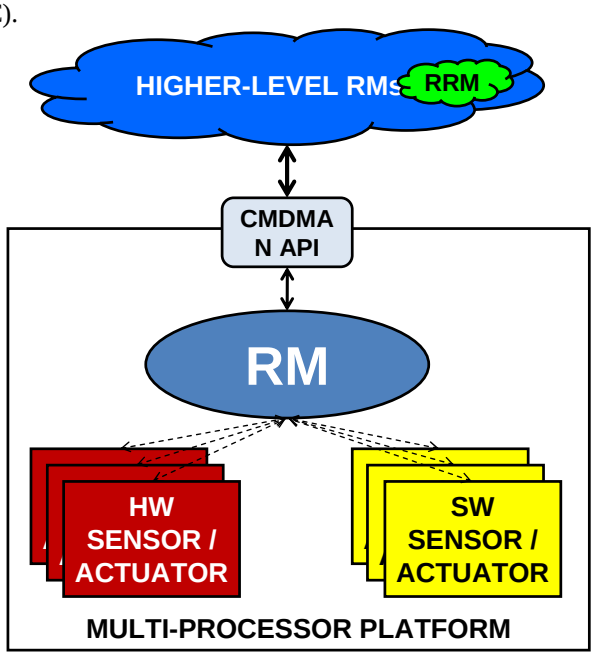


Fig. 3. Sensor / Manager Architecture

B. Computing Resource Management

ALOE supports an internal RM model de-scribed in [6]. The current model considers the resources time and bandwidth. Time is divided in discrete slots and application modules are executed in a pipelined fashion. This means that the data block produced by some module (producer module) in time slot n is not consumed before the beginning of time slot n+1 by the next module in the data flow chain (consumer module). This facilitates synchronizing the distributed data processing and ensures deterministic computing delays. The middleware controls the start and the end of the execution on each module, facilitating accurate measurements of the execution times (modules run at maximum priority and can't be preempted).

The measurement period equals the invocation period (time-slot duration), which is usually in the order of milliseconds.

Initially, resources are manually defined. After the first mapping, measurements are taken of time and bandwidth consumption and arithmetically averaged over 100 samples (100 time-slots). The execution time on the other processors is obtained with a simple predictor: after the component has been successfully mapped to processor P_n, the resources consumed on processor P_j, being C_n and C_j their capacities, measured in multiply-accumulate operations (MACs) per second (MACS), are

$$\hat{r}(P_j,k;P_n) = r(P_n,k)\frac{C_j}{C_n} \quad , \quad 0 < j < N-1 \quad (8)$$

Equation (8) assumes that processors have identical intrinsic characteristics: instruction-set, memory architecture, energy consumption per operation, and so forth. The ratio of capacities models the difference in performance. Table I shows the prediction of the resource occupation on a DSP[1] (P1) with available measurements on a PC[2] (P2). The capacities are C1=9600 MACS (8 MACs per cycle) and C2=2270 MACS (1 MAC per cycle). The difference is due to the assumption of a perfect scheduling of instructions (VLIW) in DSPs. The real performance, though, depends on the program; this reinforces the importance of the resource awareness.

TABLE I
RM PREDICTIONS IN MICROSECONDS

Component	$r(P_1)$	$\hat{r}(P_2;P_1)$	$r(P_2)$
CHSIM	252	59.58	630
INTERLEAVER	21	4.96	60
TDEC	158	37.56	480

C. ALOE CMDMAN API

The CMDMAN is a special component centralizing the interactions between higher-level control applications and ALOE. An API has been written in JAVA and C. Users can then write RRM algorithms, load, run, and stop waveforms, obtain and modify variables, get the information about the occupied and the available computing resources, and so on through a single interface.

Computing and networking resource usage is obtained as any other application's global variable, called statistic. Each component generates a new statistic value per invocation, once per time-slot. Table II lists the API functions.

TABLE II
DESCRIPTION OF THE CMDMAN API

Function	**Description**
Init	Initialize and connect to the CMDMAN.
ProcList	Get processor information and current status, including slave devices energy consumption.
WaveList	Lists the loaded waveforms.
WaveLoad	Load a new waveform.
WaveStatus	Set a new waveform status (INIT, RUN, PAUSE, etc.).
WaveInfo	Get waveform execution information: per component timestamp, mapped processor, scheduling order, etc.
StatList	Lists active statistics (includes resource consumption).
StatGet	Obtains a single sample, generated during the last time-slot.
StatSet	Sets a value of a variable for the next time-slot.
StatReport	Starts or stops a continuous report of the evolution of variables over time. Values and the instant they are generated (time-slot) are saved in a file for postprocessing.

This interface serves for the purpose of the top-level RM entity which manages the interactions between the radio and the computing environment. When the resource demands of a waveform change as a function of radio parameters (SNR, sampling frequency, channel bandwidth, etc.), new measurements are obtained and effectively characterize these changes.

V. CONCLUSIONS

This work introduced the concept of resource awareness within computing resource managers for SDR-based cognitive radios. The particular characteristics of radio environments make it difficult to employ grid or distributed computing frameworks. Future multiprocessor devices and platform-independent development approaches need to accurately characterize the platform capabilities and software performances.

These features have been added to an open-source middleware for SDR applications. An API is available to provide measurements as a function of the application's variables. This information can be used by RRM algorithms or architecture design managers [9] to calculate the cost of a selection as a function of the local computing resource occupation. The most promising feature enabled by this interface is the ability of integrating computing with the radio resource management. First, resource consumption can be considered as a cost associated to the selected RAT and introduced in the minimization cost-function. Second, given the strong influence of signal quality in the consumed computing resources in iterative algorithms, it is reasonable to integrate this cost in power-control algorithms in interference-limited channels.

ACKNOWLEDGMENT

This work was supported by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM++ (contract n. 216715).

REFERENCES

[1] T.I. C6454 DSP
[2] Intel Centrino 2.27 GHz laptop

[1] J. Mitola, “Cognitive radio: an integrated agent architecture for software defined radio,” Ph.D. dissertation, Royal Institute of Technology (KTH), Stockholm, Sweden, May 2000. Ref 1

[2] Ari Ahtiainen, Kees van Berkel, David van Kampen, Orlando Moreira, Antti Piipponen, Tommi Zetterman. “ Multiradio scheduling and resource sharing on a software defined radio computing platform”. Proceedings of the SDR ’08 Technical Conference and product Exposition, 2008 SDR Forum, Inc. Washington, D.C. on , October 26-30. 2008.

[3] ALOE Middleware Web Site http:// flexnets.upc.edu/trac/

[4] Huh, E., Welch, L. R., Shirazi, B., Tjaden, B. C., and Cavanaugh, C. 2000. “Accommodating QoS Prediction in an Adaptive Resource Management Framework“. In Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing (May 01 - 05, 2000). J. D. Rolim, Ed. Lecture Notes In Computer Science, vol. 1800. Springer-Verlag, London, 792-799.

[5] Michael A. Iverson, Füsun Özgüner, Lee Potter, "Statistical Prediction of Task Execution Times through Analytic Benchmarking for Scheduling in a Heterogeneous Environment," IEEE Transactions on Computers, vol. 48, no. 12, pp. 1374-1379, Dec. 1999

[6] V. Marojevic, X. Reves, A. Gelonch, "A Computing Resource Management Framework for Software-Defined Radios," IEEE Transactions on Computers, pp. 1399-1412, October, 2008

[7] Huh, E., Park, H., “An Efficient Statistical Profile Technique for Monitoring of Grid Applications“. KSIAM IT series vol. 6, No. 1, 51-62, 2002

[8] Santi, S.; Lin, B.; Kocarev, L.; Maggio, G.M.; Rovatti, R.; Setti, G., "On the impact of traffic statistics on quality of service for networks on chip," Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on , vol., no., pp. 2349-2352 Vol. 3, 23-26 May 2005

[9] Godard L, Moy C, Palicot J, “An Executable Meta-Model of a Hierarchical and Distributed Architecture Management for the Design of Cognitive Radio Equipments”, Annals of Telecommunications. Special Issue on Cognitive Radio, Volume 64, Numbers 7-8, 2009