UNIVERSIDAD DE SALAMANCA

Diseño e implementación de un sistema de computación distribuida con Raspberry Pi, y estudio comparativo del mismo frente a otras soluciones

by

Diego Martín Arroyo

Trabajo de Fin de Grado en el marco de los estudios de
Graduado en Ingeniería Informática

en la
Facultad de Ciencias
Departamento de Informática y Automática

19 de abril de 2015

Declaration of Authorship

I, AUTHOR NAME, declare that this thesis titled, 'THESIS TITLE' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Write a funny quote here."

If the quote is taken from someone, their name goes here

UNIVERSIDAD DE SALAMANCA

Abstract

Facultad de Ciencias
Departamento de Informática y Automática

Doctor of Philosophy

by Diego Martín Arroyo

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Índice general

Índice de figuras

Índice de cuadros

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light $c = 2{,}997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

For/Dedicated to/To my...

Capítulo 1

Introducción

La presente memoria recoge el proceso de instalación de un sistema distribuido formado por dispositivos Raspberry Pi y la creación de un conjunto de protocolos, herramientas y programas para la utilización del mismo como herramienta de investigación en el campo de la computación distribuida y como herramienta didáctica para disciplinas relacionadas con dicho área.

El sistema se compone de un conjunto de dispositivos físicos compuesto por los nodos de computación y una serie de módulos accesorios, así como los diferentes mecanismos de alimentación y refrigeración, un conjunto de herramientas software que permiten la coordinación y comunicación entre los diferentes procesos y una serie de herramientas que facilitan el trabajo con el sistema.

1.1. Contenidos de la memoria

- Definición del dominio del problema y motivación
- Evaluación de alternativas y propuesta de solución
- Plataforma física
- Herramientas creadas

    MarcoPolo

    MarcoTools

    MarcoStatusMonitor

    Deployer

    Material didáctico

Ricard Agrawala

- Aplicaciones distribuidas

  MPI

  Python

  Tomcat

- Evaluación
- Consulta a los estudiantes
- Evaluación de las prácticas en MPI
- Evaluación de las prácticas en Sistemas Distribuidos

Capítulo 2

Dominio del problema

La utilización de algoritmos distribuidos implica mejoras sustanciales en una gran cantidad de aplicaciones, incrementando la capacidad global de computación de un sistema mediante la unión de varios dispositivos de cómputo que trabajan de como una única unidad indivisible a la vez que mantienen un alto grado de independencia y una tolerancia global a fallos muy alta. Sin embargo, el desarrollo de aplicaciones distribuidas implica el uso de un conjunto de nodos cuyo coste y mantenimiento es costoso.

Dicho aumento de la potencia implica una mayor complejidad en el desarrollo de algoritmos que puedan aprovechar de forma óptima este tipo de sistemas. Varios factores como la sincronización y la comunicación entre partes, o errores tales como condiciones de carrera son mucho más comunes que en otro tipo de aplicaciones. Dichas dircunstancias no solo dificultan el desarrollo de este sistema, sino también la comprensión de los fundamentos básicos de la Computación Distribuida.

Si bien la mayoría de las aplicaciones en las que el paradigma de computación distribuida introduce mejoras suelen exigir una gran capacidad de cálculo, su desarrollo únicamente requiere un conjunto de instancias independientes de un software (sistema operativo, contenedor de servicios...) con las que trabajar. Dicha característica implica que la utilización de nodos de precio reducido (o incluso reutilizados) para el diseño, análisis y evaluación de este tipo de algoritmos constituye una alternativa válida frente a sistemas de precio superior.

Sumada a dicha motivación existe el potencial aprovechamiento de este sistema como herramienta didáctica que facilite el aprendizaje de conceptos como el reparto de procesos, balance de carga o la compartición de recursos en asignaturas centradas en este tipo de conceptos dentro de los planes de estudio de Ingeniería Informática o titulaciones afines.

Con el presente proyecto se plantea la creación de un sistema con estas características aprovechando el bajo coste de los componentes del mismo que permita en primer lugar su utilización como herramienta de análisis y diseño (o incluso su utilización como plataforma definitiva) de aplicaciones distribuidas y en segundo lugar la posibilidad de uso como herramienta educativa.

A la hora de crear el sistema se realiza un análisis de las diferentes alternativas, a fin de escoger la alternativa que mejor satisfaga los objetivos definidos.

Figura: Tabla de alternativas

2.1. Objetivos del proyecto

Este proyecto cuenta con varios objetivos muy diferentes entre sí, que se agrupan en tres categorías:

- Arquitectura subyacente
  Definición de los componentes hardware a utilizar en el sistema, interconexión de los mismos, soluciones de alimentación eléctrica, almacenamiento....
- Servicios a proveer
  Conjunto de servicios que podrán ser aprovechados por diferentes clientes para explotar la capacidad de cálculo de las máquinas.
- Componente didáctico
  Creación de aplicaciones, herramientas y documentación como alternativa a las instalaciones típicas utilizadas actualmente.

2.2. Definiciones

2.2.1. Definición del dominio del problema

El sistema se ubica en una Facultad universitaria con aproximadamente 600 alumnos (**cita requerida**) con varias asignaturas en las que se imparten áreas de conocimiento relacionados con la Computación Distribuida, en particular las asignaturas **Arquitectura de Computadores** y **Sistemas Distribuidos** [1].

2.2.2. Modelado del sistema actual

La Facultad cuenta con varias aulas y laboratorios informáticos donde los alumnos disponen de la intraestructura necesaria para realizar los ejercicios y prácticas asignadas. Dichos espacios permiten utilizar cualquier equipo como nodo, pues se integran en la misma red, siendo incluso factible la comunicación directa entre equipos situados en diferentes aulas o incluso edificios. La conexión es relativamente rápida, contando con un cableado capaz de soportar teóricamente transferencias de hasta 100Mb/s de forma bidireccional (*full-duplex*) (**Cita requerida**). La gestión de un sistema de autenticación se realiza mediante el protocolo LDAP (*Lightweight Directory Access Protocol*) [2], contando con un sistema de ficheros centralizado que permite acceder a la información de un usuario desde cualquier equipo, facilitando las tareas de replicación de la información entre nodos.

La mayoría de las prácticas asignadas a los alumnos son desarrolladas en el lenguaje **Java**, ya conocido por la totalidad de los estudiantes gracias a asignaturas previamente cursadas (**Cita requerida**) y que facilita el despliegue y la compatibilidad entre diferentes equipos de trabajo sustancialmente. En ocasiones es necesario el uso de lenguajes como C y se plantean alternativas a *Java* como Python o C#.

Problemas conocidos Si bien la infraestructura existente es capaz de proveer a los estudiantes de los recursos necesarios, identificamos una serie de problemas inicialmente:

- Cada grupo de alumnos necesita tres estaciones de trabajo para poder realizar algunos de los ejercicios propuestos.
- El servidor LDAP constituye un "cuello de botella", pues todos los alumnos acceden a él de forma intensiva, provocando el fallo por exceso de peticiones del mismo.
- Las técnicas de programación utilizadas hasta la fecha tienen un rendimiento bajo y son en ocasiones relativamente complejas.

2.2.3. Identificación de usuarios participantes

- Estudiantes de tercero y cuarto curso del Grado en Ingeniería Informática.
- Doctentes de las asignaturas Arquitectura de Computadores y Sistemas Distribuidos.
- Administradores del Sistema.

2.3. Identificación de las necesidades de cada parte

2.3.1. Necesidades de los alumnos

- Entorno de trabajo sencillo que agilice el desarrollo de sus prácticas.
- Posibilidad de observar los resultados de las ejecuciones de forma sencilla.
- Facilidades para el despliegue de los diferentes ejecutables en todas las máquinas, así como el consumo de los servicios que estos implementen.

2.3.2. Necesidades de los docentes

- Entorno versátil sobre el cual puedan llevarse a cabo la totalidad de las prácticas y ejercicios propuestos, aportando si es posible algún tipo de ventaja sobre el sistema en uso.

2.3.3. Administrador

- Sistema integrable en la intraestructura actual cuyo mantenimiento sea sencillo y cuyo enfoque garantice la escalabilidad y su durabilidad.

2.4. Propuestas para la búsqueda de necesidades

- Encuestas o entrevistas a todas las partes.
- Evaluación de la experiencia de uso en las diferentes etapas de desarrollo del sistema.

2.5. Identificación de requisitos

2.5.1. Requisitos de almacenamiento de la información

- Gestión de usuarios (credenciales de autenticación)
- Gestión de los datos de cada usuario
- *Logs* del sistema

2.5.2. Identificación de requisitos funcionales

2.5.3. Identificación de requisitos no funcionales

- El *software* debe ser mantenible y robusto[1].
- Reducción de los costes de desarrollo.
- Definición de los protocolos de comunicación.
- Definición de los protocolos de seguridad y confidencialidad.
- Definición de la interacción con el usuario.
- Integridad del sistema y fiabilidad (*uptime*, recuperación frente a fallos).
- Productos a crear.
- Compatibilidad con las prácticas y ejercicios.

2.6. Evaluación de alternativas

A la hora de evaluar las diferentes opciones que satisfagan los requisitos descritos, se consideran los siguientes aspectos:

- Coste económico.
- Prestaciones técnicas (potencia de procesamiento, entrada/salida, capacidad de almacenamiento,facilidad de interconexión con otros materiales...).
- Facilidad de trabajo y de aprendizaje (documentación disponible, proyectos similares ya realizados, conocimiento sobre la plataforma en cuestión...).
- Escalabilidad del sistema.
- Necesidades de mantenimiento del sistema.
- Consumo del sistema (consumo eléctrico).
- Obsolescencia del sistema (número de años en los que el sistema podrá ser actualizado (tanto en hardware como software) y será capaz de seguir siendo una herramienta adecuada para el propósito planteado).

[1] Siendo dicha robustez garantizada mediante el uso de *software* utilizado por una base de usuarios significativa, una arquitectura conocida, pruebas realizadas sobre él o un equipo de desarrollo en activo, entre otras

2.6.1. Propuesta de solución: Virtualización de entornos de trabajo

Crear un conjunto de nodos virtuales dentro de una máquina que simulen un sistema distribuido

Ventajas intrínsecas de la solución Simplificación del sistema (reduce las necesidades de adquisición y mantenimiento de hardware). Gestión de varias partes del sistema (sistema de ficheros centralizado, gestión de usuarios...) de forma mas sencilla.

Inconvenientes intrínsecos del sistema No se exploran apenas las posibilidades de un sistema distribuido formado por varios equipos físicamente independientes.

Facilidad de trabajo y curva de aprendizaje Requiere una etapa de formación en materia de virtualización

Prestaciones técnicas

Coste econonómico

Escalabilidad del sistema Dependiente de las capacidades de virtualización del equipo disponible, y el número de nodos y usuarios a gestionar (previsiblemente alto)

Necesidades de mantenimiento Las necesidades propias de un sistema GNU/Linux junto a las específicas de la virtualización de los equipos.

Consumo energético del sistema

Obsolescencia del sistema

Material con el que se cuenta actualmente Se plantea el aprovechamiento de equipos pertenecientes a la Universidad, por lo que se estima un coste muy pequeño a la hora de adquirir material.

Otras características

2.6.2. Propuesta de solución: Clúster con equipos de escritorio

Se plantea la reutilización de equipos de escritorio pertenecientes a la Universidad que ya no se encuentran en uso (debido a su renovación, falta de potencia como PC...) para la creación de este sistema.

Ventajas intrínsecas de la solución La potencia del sistema es mucho mayor que la de cualquier otra solucion. Reduce dramáticamente el coste de adquisición de material y permite dar nueva vida a material universitario. La arquitectura es conocida

Inconvenientes intrínsecos del sistema No se exploran las características únicas de otros sistemas menos "convencionales", tales como la utilización de sistemas embebidos. Consumo mayor. Mayor demanda de espacio. Debido a el mayor tamaño de los equipos, puede llegar a ser complicado implementar las soluciones de visualizacion planteadas (LEDs, etc)

Facilidad de trabajo y curva de aprendizaje Soporte completo de casi la totalidad de las distribuciones de GNU/Linux. Las necesidades de manipulación de hardware se minimizan.

Prestaciones técnicas Arquitectura x86/x64 Entre 2 y 4 GB de memoria Conectividad Ethernet, USB Almacenamiento en disco duro

Coste econonómico

Escalabilidad del sistema Dependiente únicamente del coste económico de la adquisición de nuevos equipos

Necesidades de mantenimiento Las necesarias en cualquier sistema GNU/Linux y las específicas del montaje dado (en materia de refrigeración, gestión de cableado, etc)

Consumo energético del sistema

Obsolescencia del sistema

Material con el que se cuenta actualmente La Facultad de Ciencias ya dispone de los equipos, pues se plantea la reutilización de los mismos

Otras características

2.6.3. Clúster con equipos embebidos multimedia

Utilización de equipos embebidos diseñados para aplicaciones multimedia en el sistema (ejemplos son Chromecast, Apple TV, Amazon Fire TV...)

Ventajas intrínsecas de la solución Relacion potencia/precio presumiblemente superior a soluciones de coste similar como las placas Raspberry Pi.

Inconvenientes intrínsecos del sistema Dificultad de conexión (generalmente la conexión a red se realiza de forma inalámbrica, ausencia casi absoluta de cualquier conexión cuya finalidad no sea la emisión de vídeo o conexión con sistemas de almacenamiento mediante USB), falta de puertos GPIO, I2C...

Facilidad de trabajo y curva de aprendizaje Es difícil determinar la viabilidad de esta solución, pues no se cuenta con experiencia previa ni una documentación amplia al respecto. Es probable que sea necesaria la manipulación del sistema a muy bajo nivel. Lo cual incrementa el grado de complejidad de la solución.

Prestaciones técnicas Arquitectura ARM 2 núcleos a 1.2 GHz 512 MB de RAM Almacenamiento: 2 GB no expandibles Alimentación por microUSB

Coste econonómico

Escalabilidad del sistema Dependiente del coste de adquisición de nuevos equipos y las facilidades de interconexión de la plataforma (previsiblemente compleja, debido a la ausencia de sistemas de interconexión más allá de WiFi)

Necesidades de mantenimiento Dependiente del número de modificaciones que se realicen a las capas más bajas. En el peor de los casos puede que el administrador del sistema tenga que someterse a una etapa de formación para realizar un mantenimiento adecuado del sistema sin depender de desarrolladores previos. Las derivadas del mantenimiento de un sistema Linux sumadas a posibles problemas de interconexión si se utiliza una red inalámbrica (conexión a la LAN de la infraestructura local, interferencias...).

Consumo energético del sistema

Obsolescencia del sistema Difícil de determinar: no se cuenta con una gran cantidad de software para este tipo de sistemas más allá de las aplicaciones multimedia. No obstante, el sistema subyacente es conocido (Linux)

Material con el que se cuenta actualmente No se dispone de material de estas o similares características

Otras características

2.6.4. Clúster con equipos embebidos multimedia

Utilización de equipos embebidos diseñados para aplicaciones multimedia en el sistema (ejemplos son Chromecast, Apple TV, Amazon Fire TV...)

Ventajas intrínsecas de la solución Relacion potencia/precio presumiblemente superior a soluciones de coste similar como las placas Raspberry Pi.

Inconvenientes intrínsecos del sistema Dificultad de conexión (generalmente la conexión a red se realiza de forma inalámbrica, ausencia casi absoluta de cualquier conexión cuya finalidad no sea la emisión de vídeo o conexión con sistemas de almacenamiento mediante USB), falta de puertos GPIO, I2C...

Facilidad de trabajo y curva de aprendizaje Es difícil determinar la viabilidad de esta solución, pues no se cuenta con experiencia previa ni una documentación amplia al respecto. Es probable que sea necesaria la manipulación del sistema a muy bajo nivel. Lo cual incrementa el grado de complejidad de la solución.

Prestaciones técnicas Arquitectura ARM 2 núcleos a 1.2 GHz 512 MB de RAM Almacenamiento: 2 GB no expandibles Alimentación por microUSB

Coste econonómico

Escalabilidad del sistema Dependiente del coste de adquisición de nuevos equipos y las facilidades de interconexión de la plataforma (previsiblemente compleja, debido a la ausencia de sistemas de interconexión más allá de WiFi)

Necesidades de mantenimiento Dependiente del número de modificaciones que se realicen a las capas más bajas. En el peor de los casos puede que el administrador del sistema tenga que someterse a una etapa de formación para realizar un mantenimiento adecuado del sistema sin depender de desarrolladores previos. Las derivadas del mantenimiento de un sistema Linux sumadas a posibles problemas de interconexión si se utiliza una red inalámbrica (conexión a la LAN de la infraestructura local, interferencias...).

Consumo energético del sistema

Obsolescencia del sistema Difícil de determinar: no se cuenta con una gran cantidad de software para este tipo de sistemas más allá de las aplicaciones multimedia. No obstante, el sistema subyacente es conocido (Linux)

Material con el que se cuenta actualmente No se dispone de material de estas o similares características

Otras características

2.6.5. Clúster con Raspberry Pi

Utilizar la plataforma de hardware libre Raspberry Pi para la creación del sistema, disponiendo los diferentes equipos en un pequeño "rack" con un sistema de alimentación propio centralizado y una conexión directa a la infraestructura local.

Ventajas intrínsecas de la solución Existen varias soluciones similares bien documentadas. El hardware es flexible, barato y el consumo es pequeño. Gran comunidad de desarrolladores alrededor de la plataforma.

Inconvenientes intrínsecos del sistema La potencia del sistema es pequeña (ver seccion prestaciones técnicas)

Facilidad de trabajo y curva de aprendizaje Ya se cuenta con experiencia en el manejo de estas placas. Amplia documentación de las prestaciones de la misma. Proyectos similares ya realizados. Soporte completo de varias distribuciones de GNU/Linux

Prestaciones técnicas Arquitectura ARM Entre 512 MB y 1 GB de memoria 1 o 4 Núcleos a 700 o 900 MHz (overclock a 1 GHz de forma segura) Conectividad Ethernet, I2C, GPIO Alimentación por microUSB/GPIO Almacenamiento entre 1 GB y 256 GB mediante tarjetas MicroSD/SD

Coste econonómico

Escalabilidad del sistema Dependiente únicamente del coste económico de la adquisición de nuevos equipos

Necesidades de mantenimiento Las mismas que cualquier sistema GNU/Linux de iguales características. Pueden surgir problemas con la fuente de alimentación, dado que es una solución propia.

Consumo energético del sistema Variable según modelo, entre 3 y 4 W, con 5V de tensión y un amperaje variable entre 0.6 y 0.8 A

Obsolescencia del sistema El software de terceros (sistema operativo, bibliotecas, etc) a incluir está respaldado por una comunidad extensa que provee actualizaciones de forma continua, por lo que previsiblemente el sistema podrá estar actualizado durante varios años. Las necesidades que el sistema cubre no demandarán previsiblemente una mayor potencia de cálculo.

Material con el que se cuenta actualmente El Departamento de Informática y Automática cuenta con varios de estos equipos se plantea la reutilización de los mismos

Otras características

2.6.6. Elección de la solución

2.6.7. Raspberry Pi: Elección de las características básicas del sistema

Comparativa de las características relevantes de los diferentes modelos de Raspberry Pi. Quedan descartados los modelos A y A+ por la carencia de puerto Ethernet (amén de otras características necesarias)

2.6.8. Elección del sistema operativo

	Modelo B	Modelo B+	Modelo B 2
Procesador	ARMv6 1 Núcleo, 700 MHz (safe overclock hasta 1GHz)	ARMv6 1 Núcleo, 700 MHz (safe overclock hasta 1GHz)	ARMv7 4 Núcleos a 900 MHz
Memoria	512 MB compartidos con GPU	512 MB compartidos con GPU	1 GB compartido con GPU
Evaluación de rendimiento con LINPACK [5][6][7]	40.64	40.64	92.88
Conexiones	2 USB, GPIO de 8 pines. Ethernet 10/100	4 USB, GPIO de 17 pines. Ethernet 10/100	4 USB, GPIO de 17 pines. Ethernet 10/100
Consumo medio [8]	700 mA, 5 V (3.5 W)	600 mA, 5 V (3 W)	800 mA, 5 V (4 W)
Almacenamiento	SD	MicroSD	MicroSD
Alimentación	Mediante MicroUSB o los pines GPIO	Mediante MicroUSB o los pines GPIO	Mediante MicroUSB o los pines GPIO
Sistemas operativos compatibles	▪ Archlinux ARM ▪ OpenELEC ▪ Puppy Linux ▪ Raspbmc ▪ RISC OS ▪ Raspbian ▪ XBian ▪ openSUSE ▪ Slackware ARM ▪ FreeBSD ▪ Plan 9 ▪ Kali Linux ▪ Sailfish OS ▪ Pidora (Fedora Remix) ▪ Lista completa en [1]	Los mismos que para el modelo B	Hasta la fecha, únicamente: ▪ Ubuntu Snappy Core ▪ Raspbian ▪ OpenELEC ▪ RISC OS ▪ Según la web de ArchLinux, también soporta este sistema operativo [2]
Otros	Modelo descatalogado, el soporte oficial y proporcionado por la comunidad probablemente será menor que para los modelos más recientes en el futuro.		Lleva poco tiempo en el mercado (apenas un mes). Se conocen pequeños fallos en el hardware (fotosensibilidad de algún componente)

Nombre	Enfoque	Características notables	Ventajas	Inconvenientes	Software disponible
ArchLinux ARM	Distribucion ligera centrada en el minimalismo y la disponibilidad de software novedoso. Requiere sin embargo que el usuario conozca el entorno GNU/Linux antes de utilizarlo	Muy optimizado con un ciclo de desarrollo que permite contar con software puntero en poco tiempo	Eficiente, gran comunidad alrededor, relativamente sencillo de utilizar	En ocasiones puede ser complejo su uso. Ya no se incluye en las distribuciones por defecto de la Fundacion Raspberry Pi, lo cual puede suponer falta de soporte oficial	8700 paquetes disponibles en los repositorios oficiales, más pequeño que para otras distribuciones, si bien no se ha encontrado aun software no compatible
Ubuntu Snappy Core	Centrado en la facilidad de uso	Es la distribución más popular (en equipos de escritorio) con gran cantidad de paquetes disponible	Fácil de configurar, gran cantidad de soporte	Aún no ha sido probado en la Raspberry de forma intensiva.El rendimiento de ubuntu suele ser menor al de otros sistemas operativos debido a la gran cantidad de paquetes incluidos por defecto.	
Raspbian					

Nombre	Enfoque	Características notables	Ventajas	Inconvenientes	Software disponible
RISC OS	Diseñado específicamente para la arquitectura ARM, aprovechando las posibilidades de dicha arquitectura	Eficiente, basado en el RISCOS original, incluyendo características del mismo. Sistema monousuario con multitarea cooperativa (en contraste con multihilo o multitarea apropriativa)	Muy eficiente	No esta basado en un sistema conocido previamente. Relativamente desfasado en cuanto a la arquitectura del Sistema Operativo. El software suele ser programado en BBC BASIC	
Gentoo	Diseñado para permitir la personalización del sistema al máximo nivel posible	Enfocado en la personalizacion, siendo el sistema compilado en la maquina sobre la que se va a utilizar en vez de ser descargado como archivo binario	Permite ser customizado de forma sencilla	Poco soportado en Raspberry Pi	
Windows 10	Diseñado para el paradigma IoT	Sencillo de utilizar Soporte de .NET	Poco probado. Esta diseñado para un proposito especifico. No compatible con software para Linux de forma nativa		

Capítulo 3

Herramientas

La complejidad que acarrea el uso de aplicaciones distribuidas hace necesario el uso de herramientas que permitan el desarrollo de forma cómoda del propio sistema, su uso posterior como herramienta de prueba de aplicaciones distribuidas y por último, facilitar el aprendizaje de algoritmos y herramientas distribuidas.

Muchas de las aplicaciones distribuidas utilizadas incluyen varias herramientas para facilitar su uso. Sin embargo estas soluciones suelen ser diseñadas para el propósito específico de dicha aplicación, y son difíciles de adaptar a otros contextos. Debido a esta carencia, se han creado varias herramientas propias que permiten aprovechar al máximo este sistema.

3.1. MarcoPolo, el protocolo de descubrimiento de servicios

Uno de los problemas típicos a la hora de crear un sistema distribuido es la localización de cada uno de los nodos que lo conforman. Soluciones como servicios de nombres (DNS) permiten crear estructuras jerárquicas donde cada nodo está identificado por un nombre previamente conocido. También existen protocolos inspirados en este como mDNS (*Multicast Domain Name Service*) donde la necesidad de un servidor de nombres desaparece, y los nodos son capaces de encontrarse entre ellos mediante multicast. Otras alternativas como Bonjour, Avahi o AppleTalk (ya descontinuado) también han sido evaluadas.

Sin embargo, estas y otras soluciones similares no responden a una de las necesidades básicas del sistema a construir: la condición de que la información que conoce cada nodo sobre el resto en el arranque del sistema es nula. Si bien con mDNS evitamos contar con un servidor de nombres, debemos conocer el nombre de cada máquina o esta debe anunciarse en la red antes de poder estar disponible (mDNS Probing). Dicho

inconveniente se suma al hecho de que DNS y protocolos similares son creados con el único propósito de resolver la correspondencia nombre - dirección de red de un equipo, y son difícilmente extensibles a otro tipo de aplicaciones. Además, la mayoría de los protocolos asumen que la información de un nodo presente de una red local es de interés para el resto de nodos de la red, lo cual dificulta la independencia de un conjunto de equipos frente al resto.

Una de las piezas clave del sistema consiste en la escalabilidad del mismo en tiempo real: no es necesario conocer qué nodos participan en el sistema hasta que no se vayan a utilizar. Además, se pretende optimizar al máximo cada uno de los nodos del sistema por separado, por lo que dedicar uno de ellos como "autoridad" frente a la que el resto de nodos se registren y esta actúe posteriormente como nodo coordinador y "resolver" supone una dedicación de recursos innecesaria y que dificulta la escalabilidad del sistema. Además, la gestión del espacio de direcciones de la red en la que se integra el sistema no es gestionado por este y además es compartido con una gran cantidad de equipos adicionales. Esto implica que las direcciones de cada nodo son asignadas por un servidor DHCP (*Dynamic Host Configuration Protocol*) sobre el que no se tiene control, y cuyas direcciones son asignadas para intervalos de tiempo pequeños[1]. Por otro lado, la clave de este sistema no la constituye la disponibilidad de un nodo, sino las aplicaciones distribuidas que pueden utilizarse en el mismo (de ahora en adelante denominaremos a estas "servicios"). Un nodo puede contar con un conjunto de servicios diferente al de sus vecinos, y por tanto colaborará en unas tareas y en otras no en virtud de dicha disponibilidad. Este requisito no es satisfecho por la mayoría de los sistemas anteriormente mencionados.

Motivada por esta serie de características surge la necesidad de crear un pequeño protocolo de descubrimiento de nodos basado principalmente en los servicios que dichos nodos pueden (y desean) ofrecer al conjunto de la malla. Además, siendo uno de los objetivos funcionales del sistema el aprovechamiento del mismo como herramienta didáctica, surge la necesidad de que dos conjuntos de nodos puedan trabajar en la misma red de forma independiente. Como aproximación para satisfacer estas necesidades surge el procolo de descubrimiento de servicios **MarcoPolo**

3.1.1. MarcoPolo: Introducción

MarcoPolo es un protocolo de descubrimiento de servicios cuya dinámica y nombre se inspiran en el juego homónimo, en el cual uno de los integrantes debe encontrar al resto privado de visión mediante ecolocalización (gritando la palabra clave "Marco", cuya

[1] Durante el desarrollo del sistema se observa que las direcciones son asignadas por periodos de tiempo pequeños y no suelen repetirse a menos que dicha dirección no haya sido asignada anteriormente, fenómeno que suele darse con bastante frecuencia.

respuesta por parte del resto de jugadores es "Polo"). El protocolo se compone de dos roles claramente diferenciados (y prácticamente independientes aún siendo ejecutados en el mismo nodo): **Marco**, encargado de enviar consultas a la red y **Polo**, que emite una respuesta a dichos comandos y gestiona la información de cada nodo.

Con el objetivo de posibilitar la coexistencia de varias "mallas" de nodos independientes (donde los servicios ofrecidos por un nodo únicamente sean conocidos y consecuentemente aprovechables por el resto de sus vecinos) a la vez que las consultas son realizadas a todos los integrantes sin necesidad de conocer su identificador en la red (dirección a nivel de red o enlace, nombre *DNS*) se utilizan mensajes uno-a-muchos, conocidos generalmente con el nombre *multicast*, donde cada una de las *mallas* se comunicará con el resto de integrantes de la misma a través de un grupo preestablecido (o consensuado por dichos nodos).

El protocolo consiste en una serie de mensajes (a partir de ahora denominados *comandos*) que contienen las consultas sobre la información de uno o varios servicios, nodos o información sobre la propia *malla* que un nodo desee conocer, así como la respuesta a dichas consultas. Dichos mensajes son enviados como cadenas de texto que almacenan la información en estructuras de datos JSON (*JavaScript Object Notation*) debido a la gran legibilidad de estas por humanos y la gran cantidad de herramientas disponibles para su creación y procesado.

3.1.2. Arquitectura en detalle

3.1.3. Comandos

Los comandos de MarcoPolo constituyen las primitivas del protocolo. Actualmente se cuenta con las siguientes primitivas y las correspondientes respuestas:

3.1.4. Esquemas de comunicación

3.1.4.1. Comando Marco

El comando Marco se envía al grupo *multicast* definido en la configuración de la instancia local de **Marco**. Los nodos suscritos a dicho grupo (aquellos que pertenecen a la "malla") reciben el mensaje y emiten una respuesta **Polo**. Debido a la falta de una conexión entre los nodos (debido a que todos los mensajes son intercambiados utilizando el protocolo UDP) se fija un tiempo de espera de respuesta, durante el cual se reciben y acumulan

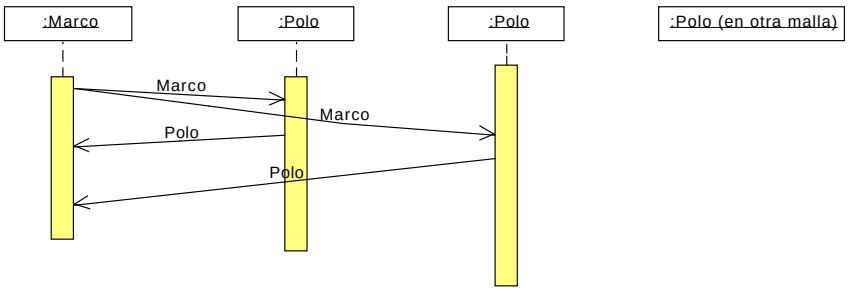

FIGURA 3.1: Interacción al enviar el comando **Marco**

todas las respuestas. Al final dicho tiempo de espera, se retornan los resultados y el resto de respuestas son ignoradas.

3.2. Aplicaciones construidas sobre MarcoPolo

Nombre	Agente emisor	Función	Información	Respuesta esperada	Protocolo y puerto
Marco	Marco	Descubrir todos los nodos presentes en la *malla*	Únicamente se incluye el nombre del comando	Un comando *Polo* por cada nodo disponible en la red, incluyendo como parámetros opcionales información sobre el nodo o *ninguna* si no existe ningún nodo disponible.	UDP *multicast* al puerto 1338.
Polo	Polo	Informar a un nodo de la existencia del emisor	Información sobre el nodo opcional (servicios disponibles, información sobre el nodo o la instancia de Polo...)	*Ninguna*	UDP *unicast* al puerto efímero del mensaje de pregunta.
Request-For	Marco	Conocer todos los nodos que ofrecen un servicio identificado por su nombre único en el sistema	Identificador unívoco del servicio a descubrir	**OK** con información opcional sobre el nodo o el servicio	UDP *multicast* al puerto 1338.
OK	Polo	Comando utilizado para emitir una respuesta a una petición, siendo la información de interés contenida en los parámetros de respuesta.	Respuesta a un comando con la información solicitada	*Ninguna*	UDP *unicast* al puerto efímero de la pregunta.
Services	Marco	Descubrir todos los servicios ofrecidos por un nodo	No se envía información adicional con el comando	**OK** con una lista de los identificadores del servicio o *ninguna* si el nodo no está en la red.	UDP *unicast* al puerto 1338.

Apéndice A

An Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus at pulvinar nisi. Phasellus hendrerit, diam placerat interdum iaculis, mauris justo cursus risus, in viverra purus eros at ligula. Ut metus justo, consequat a tristique posuere, laoreet nec nibh. Etiam et scelerisque mauris. Phasellus vel massa magna. Ut non neque id tortor pharetra bibendum vitae sit amet nisi. Duis nec quam quam, sed euismod justo. Pellentesque eu tellus vitae ante tempus malesuada. Nunc accumsan, quam in congue consequat, lectus lectus dapibus erat, id aliquet urna neque at massa. Nulla facilisi. Morbi ullamcorper eleifend posuere. Donec libero leo, faucibus nec bibendum at, mattis et urna. Proin consectetur, nunc ut imperdiet lobortis, magna neque tincidunt lectus, id iaculis nisi justo id nibh. Pellentesque vel sem in erat vulputate faucibus molestie ut lorem.

Quisque tristique urna in lorem laoreet at laoreet quam congue. Donec dolor turpis, blandit non imperdiet aliquet, blandit et felis. In lorem nisi, pretium sit amet vestibulum sed, tempus et sem. Proin non ante turpis. Nulla imperdiet fringilla convallis. Vivamus vel bibendum nisl. Pellentesque justo lectus, molestie vel luctus sed, lobortis in libero. Nulla facilisi. Aliquam erat volutpat. Suspendisse vitae nunc nunc. Sed aliquet est suscipit sapien rhoncus non adipiscing nibh consequat. Aliquam metus urna, faucibus eu vulputate non, luctus eu justo.

Donec urna leo, vulputate vitae porta eu, vehicula blandit libero. Phasellus eget massa et leo condimentum mollis. Nullam molestie, justo at pellentesque vulputate, sapien velit ornare diam, nec gravida lacus augue non diam. Integer mattis lacus id libero ultrices sit amet mollis neque molestie. Integer ut leo eget mi volutpat congue. Vivamus sodales, turpis id venenatis placerat, tellus purus adipiscing magna, eu aliquam nibh dolor id nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed cursus convallis quam nec vehicula. Sed vulputate neque eget odio fringilla ac sodales urna feugiat.

Phasellus nisi quam, volutpat non ullamcorper eget, congue fringilla leo. Cras et erat et nibh placerat commodo id ornare est. Nulla facilisi. Aenean pulvinar scelerisque eros eget interdum. Nunc pulvinar magna ut felis varius in hendrerit dolor accumsan. Nunc pellentesque magna quis magna bibendum non laoreet erat tincidunt. Nulla facilisi.

Duis eget massa sem, gravida interdum ipsum. Nulla nunc nisl, hendrerit sit amet commodo vel, varius id tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor est. Suspendisse ultrices tincidunt metus eget accumsan. Nullam facilisis, justo vitae convallis sollicitudin, eros augue malesuada metus, nec sagittis diam nibh ut sapien. Duis blandit lectus vitae lorem aliquam nec euismod nisi volutpat. Vestibulum ornare dictum tortor, at faucibus justo tempor non. Nulla facilisi. Cras non massa nunc, eget euismod purus. Nunc metus ipsum, euismod a consectetur vel, hendrerit nec nunc.

Bibliografía

[1] U. de Salamanca, "Titulación y programa formativo - grado en ingeniería informática." http://http://www.usal.es/webusal/files/Grado_en_Ingenieria_Informatica_2014_1%C2%AA%20parte-actualizado%202-10-14.pdf, 10 2014.

[2] N. W. Group, "Comment on rfc 4516 - lightweight directory access protocol (ldap)," *RFC*, June 2006.