https://doi.org/10.17979/spudc.9788497498081.1116

ESTRATEGIA AUTO-ADAPTATIVA BASADA EN TÉCNICAS DE INGENIERÍA DE CONTROL PARA SISTEMAS EMPOTRADOS DISTRIBUIDOS

Raúl Mario del Toro Matamoros, Rodolfo Haber
Centro de Automática y Robótica (CSIC-UPM), Ctra. Campo Real, Km 0.200, 28500 Arganda del Rey, España, {raul.deltoro, rodolfo.haber}@car.upm-csic.es

Andrei Pruteanu
Delft University of Technology, Mekelweg 4, 2628 CD Delft, Países Bajos, andrei.pruteanu@gmail.com

Resumen

Este trabajo tiene como objetivo principal el diseño e implementación de estrategias auto-adaptativas para sistemas empotrados distribuidos utilizando metodologías de diseño de la ingeniería de control. Se ha hecho frente a la problemática relacionada con los cambios en el flujo de información a través de las conexiones de la red y al exceso del consumo de energía que ello conlleva en estos sistemas. El problema se ha abordado mediante el desarrollo de una estrategia de auto-adaptación basada en técnicas de control anticipativo, la cual ha sido aplicada a la mejora del comportamiento dinámico del proceso de planificación de tareas en dispositivos distribuidos. La estrategia desarrollada ha sido evaluada satisfactoriamente de forma experimental.

Palabras Clave: sistemas empotrados distribuidos, auto-adaptación, modelado, ingeniería de control.

1 INTRODUCCIÓN

Los sistemas empotrados adaptativos distribuidos o conectados en red constituyen un área de crecimiento potencial a nivel mundial, con un número cada vez mayor de aplicaciones en diferentes dominios. Entre las aplicaciones podemos mencionar, a modo de ejemplo, los sistemas de monitoreo y control de entornos de movilidad urbana, monitorización de sistemas logísticos, aplicaciones de monitoreo y control en sistemas de transporte de alta velocidad, sistemas de extracción petrolera, sistemas de monitorización de la seguridad pública y sistemas de asistencia médica domiciliaria remota.

Sin embargo, la introducción de los sistemas empotrados adaptativos distribuidos a gran escala está dificultada por una serie de obstáculos relacionados con métodos y técnicas para su diseño, control y operación. Dado que el comportamiento dinámico de estos sistemas es considerablemente diferente al comportamiento dinámico de los sistemas centralizados, la capacidad de predicción del desempeño de estos sistemas es reducida. Esto requiere que estos sistemas posean capacidades especiales de tipo "auto-" para mantener un desempeño óptimo, tales como auto-adaptación, auto-optimización y auto-organización.

Este trabajo está enfocado hacia el desarrollo de la capacidad de auto-adaptación. De forma específica, se hace frente al problema de los cambios en el flujo de información a través de las conexiones de la red y al exceso del consumo de energía que ello conlleva. El problema se ha abordado mediante el desarrollo de una estrategia basada en técnicas de control anticipativo, la cual ha sido aplicada a la mejora del comportamiento dinámico del proceso de planificación de tareas en sistemas empotrados distribuidos de baja capacidad de cómputo.

1.1 AUTO-ADAPTACION EN SOFTWARE EMPOTRADO

Los temas de auto-adaptación en programas empotrados han sido investigados a través de diferentes áreas de la ingeniería del *software*, tales como la ingeniería de requisitos [1, 2], arquitectura de *software* [3-7], arquitecturas intermediarias [8-10], el desarrollo basado en componentes [11, 12] y los modelos guiados por objetivos [14], siendo la mayoría de estos trabajos iniciativas aisladas. En la literatura aparecen recogidas algunas técnicas para el desarrollo de *software* auto-adaptativo, habiéndose obtenido resultados mediante algunas de ellas y otras se proponen como iniciativas para trabajos futuros [5].

A pesar del carácter innovador de las iniciativas propuestas, estas tienen un carácter aislado y no proponen una metodología general para el diseño, bajo la cual se puedan tratar además los efectos que puedan tener en el sistema la introducción de mecanismos de realimentación. A lo largo de los

años, la disciplina de ingeniería de *software* realizó un fuerte énfasis en el carácter estático de la arquitectura de un sistema de *software* y descuidando en cierta medida los aspectos dinámicos. En contraste, la ingeniería de control hace un gran énfasis en los lazos de realimentación, los cuales son elevados como entidades de primera clase [15-17], al ser un elemento prácticamente imprescindible para hacer frente a cambios en el comportamiento dinámico de los sistemas y la aparición de perturbaciones externas.

En la literatura también se recogen algunos trabajos enfocados hacia el modelado de los procesos de planificación de tareas en sistemas empotrados conectados en red y la aplicación de técnicas de control para dotar a estos sistemas de capacidades adaptativas. Sólo por mencionar algunos trabajos previos, Cervin et al. [18] han desarrollado un sistema de control realimentado con mecanismos de anticipación para el proceso de planificación de tareas de control de tiempo real. La estrategia de planificación ajusta el periodo de muestreo de las tareas mediante una realimentación de las medidas estimadas del tiempo de ejecución de las tareas y una anticipación de cambios en la carga de trabajo de las tareas. El objetivo de este enfoque es mejorar el desempeño de las tareas de control a partir del alcance de una mayor utilización de los recursos compartidos de la CPU.

Lindberg et al. [19] también han desarrollado un enfoque basado en estrategias de control realimentado y acciones anticipativas con el objetivo de gestionar la temperatura de la CPU y su tiempo computacional demandado por varias tareas con dependencias de múltiples recursos. Han definido un modelo dinámico de recursos hardware y software, los cuales utilizan el flujo de recursos como entidad común.

El análisis de la literatura revela que la metodología de diseño desde la perspectiva de la ingeniería de control, las estrategias de control basadas en modelos y acciones de control anticipativas, resultan viables y pueden proveer excelentes resultados en su aplicación al diseño de mecanismos de auto-adaptación en sistemas computacionales en red de baja capacidad de cómputo.

2 MODELADO TEÓRICO Y EXPERIMENTAL

A continuación se procede a describir el proceso de planificación de tareas en sistemas empotrados, tanto desde el punto de vista teórico como experimental. La idea subyacente es realizar una caracterización de este proceso aplicando un enfoque de la ingeniería de sistemas y obtener un modelo que sea de utilidad para el diseño de estrategias de control que mejoren dicho proceso.

La caracterización del proceso como sistema significa encontrar las relaciones causa-efecto del sistema. Es decir, relaciones entre variables de entrada y de salida, definiéndose esas variables como:

- *Salidas*: variables medibles o estimables relacionadas con los requisitos u objetivos que debe cumplir el sistema empotrado tales como consumo de energía, consumo de tiempo, rendimiento del sistema, etc.
- *Entradas*: variables independientes, también medibles, que determinan el comportamiento dinámico del sistema como por ejemplo, duración del ciclo de ejecución básico de tareas, frecuencia de escucha del canal de comunicación, recursos manejados por el sistema (bytes transmitidos y recibidos, ciclos de procesador consumidos por cada tarea, etc.).

2.1 BREVE FORMALIZACIÓN TEÓRICA

En una red de dispositivos embebidos, cada dispositivo constituye un nodo que ejecuta un conjunto de tareas que siguen un modelo de ejecución de tiempo discreto. En este modelo de ejecución cada nodo ejecuta en un intervalo de tiempo dado un número aproximadamente constante de acciones o tareas. Este intervalo de tiempo se denomina ciclo de ejecución (CE) o simplemente ciclo. El comienzo de cada ciclo no está sincronizado entre los nodos de la red (ver Figura 1). Al inicio de cada ciclo, el nodo ejecuta tareas de forma secuencial tales como, adquisición de datos, filtrado, fusionado y tareas de comunicación. A consecución el nodo entra en modo de suspensión y chequea a intervalos constante de tiempo el canal de comunicación para recibir paquetes de datos.

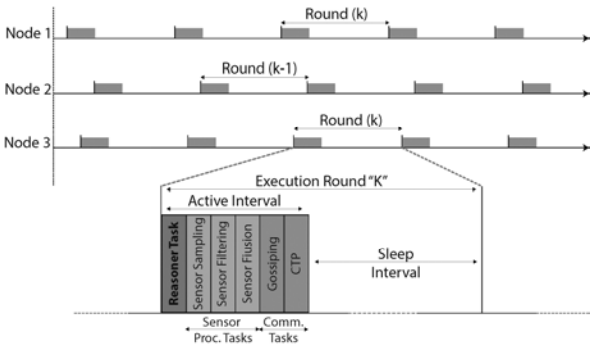

Figura 1: Descripción gráfica sobre el proceso de planificación y ejecución de tareas.

En general, el desempeño del proceso se puede caracterizar a través del tiempo consumido en cada ciclo de ejecución por cada una de las tareas que se

ejecutan y a través de la energía o potencia consumida por los componentes hardware (CPU, transceptor, etc.) al ejecutar dichas tareas. Las tareas se clasifican principalmente en tareas de comunicación (transmisión, recepción, etc.) y tareas de procesamiento (adquisición y filtrado). A continuación se describe el modelado teórico del proceso de planificación.

2.1.1 Tiempo consumido por las tareas

Según las tareas que se describen en el modelo de ejecución de la Figura 1, el tiempo consumido por cada una de ellas, en cada ciclo de ejecución k, puede describirse de forma general de la siguiente forma:

- $t_{r_{tx}}$ y $t_{r_{rx}}$, intervalos de tiempo para recibir y enviar paquetes en un ciclo de ejecución. Estos intervalos dependen principalmente en factores tales como la frecuencia de comunicación o tasa de bits del transceptor F_b, el número de bytes transmitidos (B_{tx}) y recibidos (B_{rx}), retardos en la comunicación debido al estado del canal de comunicación ($\tau_{r_{tx}}$, $\tau_{r_{rx}}$), las especificaciones del protocolo de comunicación entre dispositivos, entre otros.

$$\begin{aligned} t_{r_{tx}}(k) &= f_{tx}\left(B_{tx}, F_b, \tau_{r_{tx}}, k\right) \\ t_{r_{rx}}(k) &= f_{rx}\left(B_{rx}, F_b, \tau_{r_{rx}}, k\right) \end{aligned} \tag{1}$$

- $t_{r_{lt}}$, tiempo total de escucha del canal por cada ciclo de ejecución. Se calcula a partir del periodo de tiempo (T_{wu}) o frecuencia (F_{wu}) para escuchar el canal y el intervalo de tiempo ($\tau_{r_{lt}}$) empleado durante cada periodo para escuchar el canal.

$$t_{r_{lt}}(k) = f_{r_{lt}}\left(T_{wu}, \tau_{r_{lt}}, k\right) \tag{2}$$

- $t_{p_{tk}}$, tiempo de cómputo empleado por el procesador para ejecutar tareas. Este intervalo depende principalmente de la frecuencia de reloj del procesador y el número de ciclos (C_P) u operaciones requeridas por las tareas, entre otros factores.

$$t_{p_{tk}}(k) = f_{tk}\left(C_P, F_{CPU}, k\right) \tag{3}$$

- $t_{s_{acq}}$, tiempo requerido para adquirir muestra. Este intervalo es función del número de muestras a adquirir, S_{acq}, y el tiempo requerido para adquirir una muestra, $\tau_{s_{acq}}$.

$$t_{s_{acq}}(k) = f_{acq}\left(S_{acq}, \tau_{s_{acq}}, k\right) \tag{4}$$

2.1.2 Energía y potencia media consumida en cada ciclo de ejecución

La energía total E consumida por el dispositivo constituye el valor integral de la potencia instantánea consumida por todos sus componentes hardware en cada ciclo de ejecución. En otras palabras, es principalmente la suma de la energía consumida por el transceptor (E_r), la CPU (E_p), el sistema de adquisición (E_s) y otros periféricos (E_d):

$$\begin{aligned} E(k) &= \int_{t_0(k)}^{t_0(k)+T_{round}} P(t)dt \\ E(k) &= E_r(k) + E_p(k) + E_s(k) + E_d(k) \end{aligned} \tag{5}$$

Donde la potencia media consumida en un ciclo de ejecución, de forma general se puede expresar como:

$$P(k) = \frac{1}{T_{round}} \int_{t_0(k)}^{t_0(k)+T_{round}} P(t)dt = \frac{E(k)}{T_{round}} \tag{6}$$

La energía consumida también depende de las tareas u operaciones que ejecuta cada componente de hardware en cada ciclo de ejecución. Para el modelado se han realizado las siguientes consideraciones: (1) la corriente instantánea que demanda el componente de hardware durante la ejecución de la operación es diferente de 0 solamente durante el intervalo de tiempo en que se ejecuta dicha operación o tarea. Por ejemplo, los valores instantáneos de corriente consumida $I_{r_{tx}}$ por el transceptor durante la transmisión de paquetes, se alcanzan solamente dentro del intervalo de tiempo $t_{r_{tx}}$ (ver ecuación (1)); (2) se considera además, un valor constante de voltaje de alimentación V_{cc} a los componentes hardware. Teniendo en cuenta las consideraciones anteriores, la energía consumida por cada componente y el valor medio de potencia consumida dentro del intervalo de tiempo correspondiente a la operación, se calculan de la siguiente forma:

- Energía consumida por la CPU y el sistema de adquisición a partir de la corriente consumida cuando la CPU está activa $I_{p_{ON}}$ o en modo de bajo consumo $I_{p_{sl}}$, corriente consumida para la adquisición $I_{s_{acq}}$ y los sensores $I_{s_{ON}}$:

$$\begin{aligned} E_p(k) &= V_{cc} \int_{t_0(k)}^{t_0(k)+T_{round}} \left[I_{p_{ON}}(t) + I_{p_{sl}}(t) \right] dt \\ E_p(k) &= P_{ON}(k) t_{p_{ON}}(k) + P_{p_{sl}}(k) t_{p_{sl}}(k) \end{aligned} \tag{7}$$

$$E_s(k) = V_{cc}\int_{t_0(k)}^{t_0(k)+T_{round}} \left[I_{s_{acq}}(t) + I_{s_{ON}}(t) \right] dt$$
$$E_s(k) = P_{s_{acq}}(k) t_{s_{acq}}(k) + P_{s_{ON}}(k) t_{s_{ON}}(k) \quad (8)$$

- Energía y potencia media consumida por el transceptor a partir de la corriente consumida durante la transmisión $I_{r_{tx}}$, recepción $I_{r_{rx}}$, escuchando el canal de comunicación $I_{r_{sl}}$ o en modo de bajo consumo $I_{r_{lt}}$ ($t_f(k) = t_0(k) + T_{round}$):

$$P_{tx}(k) = \frac{V_{cc}}{t_{r_{tx}}(k)} \int_{t_0(k)}^{t_f(k)} I_{r_{tx}}(t) dt$$
$$P_{rx}(k) = \frac{V_{cc}}{t_{r_{rx}}(k)} \int_{t_0(k)}^{t_f(k)} I_{r_{rx}}(t) dt$$
$$P_{sl}(k) = \frac{V_{cc}}{t_{r_{sl}}(k)} \int_{t_0(k)}^{t_f(k)} I_{r_{sl}}(t) dt \quad (9)$$
$$P_{lt}(k) = \frac{V_{cc}}{t_{r_{lt}}(k)} \int_{t_0(k)}^{t_f(k)} I_{r_{lt}}(t) dt$$

$$E_r(k) = P_{tx}(k) t_{r_{tx}}(k) + P_{rx}(k) t_{r_{rx}}(k) + P_{r_{sl}}(k) t_{r_{sl}}(k) + P_{lt}(k) t_{r_{lt}}(k) \quad (10)$$

2.2 MODELADO EXPERIMENTAL

La formalización teórica de la sección anterior permite conocer cuáles variables y parámetros del proceso de planificación y ejecución de tareas se encuentran relacionados entre sí, estableciendo relaciones causa-efecto y dando por tanto unas pautas generales del tipo de relación. Pero en las relaciones formuladas no se describen detalles específicos de, por ejemplo, como un cambio en la magnitud de una variable de entrada afecta el cambio de magnitud de una variable de salida. Por ejemplo, en la ecuación (9) la potencia media consumida durante la transmisión de paquetes P_{tx} depende del tiempo requerido por el transceptor para transmitir paquetes, $t_{r_{tx}}$, y de la corriente $I_{r_{tx}}$ y voltaje consumidos V_{cc} durante la transmisión.

Se realizaron experimentos para caracterizar el consumo de energía y potencia media durante la transmisión de paquetes, recepción de paquetes y ejecución de tareas de adquisición, filtrado y fusionado. Para ello se utilizaron dos sensores inalámbricos donde uno de ellos hace la función de emisor de paquetes y el otro de receptor. A continuación se relacionan los parámetros experimentales empleados, donde para cada uno de los valores de frecuencia de escucha por parte del nodo receptor, se realizaron experimentos enviando paquetes de diferentes longitudes:

Bytes transmitidos o tamaño del paquete: 18, 28, 38, 48, 58, 68, 78, 88, 98, 108

Frecuencia de escucha en Hertz: 2, 4, 8, 16, 32

Se utilizaron además las siguientes herramientas y equipamiento: dispositivo de monitorización de potencia Monsoon Power Monitor, herramienta para monitorización de potencia Monsoon Power Tool, sensores inalámbricos SOWNet G-Node G301 Wireless Sensor Node, placa Arduino para marcar eventos específicos a través de los pines GPIO y Matlab como herramienta para análisis de los datos, procesamiento y modelado.

Utilizando técnicas de regresión se obtuvieron modelos de energía consumida y tiempo para cada uno de los tipos de eventos por ciclo de ejecución. En los modelos se define n_{tx} y n_{rx} como el total de secuencias para cada tipo de evento, es decir, eventos de transmisión y recepción, de forma respectiva, que suceden en cada ciclo de ejecución *k*. Los modelos obtenidos, sus parámetros y coeficiente de determinación del ajuste $\mathbf{R^2}$, se resumen a continuación:

- ***Modelos de intervalos de tiempo:***
  - *Transmisión de paquetes (R^2: 0.9992, τ_{atx} = 0.0001722, τ_{btx} = 0.001945, τ_{ctx} = 1.098, τ_{dtx} = 0.02437)*

$$t_{r_{tx}}(k) = g_{tx}\left(B_{tx}, n_{tx}, T_{wu}, k\right) = \left(\tau_{atx} + \tau_{btx} T_{wu}\right) B_{tx}(k) + n_{tx}\left(\tau_{ctx} T_{wu} + \tau_{dtx}\right) \quad (11)$$

  - *Recepción de paquetes (R^2: 0.3364, τ_{arx} = 2.876e-005, τ_{brx} = 0.02634)*

$$t_{r_{rx}}(k) = g_{rx}\left(B_{rx}, n_{rx}, k\right) = \tau_{arx} B_{rx}(k) + n_{rx}\tau_{brx} \quad (12)$$

  - *Período de escucha (R^2: 0.9997, $\tau_{r_{lt}}$ = 0.01061)*

$$t_{r_{lt}}(k) = g_{r_{lt}}\left(T_{wu}, t_{r_{tx}}, t_{r_{rx}}, k\right) = \tau_{r_{lt}}\left(T_{round} - t_{r_{tx}}(k) - t_{r_{rx}}(k)\right) \Big/ T_{wu} \quad (13)$$

  - *Adquisición y ejecución de tareas de filtrado (R^2: 0.9981, $\tau_{s_{acq}}$ = 0.3123, τ_{tk} = 0.1199)*

$$t_{p_{tk}+s_{acq}}(k) = g_{tk+acq}\left(S_{acq}, \tau_{s_{acq}}, C_P, F_{CPU}, k\right) = \tau_{s_{acq}} S_{acq}(k) + F_{CPU}^{-1} C_{P_{tkf}}(k) + \tau_{tk} \quad (14)$$

- ***Modelos de energía consumida en cada ciclo de ejecución:***

- *Transmisión de paquetes (R^2: 0.9992, ε_{atx} = 0.002544, ε_{btx} = 0.2316, ε_{ctx} = 18.68, ε_{dtx} = 1.399)*

$$\begin{aligned} E_{tx}(k) &= f_{tx}\left(B_{tx}, n_{tx}, T_{wu}, k\right) \\ &= \left(\varepsilon_{atx} + \varepsilon_{btx} T_{wu}\right) B_{tx}(k) + n_{tx}\left(\varepsilon_{ctx} T_{wu} + \varepsilon_{dtx}\right) \end{aligned} \quad (15)$$

- *Recepción de paquetes (R^2: 0.2750, ε_{arx} = 0.001091, ε_{brx} = 0.4509)*

$$E_{rx}(k) = f_{rx}\left(B_{rx}, n_{rx}, k\right) = \varepsilon_{arx} B_{rx}(k) + n_{rx}\varepsilon_{brx} \quad (16)$$

- *Período de escucha (R^2: 0.9996, ε_{alt} = 0.4435)*

$$\begin{aligned} E_{lt}(k) &= f_{lt}\left(T_{wu}, k\right) \\ &= \varepsilon_{alt}\left(T_{round} - t_{r_{tx}}(k) - t_{r_{rx}}(k)\right) \Big/ T_{wu} \end{aligned} \quad (17)$$

- *Adquisición y ejecución de tareas de filtrado (R^2: 0.9152, ε_{atk} = 0.5987, ε_{btk} = 3.797, ε_{ctk} = 7.093)*

$$\begin{aligned} E_{tk+acq}(k) &= f_{tk+acq}\left(S_{acq}, C_{P_{tk}}, k\right) \\ &= \varepsilon_{atk} S_{acq}(k) + \varepsilon_{btk} F_{CPU}{}^{-1} C_{P_{tk}}(k) + \varepsilon_{ctk} \end{aligned} \quad (18)$$

Dado que todos los eventos o tareas no son ejecutados en cada ciclo de ejecución, la energía total consumida por tanto también varía. Sin embargo las tareas se ejecutan de forma periódica cada un cierto número de ciclos de ejecución. Por ejemplo, la tarea para lanzar el protocolo de árbol de colección (CTP, *collection tree protocol*) no se ejecuta en cada ciclo, por tanto en el ciclo que se lanza esta tarea el total de paquetes de datos y bytes transmitidos es diferente a los ciclos en los que no se ejecuta. El periodo de ejecución de este tipo de tarea se puede expresar de la forma $l = L_k k$, siendo L_k la relación de periodicidad respecto al ciclo de ejecución básico k. Por tanto, la energía consumida para el periodo de ejecución l, su duración temporal y potencia media consumida para este periodo pueden expresarse de la siguiente forma:

$$\begin{aligned} T_{l\_rnd}(l) &= \sum_{k=k_0}^{k=k_0+L_k} T_{round}(k) \\ E(l) &= \sum_{k=k_0}^{k=k_0+L_k} E(k); P(l) = \frac{E(l)}{T_{l\_rnd}(l)} \end{aligned} \quad (19)$$

3 ESTRATEGIA AUTO-ADAPTATIVA BASADA EN INGENIERÍA DE CONTROL

Se ha diseñado una estrategia de auto-adaptación para el proceso de planificación y ejecución de tareas en dispositivos con bajos recursos computacionales empleando una arquitectura de control anticipativo. (ver Figura 2).

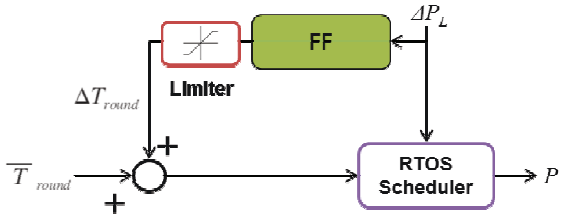

Figura 2: Estrategia de auto-adaptación basada en sistema de control anticipativo.

El control anticipativo es un método basado en modelos y que proporciona una acción de control a lazo abierto (acción anticipativa). De esta forma se reducirían posibles efectos que pudieran introducirse en la dinámica del sistema producto de la realimentación negativa, además de la mayor complejidad que pueda implicar el diseño de sistemas de control realimentados.

La estrategia de control tiene como objetivo cancelar o reducir al mínimo los efectos de las perturbaciones o carga en el consumo medio de potencia mediante ligeras correcciones, $\Delta T_{round}(k)$, del período de tiempo del ciclo de ejecución (acción de control). La carga en el consumo medio de potencia (ΔP_L) del nodo se calcula como la diferencia sobre su valor nominal o esperado durante todo el período de ejecución del protocolo de árbol de la colección (CTP). El controlador solo compensará cargas superiores a cero, es decir, consumos por encima del valor nominal, pero de forma limitada de tal forma que no se cambien los valores nominales temporales de funcionamiento del proceso de planificación. El consumo de energía se estima utilizando los modelos experimentales presentados en secciones anteriores, específicamente los modelos que estiman el consumo durante la recepción y transmisión de paquetes, adquisición de datos y ejecución de tareas de adquisición y filtrado.

A continuación se describen los pasos realizados para obtener la ecuación de la estrategia de control:

- Planteamiento de la ecuación de potencia media consumida por periodo de ejecución del protocolo CTP, $P(l)$, considerando el valor nominal de la potencia media consumida, $\bar{P}_l$, y la carga en el consumo medio de potencia, ΔP_L:

$$P(l) = \bar{P}_l + \Delta P_L \quad (20)$$

- Partiendo de la ecuación (19), calcular la potencia media nominal consumida $\overline{P}_l$ a partir de la estimación de energía nominal consumida (por periodo CTP, $\overline{E}_l$, y ciclo de ejecución simple, $\overline{E}_k$) y el valor nominal del tiempo de duración del ciclo CTP, $\overline{T}_{l\_rnd}$, y el ciclo simple, $\overline{T}_{round}$.

$$\overline{P}_l = \frac{\overline{E}_l}{\overline{T}_{l\_rnd}};\ \overline{E}_l = \sum_{k=k_0}^{k=k_0+L_k} \overline{E}_k;\ \overline{T}_{l\_rnd} = L_k \overline{T}_{round} \quad (21)$$

- Ecuaciones de la acción de control anticipativa

$$\begin{aligned} \Delta P_L &= P(l) - \overline{P}_l = 0 \\ E(l)\overline{T}_{l\_rnd} - \overline{E}_l T_{l\_rnd}(l) &= 0 \end{aligned} \quad (22)$$

- Introducción de la variable de control $\Delta T_{round}(k)$ y definición de la ecuación para el cálculo de la acción de control para un ciclo simple de ejecución:

$$\begin{aligned} T_{l\_rnd}(l) &= L_k\left(\overline{T}_{round} + \Delta T_{round}(k)\right) \\ \Delta T_{round}(k) &= \left(E(l) - \overline{E}_l\right)/\alpha_{ff};\ \alpha_{ff} = \overline{E}_l/\overline{T}_{round} \end{aligned} \quad (23)$$

- Diseño del limitador de la magnitud de la acción de control

$$\begin{aligned} \Delta T_{round}(k) \geq \left|\Delta T_{round}^{LIM}\right| &: \Delta T_{round}(k) = \left|\Delta T_{round}^{LIM}\right| \\ -\left|\Delta T_{round}^{LIM}\right| < \Delta T_{round}(k) &< \left|\Delta T_{round}^{LIM}\right| \\ \Delta T_{round}(k) \leq -\left|\Delta T_{round}^{LIM}\right| &: \Delta T_{round}(k) = -\left|\Delta T_{round}^{LIM}\right| \\ \left|\Delta T_{round}^{LIM}\right| = \Delta T_{LIM}\overline{T}_{round} &;0 < \Delta T_{LIM} < 1 \end{aligned} \quad (24)$$

3.1 EVALUACIÓN DE LA ESTRATEGIA AUTO-ADAPTATIVA

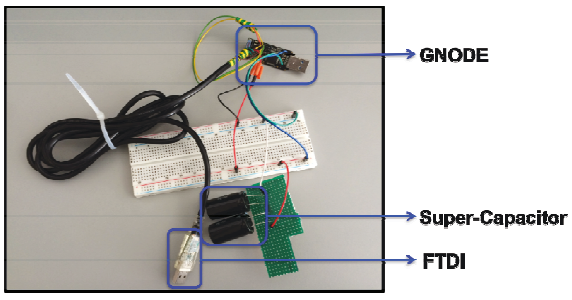

Figura 3: Instalación del sensor alimentado mediante super-capacitor para pruebas de evaluación.

Con el objetivo de realizar las pruebas de evaluación se desplegaron 6 sensores inalámbricos modelos SOWNET GNODE y sistema operativo Contiki-OS-2.7 en ejecución. Los nodos se alimentaron con super-capacitores de tal forma que pudiera evaluarse durante un periodo corto de tiempo los cambios que pudiera introducir el sistema de control en el consumo de energía de los sensores (ver Figura 3). El resto de nodos utilizan un tiempo de ciclo de ejecución variable de forma aleatoria con el objetivo de introducir perturbaciones en las comunicaciones en la red y por tanto sobrecarga en el número de paquetes esperado por el algoritmo de control.

A continuación se listan detalles sobre los experimentos realizados:

- El algoritmo de control ha sido implementada en lenguaje C.
- El algoritmo de control estaba activado en el nodo NCTRL_ON por lo que el periodo del ciclo de ejecución es variable.
- En el nodo NCTRL_OFF, alimentado por super-capacitor, el algoritmo de control estaba desactivado y el periodo del ciclo de ejecución de tareas se fijó a 5 segundos.
- Tiempo de ciclo de ejecución variable de forma aleatoria en un rango entre 4.5 y 5.5 (±10 % alrededor valor nominal) segundos en el resto de nodos. De esta forma, se provoca una variación del número de paquetes enviados durante el periodo CTP, lo cual representa una perturbación en la red.
  - Periodo mayor de 5 segundos implica menor número de paquetes enviados que lo esperado.
  - Periodo menor de 5 segundos implica un incremento del número de paquetes enviados sobre lo esperado.
- Adquisición a través de cable FTDI por cada ciclo de ejecución de la siguiente información almacenada en los nodos: duración del ciclo de ejecución, voltaje de alimentación, consumo de energía estimado, número de paquetes enviados y recibidos, entre otras variables.
- Utilización de la tasa de variación (o tasa de descarga de la batería) del voltaje de alimentación del nodo por unidad de tiempo como variable de referencia para evaluar la variación en el consumo medio de potencia del nodo. Esta tasa es proporcional a la corriente suministrada por la batería o super-capacitor, que se corresponde a su vez con la variación de su carga.

Se realizaron 3 pruebas en total, alimentando nodos con un super-capacitor. Dada la menor carga de este componente comparada con la de una batería, esto permitió realizar un análisis mediante pruebas de menor duración. La Tabla 1 resume los datos de los experimentos realizados. Para los cuales se utilizaron dos frecuencias diferentes de escucha del canal (experimento 1 frecuencia de 28 Hz y 2 Hz en los otros 2 experimentos).

Tabla 1: Resumen de datos experimentales.

Exp. No.	NCTRL_ON dV/dt [mV/s]	NCTRL_OFF dV/dt [mV/s]	Diferencia en la tasa de descarga [%]
1	-0.31893	-0.39127	18.49
2	-0.30852	-0.38576	20.02
3	-0.30147	-0.37877	20.41
		Valor medio	19.64

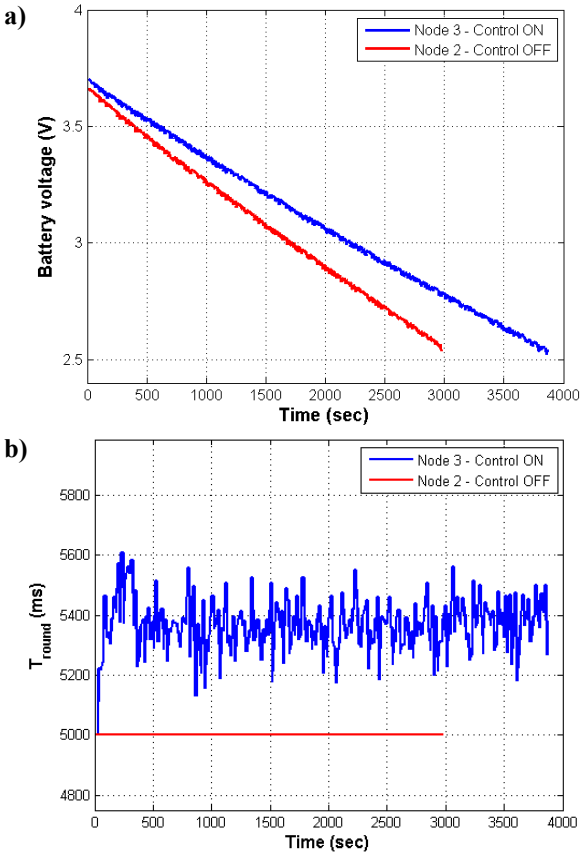

Figura 4: Datos obtenidos de pruebas con super-capacitor en el Exp. No. 3. **a)** Voltaje de suministro y **b)** duración del ciclo de ejecución.

A modo de ejemplo, la Figura 4 muestra el comportamiento del voltaje de alimentación y de la duración del ciclo de ejecución para los nodos NCTRL_ON y NCTRL_OFF durante el experimento número 3. Al estar el algoritmo de control anticipativo activo en el nodo NCTRL_ON el ciclo de ejecución varía para compensar la sobrecarga de paquetes que se reciben y de esta manera el consumo medio de potencia por periodo CTP. La pendiente de la curva de voltaje de alimentación es menor para el nodo NCTRL_ON comparado con el nodo NCTRL_OFF. Esto significa que la tasa de variación del voltaje de alimentación del nodo por unidad de tiempo es menor y por tanto el consumo medio de potencia también lo es.

En la Tabla 1 también se especifica el valor medio de tasa de descarga de la batería para ambos nodos, que se corresponde a su vez con la pendiente de la curva de voltaje de alimentación, y el porcentaje de diferencia entre ambos. El valor medio de la diferencia entre ambos para todos los experimentos es de 19.64 %, que a su vez representa un incremento también igual a este valor de la vida útil media de la batería. Estos datos demuestran las mejoras en el consumo de potencia que introduce el algoritmo de control diseñado.

4 CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se ha utilizado una metodología de diseño de estrategias de auto-adaptación basada en ingeniería de control y estrategias de control basadas en modelo, como es el caso del control anticipativo. El método propuesto permitiría que diseñadores de algoritmos utilicen métodos analíticos, apropiados de la ingeniería de control, para desarrollar estrategias de planificación de tareas auto-adaptativas para redes inalámbricas de sensores. El enfoque utilizado se diferencia de los enfoques *ad hoc*, normalmente empleados por ingenieros de software, que dependen de numerosas iteraciones durante la etapa de diseño. Comparado con estos enfoques, el enfoque propuesto basado en técnicas de ingeniería de control reduce el tiempo de diseño de redes adaptativas de sensores inalámbricos.

Quedan varias interrogantes abiertas y que a modo se relacionan a continuación. ¿Cómo afecta, desde el punto de vista dinámico, la solución propuesta al resto de nodos de la red? Soluciones basadas en toma de decisiones distribuidas, como es el caso de los algoritmos basados en consenso, ¿serían capaces de mejorar de forma general el desempeño del sistema? De cara a ser utilizados por diseñadores ¿sería viable y de utilidad introducir los resultados alcanzados (modelos, algoritmos, etc.) en herramientas de diseño de redes adaptativas de sensores inalámbricos?

Estas y otras posibles interrogantes serían temas interesantes a estudiar y desarrollar en trabajos futuros. Los resultados alcanzados en este trabajo servirían de base para desarrollar herramientas que sirvan de apoyo a diseñadores de redes adaptativas de sensores inalámbricos. Este tipo de herramientas podrán ser transferidas a la industria vinculada a las Tecnologías de la Información y las Comunicaciones, por lo que tendría un alto impacto a nivel económico y también social. Esto se debe principalmente al peso que va ganando cada vez más en diferentes sectores industriales, los sistemas distribuidos y el Internet de las Cosas (IoT, *Internet of Things*).

Agradecimientos

Este trabajo se ha desarrollado con el apoyo de la ayuda José Castillejo para jóvenes doctores CAS14/00423 del Ministerio de Educación, Cultura y Deporte de España y el proyecto IoSENSE "Flexible FE/BE Sensor Pilot Line for the Internet of Everything" del Ministerio de Economía y Competitividad y el programa ECSEL Joint Undertaking. Se agradece la colaboración en la fase experimental de la Ing. Soumya Subramanya (Delft University of Technology).

Referencias

[1] P. Sawyer, N. Bencomo, J. Whittle, E. Letie, and A. Finkelstein, "Requirements-aware systems: A research agenda for RE for self-adaptive systems," Sydney, NSW, 2010, pp. 95-103.

[2] G. Brown, B. H. C. Cheng, H. Goldsby, and J. Zhang, "Goal-oriented specification of adaptation requirements engineering in adaptive systems," presented at the Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems, Shanghai, China, 2006.

[3] P. Den Hamer and T. Skramstad, "Autonomic service-oriented architecture for resilient complex systems," Madrid, 2011, pp. 62-66.

[4] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *Intelligent Systems and their Applications, IEEE,* vol. 14, pp. 54-62, 1999.

[5] F. D. Macías-Escrivá, R. Haber, R. Del Toro, and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications," *Expert Systems with Applications,* vol. 40, pp. 7267-7279, 2013.

[6] U. Richter, M. M. and, J. Branke, C. Müller-Schloer, and H. Schmeck, "Towards a generic observer/controller architecture for Organic Computing," in *INFORMATIK 2006: Informatik für Menschen. GI-Edition - Lecture Notes in Informatics*, 2006, pp. 112-119.

[7] D. Garlan, S.-W. Cheng, and B. Schmerl, "Increasing system dependability through architecture-based self-repair," in *Architecting dependable systems*, Rog, L. rio De, G. Cristina, and R. Alexander, Eds., ed: Springer-Verlag, 2003, pp. 61-89.

[8] J. Schmitt, M. Roth, R. Kiefhaber, F. Kluge, and T. Ungerer, "Realizing self-x properties by an automated planner," Karlsruhe, 2011, pp. 185-186.

[9] K. Geihs, P. Barone, F. Eliassen, J. Floch, R. Fricke, E. Gjorven, S. Hallsteinsen, G. Horn, M. U. Khan, A. Mamelli, G. A. Papadopoulos, N. Paspallis, R. Reichle, and E. Stav, "A comprehensive solution for application-level adaptation," *Software: Practice and Experience,* vol. 39, pp. 385-422, 2009.

[10] H. Liu and M. Parashar, "Accord: a programming framework for autonomic applications," *Trans. Sys. Man Cyber Part C,* vol. 36, pp. 341-352, 2006.

[11] N. Bencomo, P. Grace, C. Flores, D. Hughes, and G. Blair, "Genie: supporting the model driven development of reflective, component-based adaptive systems," presented at the Proceedings of the 30th international conference on Software engineering, Leipzig, Germany, 2008.

[12] C. Peper and D. Schneider, "Component engineering for adaptive ad-hoc systems," presented at the Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems, Leipzig, Germany, 2008.

[13] T. Vogel, S. Neumann, S. Hildebrandt, H. Giese, and B. Becker, "Model-driven architectural monitoring and adaptation for autonomic systems," presented at the Proceedings of the 6th international conference on Autonomic computing, Barcelona, Spain, 2009.

[14] M. Salehie and L. Tahvildari, "Towards a goal-driven approach to action selection in self-adaptive software," *Software - Practice and Experience,* 2011.

[15] J. Tanner, "Feedback control in living prototypes: A new vista in control engineering," *Medical and Biological Engineering and Computing,* vol. 1, pp. 333-351, 1963.

[16] J. Hellerstein, S. Parekh, Y. Diao, and D. M. Tilbury, *Feedback control of computing systems*: Wiley-IEEE Press, 2004.

[17] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*: Prentice Hall: Englewood Cliffs, New Jersey, 2006.

[18] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback–Feedforward Scheduling of Control Tasks," Real-Time Systems, vol. 23, pp. 25-53, 2002/07/01 2002.

[19] M. Lindberg and K. Arzen, "Feedback Control of Cyber-physical Systems with Multi Resource Dependencies and Model Uncertainties," in Real-Time Systems Symposium (RTSS), 2010 IEEE 31st, 2010, pp. 85-94.