Implementación de un clúster de red Ad-Hoc para la adquisición de datos en la coordinación de redes logísticas

Julian Felipe Latorre Ochoa

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Industrial y de Sistemas
Ciudad, Colombia
2015

Implementación de un clúster de red Ad-Hoc para la adquisición de datos en la coordinación de redes logísticas

Julian Felipe Latorre Ochoa

Trabajo de investigación presentado como requisito parcial para optar al título de:
Ingeniero de Sistemas

Director:
Ph.D. Jorge Eduardo Ortiz Triviño
Codirector:
Ph.D. Wilson Adarme Jaimes

Línea de Investigación:
Telecomunicaciones y Sistemas Distribuidos
Grupos de Investigación:
TLÖN
SEPRO

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Industrial y de Sistemas
Ciudad, Colombia
2015

"-Es posible- proyectar de nuevo las instituciones de un gobierno de acuerdo a los principios y prácticas de la cibernética, lo cual consistirá en transferir al individuo la decisión de decidir; que el pueblo por sí mismo tome el control de la ciencia por medio de procesos democráticos; suministrando a aquel o a los gobiernos, nuevos canales de comunicación, nuevos sistemas de educación y nuevos sistemas de difusión"

Stafford Beer

Agradecimientos

A lo largo del desarrollo del presente trabajo, han pasado por mi vida entre charlas, debates, jornadas de estudio y en general valiosos momentos de compartir, muchas personas que desde sus propios saberes han aportado de forma inconmensurable para la concepción del mismo. Es posible que muchas de ellas no recuerden siquiera tales momentos así como para mí también resultan innumerables, sin embargo todos ellos, indispensables para el logro actual. Las anteriores razones, me ha llevado a considerar especiales agradecimientos a los siguientes compañeros (en el más cálido sentido de la palabra) e instituciones:

- A mis amigos y amigas de la Facultad de Ciencias Agrarias de la Universidad Nacional de Colombia, quienes con su compañía y pensamiento crítico han acompañado mi formación como sujeto político, académico y propositivo.
- A mis amigos y amigas del la "comunidad mundial de software libre" quienes con su compromiso altruista por la liberación del conocimiento han inspirado mis más sinceros reconocimientos.
- A mis amigos y amigas del grupo de investigación SEPRO de la Universidad Nacional de Colombia, quienes en cabeza de su director, el profesor Wilson Adarme, me han brindado su apoyo incondicional en los últimos años con sus trabajos de alta calidad, compromiso académico y calor humano.
- A mis amigos del grupo de investigación TLÖN quienes encabezados por su director, el profesor Jorge Eduardo Ortiz, fueron con seguridad la pieza clave para el exitoso desarrollo de la presente investigación. En particular a Henry Zarate y Joaquín Sánchez que de forma perseverante guiaron el desarrollo general de la investigación en su contenido técnico-científico.
- De manera especial a mis amigos Nelson David Sotelo, Juan Carlos Rubiano y Diego Salgado quienes concretamente me ayudaron a escribir más de una de las páginas contenidas en el presente documento.
- A mis padres porque de ser necesario escribir aquí los agradecimientos, el documento duplicaría su extensión.

Al pueblo Colombiano y su valentía para pervivir a pesar de ello…

Resumen

Este trabajo se centra en la implementación de un clúster de red computacional Ad-Hoc, para la adquisición y gestión *descentralizada* de información presente al interior de cadenas logísticas de suministro agropecuarias en miras a su coordinación. Se enuncia un modelo de trabajo sobre el cual se desarrolla la investigación e implementación del protocolo de enrutamiento B.A.T.M.A.N. (*The Better Approach To Mobile Adhoc Networking*) para establecer de manera espontánea y sin infraestructura un clúster de red móvil y potencialmente *heterogéneo* de dispositivos Raspeberry Pi y computadoras portátiles. Dichos nodos, actúan de manera dual dentro de la red haciendo las veces de proveedores y consumidores (clientes y servidores) de información, configurándose como una Red Inalámbrica de Sensores (WNS por sus siglas en ingles) que declara y presta sus servicios de sensado mediante técnicas de multidifusión y auto-direccionamiento.

Palabras clave: Agroindustria, Logística, Cadena de Suministro, Tecnologías de información y comunicación (TIC), Redes Ad-Hoc, Raspberry Pi.

Abstract

This work focuses on the implementation of a computational Ad-Hoc network cluster, for decentralized management and acquisition of information present within agricultural logistic supply chains in order to their coordination. A working model on which the research and implementation of "The Better Approach To Mobile Adhoc Networking" (B.A.T.M.A.N.) routing protocol is enunciated to establish a spontaneous and infrastructure less mobile networking cluster of potentially heterogeneous Raspeberry Pi devices and laptops. Those nodes acts dually within the network, acting as providers and consumers (clients and servers) of information, becoming a Wireless Sensor Network (WSN) that declares and serves sensing as a service trough multicast and self-addressing techniques.

Keywords: Agribusiness, Logistics, Supply Chain, Information and Communication Technology (ICT), Ad- Hoc Networks, Raspberry Pi.

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

INTRODUCCIÓN

1.1 Antecedentes

El grupo de investigación Sociedad, Economía y Productividad SEPRO, línea logística – Supply Chain Management (SCM), es un grupo multidisciplinario con integrantes de nivel doctoral, de maestría y pregrado, que desde el año 2003 realiza investigación en los temas de logística y SCM. La experiencia investigativa en el sector agroindustrial le ha permitido al grupo llegar a niveles de conocimiento detallado de la operación logística de estas cadenas, que muestran las debilidades del sector, en lo relacionado con la operación logística, la generación de nuevo conocimiento basado en las características propias del contexto estudiado [1], y los futuros campos de investigación que tienen cabida de acuerdo a las necesidades evidenciadas.

Es el caso del proyecto "*Propuesta Metodológica Para Coordinar Procesos Logísticos de Producción y Distribución de Cacao y Plátano en las Zonas de Caricare y Caño Limón"*[2], desarrollado por el grupo SEPRO durante el primer semestre del 2012, en el que una de las áreas evidentes de investigación futura, es la del manejo de la información estratégica en los procesos de carga y distribución de productos perecederos (logística del manejo pos cosecha),

El grupo también identifica la relevancia de la investigación en logística agroindustrial a través de las necesidades expresadas por actores del sector servicios, particularmente los operadores logísticos, que en el establecimiento de estrategias de trabajo conjunto con la Universidad Nacional de Colombia por medio del grupo SEPRO, destacan la urgencia que tienen de mejorar los procesos de manipulación de bienes (cargue, descargue, almacenamiento temporal/permanente, entre otros), que actualmente representan un problema debido a las pérdidas en tiempos y costos que estas actividades generan por ausencia de modelos y plataformas tecnológicas suficientes para gestión de la información en la coordinación de los agentes de la cadena, particularmente de productos perecederos, y que se encuentre la mejor combinación entre los costos de transporte, inventarios y producción [3].

El desarrollo de un proyecto de investigación en nuevas técnicas computacionales para la gestión de información distribuida, es también pieza clave en los propósitos investigativos del grupo de investigación TLON, quienes han planteado en su proyecto insignia con suficiente rigurosidad y detalle la necesidad de implementar sistemas de hardware que soporten la ejecución de aplicaciones en redes distribuidas de generación espontanea tipo Ad-Hoc las cuales resultan promisorias en la gestión de redes móviles, como las requeridas en las zonas de operación logística agropecuaria.

Así pues, el trabajo podrá representar un pequeño avance en el estado del arte en el que se han identificado brechas relacionadas con la contextualización de modelos logísticos en operaciones logísticas del sector primario y especialmente, en cadenas de productos perecederos [4], y de manera más general, brechas en los avances de la implementación de modelos de planeación de sistemas logísticos agropecuarios [5]–[7]. Así mismo, el tema de investigación responde a necesidades explícitas del sector agropecuario en Colombia, en el que las pérdidas por operaciones logísticas ineficientes –entre ellas el manejo de carga– se encuentran entre el 25% y 30% [8].

1.2 Justificación

1.2.1 Aspectos Económicos y Productivos

En las últimas décadas, la economía del departamento de Arauca ha sido dinamizada principalmente por la extracción petrolera, la cual representó para 2011 un ingreso cercano a los $4'008.000.000.000, sin embargo según palabras del gobernador del departamento José Facundo Castillo Cisneros, "a pesar de constituir el 61% del producto interno bruto, esta economía de "distracción" no ha generado los encadenamientos productivos necesarios para la construcción de una economía endógena catalizadora de oportunidades para las nuevas generaciones de araucanos. Por el contrario, más bien ha generado conflictos internos y ha promovido presiones migratorias, exigiendo grandes esfuerzos presupuestales para proveerles servicios del Estado."[9].

Es así como el desequilibrio en la economía y por consiguiente en las cadenas logísticas que están a su servicio, acompañado de la falta de mecanismos existentes, viables y accesibles de veeduría tanto ciudadana como gubernamental del mercado, impide una justa retribución a los varios eslabones de las cadenas de valor y producción como sucede en el caso de las diferentes cadenas agropecuarias del departamento.

Aunque los presupuestos públicos están fuertemente respaldados por la economía petrolera, la mayor parte de la población productiva se encuentra vinculada de manera directa o indirecta a las actividades asociadas con a la producción agropecuaria, sobre todo en las zonas rurales del departamento. Es por ello que las problemáticas presentes en los encadenamientos productivos impacta de manera directa la mayor parte de la población, tal como se evidencia al observar los indicadores sociales reportados en 2012 entre los cuales sobresale un índice de Necesidades Básicas Insatisfechas (NBI) de 35,6%, muy por encima del promedio nacional del 27,7%.

En Arauca, se erige en un segundo lugar de importancia económica, la producción agropecuaria, responsable del ingreso de alrededor de 841 mil millones de pesos anuales (14,1% del PIB del departamento), de los cuales, los 329 mil millones en el caso del cultivo de productos agrícolas, equiparan en magnitud a los 338 mil millones percibidos por alrededor de un millón de hectáreas dedicadas a la ganadería en la región [10].

Las oportunidades y capacidades para este sector son significativas y están aún sin explotar en su totalidad. Es tal la vocación agropecuaria del mismo, que a pesar de su primer puesto reconocido a nivel mundial en cuanto a calidad del grano de cacao [11] y los varios reconocimientos a nivel nacional como primer productor platanero, sus métodos en los procesos productivos aún utilizan tecnologías bastante precarias [12] y más allá, muchos procesos productivos deseables, permanecen inexistentes o en proceso de creación [13]. El departamento tiene así "amplias fortalezas en el sector primario, pero infortunadamente no ha logrado construir encadenamientos hacia adelante relevantes, como se infiere de la baja participación de los sectores de industria, comercio y servicios"[9].

Esta destacada producción agropecuaria, ha impulsado el surgimiento de importantes agremiaciones productivas y una mirada atenta por parte del gobierno departamental y su secretaria de desarrollo agrícola hacia dicho sector. Ejemplo de ello es la alta capacidad de gestión de las distintas organizaciones productivas presentes en la región, entre las que se destacan CODEPLAR (Comité Departamental de Plataneros de Arauca) y sus diferentes representaciones gremiales a nivel municipal (ASOPLAFOR, APTA, ASOPLASA Y ASDEPLAR) quienes aglutinan cientos de productores así como la cooperativa agraria del Sarare (COAGROSARARE) la que cuenta con cerca de 1800 asociados para la fecha.

Dichas organizaciones son responsables de mantener en alto algunas de las mejores cifras productivas del país, como sucede en el caso del plátano y cacao, donde sus índices de rendimiento por hectárea en el departamento, han superado considerablemente los promedios nacionales desde hace varios años y respondieron en 2012 por el

12,5% y 18,2% de la producción total nacional respectivamente [14].

De manera unificada las agremiaciones productivas, han logrado posicionar en la agenda gubernamental proyectos de distinta índole para el beneficio de sus asociados en temas como vivienda rural, pavimentación y rehabilitación de vías terciarias y secundarias, ampliación y construcción de redes eléctricas acueductos entre otros. Es así como gracias al esfuerzo conjunto de las distintas fuerzas vivas en la producción de la región se vienen consolidando serios procesos y proyectos encaminados a viabilizar la reconfiguración de las dinámicas del capital material e inmaterial del departamento, de tal forma que el principal receptor de sus beneficios económicos sea el departamento en sí mismo y sus habitantes.

No obstante, las políticas de comercio, agro industriales y logísticas del país en general siguen en la búsqueda por generar condiciones favorables para los mercados agrícolas y los pequeños y medianos productores pero sin mayores resultados hasta el momento. Ejemplo de ello son las drásticas caídas de comercio exterior en lo que se refiere a los renglones agropecuarios. Siendo Arauca un territorio de significativa importancia para las relaciones internacionales dada su condición fronteriza, una caída del 95,6% en la exportación de productos agropecuarios y del 80,4% en la de productos agroindustriales entre 2012 y 2013 así lo evidencia [15]. Así mismo, los distintos acuerdos comerciales suscritos por el país aún no generan los beneficios esperados en cuanto a potencialidad de explotación de productos agropecuarios, a excepción del incremento en las exportaciones de petróleo, las que alcanzaron una cifra cercana a los 110 millones de dólares FOB para el año 2013 [10], las políticas macroeconómicas generales aún están en deuda de impulsar las economías locales y sus potencialidades. Más aún, los acuerdos suscritos exigen la modernización e implementación de nuevas tecnologías que conduzcan a una mayor eficiencia en los procesos productivos.

En las cadenas logísticas de distribución, se encuentra en su transcurrir un problema muy común, el cual dificulta la buena articulación entre sus eslabones. Este es el conocido problema de la asimetría en la información que poseen tanto productores, acopiadores e intermediarios como industriales cuando realizan la transacción física del producto entre sí; factores como la calidad y cantidad del producto, su manejo agronómico en el cultivo, los precios de compra y venta en el mercado nacional y la demanda agregada de los productos, entre otros, son ejemplos de la información que se comparte de manera asimétrica a lo largo de la cadena de valor.

En el programa del seminario internacional "Políticas para la Agricultura en América Latina y el Caribe: Competitividad, Sostenibilidad e Inclusión Social", se argumenta que "en muchos países los gobiernos y los actores privados han empezado a impulsar sus estrategias de desarrollo agrícola y rural en forma conjunta y con una visión de largo plazo, generando una nueva dinámica de diseño e implementación de políticas. En algunos países se han diseñado planes de acción por agro - cadenas, sustentados en alianzas público - privadas, para mejorar su competitividad. Estos planes son una respuesta a la existencia de grandes asimetrías entre los actores que operan a nivel microeconómico, que obligan a aplicar enfoques que se apartan de los modelos de economía pura, que postulan que todos los mercados se equilibran en razón de la plena racionalidad de los individuos y las empresas, la completa disponibilidad de información y la coordinación instantánea de los actores gracias a los ajustes simultáneos de los precios y las cantidades" [16].

Sin embargo, en el escenario productivo Colombiano, los efectos negativos de las asimetrías en las cadenas productivas aún se sienten con fuerza, los sobre costos presentes en las mismas son comúnmente reconocidos y asumidos por los eslabones más iniciales, recayendo sobre éstos las fluctuaciones en los costos de producción y en precios de mercado, lo que hace a los productores vulnerables frente a la dinámica inherente al proceso productivo. Contrariamente se mantienen las ventajas para los eslabones más avanzados dentro de la cadena. Ejemplo de ello es el caso de los combustibles, en el cual, el valor asociado a ellos, sumado a sus constantes incrementos, es asumido una y otra vez por improvisadas alianzas entre productores y transportadores para lograr la salida de sus productos al mercado nacional, asumiendo tales sobre costos, y aceptando que aguas abajo en la cadena de valor, las empresas comercializadoras y grandes superficies a través de sus agentes de intermediación, eviten asumirlos en sus relaciones contables. En resumen, la desorganización presente en el proceso productivo de estas cadenas, genera consecuentemente una redistribución inequitativa entre los diferentes agentes que hacen parte de la misma.

El proceso de negociación no resulta ajeno a las lógicas expuestas anteriormente y es presa fácil de tales particularidades presentes en la cadena de valor, cada una de las articulaciones de la cadena es susceptible a fortalecer el fenómeno de asimetría en la información y por tanto en la negociación, esto a falta de un sistema transparente que logre conversar y reconciliar los intereses de los involucrados en cada una de las etapas del suministro. La información cobra entonces un lugar privilegiado para el provecho de su depositario, a costa en ciertos casos, del bienestar, eficiencia y estabilidad misma de la cadena en términos generales.

1.2.2 Aspectos Informáticos y Tecnológicos

La información constituye un flujo principal tanto en la cadena logística como económica de cualquier producto de consumo. El acceso a ella en el contexto agropecuario, está aún estrechamente ligada a los agentes que participan en los eslabones más avanzados de la cadena productiva, generándose por consiguiente escenarios asimétricos a lo largo de las mismas, los cuales redundan en condiciones desfavorables para el juego de negociación entre los agentes participantes.

El problema de la información y en particular problema del acceso a la información es un problema digno de estudiar a profundidad al interior de las cadenas de suministro en cuestión. En los procesos de producción primaria, una de las mayores reivindicaciones de sus agentes es la del acceso a información técnico-científica para el buen manejo de sus cultivos, esta sin embargo es solo una de las muchas componente de la matriz de información solicitada por dicho eslabón, la información asociada a los insumos agrícolas, mano de obra, transporte, almacenamiento, etc., son claros ejemplos de importantes nichos informáticos requeridos por el eslabón primario.

En la actualidad, la gestión de la información productiva es altamente heterogénea. Los esfuerzos e iniciativas emprendidas por los Ministerios de Agricultura y de Tecnologías de la Información y Comunicaciones, empiezan a adquirir relevancia en el ámbito nacional agropecuario continuando sin embargo sin solucionarse los varios problemas de acceso a la red por una vasta mayoría especialmente cuando se trata de los mismos productores agropecuarios, entre las iniciativas de mayor relevancia al respecto podemos encontrar al sistema de información “Agronet” [17], el cual mediante diferentes estrategias recolecta información de alto valor para el soporte en la toma de decisiones de procesos críticos que busca gestionar. Se tiene también al Sistema de Alerta Temprana del valle de Aburrá - SIATA[18], el cual cuenta en la actualidad con diversas redes para el monitoreo de lluvia (71 sensores); variables meteorológicas, como temperatura y dirección y velocidad de viento (7 estaciones); Nivel de las Quebradas (8 sensores); Humedad del Suelo (en calibración) y una red de Cámaras en LiveStreaming (7 cámaras) enfocado a la toma de decisiones en la prevención, atención y gestión del riesgo en sus áreas de influencia [18].

Distintas instituciones en sus planes de competitividad y desarrollo incluyen en sus informes la anotación que para lograr generar condiciones favorables en el desarrollo del sector agropecuario es preciso, entre otras:

- Coordinar esfuerzos inter organizacionales y gubernamentales a lo largo de las cadenas agropecuarias.
- Crear nuevas redes de comercialización y distribución y fortalecer las existentes.
- Incentivar la asociatividad.
- Mejorar la infraestructura de la región.
- Impulsar la educación, la ciencia, la tecnología y la innovación.
- Desarrollar las cadenas productivas.
- Utilizar de manera transversal las tecnologías de la información y la comunicación (TIC) en el desarrollo de los puntos anteriores.

Se evidencia la importancia en un sistema de producción, de generar prácticas de gestión que soporten la coordinación de sus actividades productivas y logísticas, en ese sentido Thakur y Hurbugh proponen la trazabilidad como una de las estrategias de gestión [19], las cuales permiten responder a los requerimientos de la calidad de una cadena productiva; y aunque la trazabilidad pueda estar asociada a desarrollos tecnológicos y sistemas de información, según Rong y Grunow el diseño de sistemas logísticos es un medio pertinente para establecer la trazabilidad de productos agropecuarios, específicamente en los contextos productivos de los países en vía de desarrollo [20].

La propuesta de los sistemas logísticos, como medio para la trazabilidad en agro-cadenas, se basa en el soporte que brindan para el diseño de relaciones entre los agentes de las cadenas, pues permiten establecer el flujo de los productos desde un origen hasta un destino determinados, a través de canales y medios específicos. De esta manera, en cualquier punto de la red es posible reconocer tanto la procedencia del producto y su destino como sus características asociadas. Esto, facilita la ejecución de medidas que busquen garantizar la inocuidad, sanidad y calidad de los alimentos comercializados a través de la red y encontrando las fuentes de posibles perturbaciones, lo que nos garantiza un mejoramiento constante de la cadena productiva.

Dentro de la estrategia del mejoramiento de cadenas productivas y articulación de actores se comprende la agrupación geográfica de planes y actividades económicas en el departamento Araucano, identificando zonas estratégicas potenciales para la ubicación de centros productivos, centros de acopio, corredores económicos, centros de transformación, distribución y venta al detal de los productos que hacen parte de las apuestas productivas [2].

Estas decisiones se enmarcan dentro de lo que Cordeau, *et al* (2006)[21] denominan como decisiones estratégicas y táctico/operativas para el diseño de la red de la cadena de suministro, con la particularidad de tener en cuenta las

características geográficas, climáticas, biológicas, políticas, sociales y económicas de la región, consolidando así lo que se definió como un SLI que sirva de base para el fortalecimiento de las apuestas productivas y para la definición de políticas departamentales y municipales acordes al desarrollo y establecimiento de potenciales productivos que busquen el bienestar del sector agropecuario del departamento en su totalidad.

Una red móvil ad-hoc o MANET, del inglés Mobile Ad-hoc Networks [22] es una colección de nodos inalámbricos móviles que se comunican de manera espontánea y auto organizada constituyendo una red temporal sin la ayuda de ninguna infraestructura preestablecida (como puntos de acceso WiFi o torres de estaciones base celulares con antenas 2G, 3G o 4G) ni administración centralizada [22]. Una de las principales ventajas de una MANET es la posibilidad de integrarla a una red de infraestructura con diferentes fines, tal como el acceso a aplicaciones de una organización desde un dispositivo móvil [23].

Es aquí donde las MANET adquieren un valor especial en los fines anteriormente mencionados, más aún si se tiene en cuenta que tradicionalmente, la topología de una red rural ha sido fija [24]; esto significa que cada nodo de la red tiene una posición permanente; y que los enlaces requieren distancias relativamente grandes, esto último hace que la cantidad y altura de los obstáculos sea considerable, por lo tanto para constituir un enlace de larga distancia (mantener línea de vista entre las antenas), se requiera una altura suficiente para las torres que las soportan, de tal forma que estén sobre los obstáculos (arboles, construcciones y terreno). El costo de las torres, es proporcional a su altura y está relacionado con material de construcción, por ejemplo para un enlace de entre 7-8 Km (distancias típicas) se necesitarían torres de entre 30 m y 45 m con costos de entre 25 y 38 millones de pesos colombianos. Este costo es de varios órdenes de magnitud mayor que el de los equipos de comunicaciones, de manera que el principal problema de construcción de redes rurales radica en lograr una topología que minimice el costo su infraestructura fija [25].

1.3 Objetivo General

- Implementar un clúster de red Ad-Hoc de dispositivos de adquisición de datos, con el fin de obtener información presente en la cadena logística de un producto agrícola.

1.4 Objetivos Específicos

- Implementar un escenario de comunicación serial entre un dispositivo Raspberry Pi y el microcontrolador MSP-430 para la adquisición de datos.

- Realizar una simulación del comportamiento de la red propuesta en NS-3.

- Implementar una red Ad-hoc entre Raspberry Pi’s mediante la utilización de equipos de transmisión y recepción, bajo el estándar IEEE 802.11 en la banda de 2,4 GHz (Wifi).

2. MARCO TEÓRICO

2.1 Cadena de Suministro

Sobre gestión de la cadena de abastecimiento, el Council of Supply Chain Management (CLSMP), conceptualiza a la cadena de suministro como:
1) Iniciando con materias primas no procesadas y finalizando con los bienes terminados siendo consumidos por el cliente final, la cadena de suministro enlaza muchas firmas.
2) Los intercambios de materiales e información en el proceso logístico se extienden desde la adquisición de materias primas hasta la entrega de productos terminados al consumidor final. Es así como los vendedores, proveedores de servicio y clientes son los vínculos al interior de la cadena de suministro.

El concepto de cadena de suministro se ha discutido ampliamente y se destacan las siguientes apreciaciones:

Para Christopher (1998)[26] es una red de organizaciones conectadas e interdependientes que trabajan mutua y cooperativamente para controlar, administrar y mejorar el flujo de material e información desde proveedores a consumidores finales.

Johansson (2002)[27]: sistema cuyas partes constitutivas incluyen proveedores de materias primas, instalaciones para la producción, servicios de distribución y clientes vinculados mediante el flujo de materiales hacia adelante y el flujo de información hacia atrás.

Waters (2003)[28]: conjunto de actividades y organizaciones en donde los materiales se mueven a lo largo de su ruta desde proveedores iniciales hasta consumidores finales.

Sucky (2009)[29]: red de diferentes locaciones dispersadas geográficamente, en donde materias primas, productos intermedios o productos terminados son transformados, y existen vínculos de transporte que conectan las locaciones.

2.1.1 Estructura de la cadena de suministro

Existe un conjunto de elementos relacionados con la estructura de la cadena y el funcionamiento de los vínculos entre sus miembros, que afectan significativamente la manera, la funcionabilidad y operación de la misma. Éstos corresponden a:
Cadenas Centralizadas o descentralizadas: una cadena centralizada consiste en un sistema de múltiples niveles, en donde existe un sólo tomador de decisiones o un equipo de toma de decisiones, que tiene la autoridad de decidir por todos los niveles (Chu, 2007)[30]. En este tipo de cadenas de suministro se coordinan las respectivas estrategias de producción, inventario y distribución, que generen un mejor desempeño para el sistema.

El caso contrario, que se asemeja mucho a las situaciones presentes en la realidad de las cadenas de suministro, es aquel en donde todos los miembros actúan independientemente para mejorar su desempeño individual, de manera que cada uno de ellos cuenta con un conjunto de información que le permite tomar ciertas decisiones con el fin de optimizar algún objetivo de su propio interés.

Relaciones de poder al interior de la cadena de suministro: La integración al interior de la cadena de suministro sólo puede tener éxito en el marco de la confianza y de la existencia de un adecuado mecanismo de administración de las relaciones de poder al interior de la cadena; puesto que, por un lado, la ausencia de confianza hace que las firmas se abstengan de colaborar con sus asociados (Fawcett et al. 2002)[31]. Por otra parte, los asociados en una cadena de suministro están condicionados por el poder que tengan para realizar inversiones que permitan dar sostenimiento a sus vínculos de cooperación, de acuerdo con Cox (2001)[32]. La confianza en este contexto se define como "el grado en el que una firma cree que su asociado en la cadena de suministro es honesto y/o benévolo" (Geyskens 1998)[33].

Ahora bien, en cuanto a las relaciones de poder, éste es conocido como "la habilidad que presenta un miembro de la cadena para influenciar el comportamiento y las decisiones de otros miembros" (Cox, 2001)[32].

2.1.2 Coordinación en la cadena de suministro

La función de integración entre diferentes tipos de firmas se orienta a dar cumplimiento a las necesidades de demanda impuestas por el mercado, bajo un ambiente de complejidad en donde cada firma busca cumplir con sus objetivos particulares.

Ello se evidencia en los planteamientos de Wang (2010)[30], quien explica como existen cuatro tipos de organizaciones en la

coordinación de la cadena de suministro, cada una con un conjunto de objetivos diferentes y particulares a su quehacer, los cuales corresponden a:

a. Productores: su objetivo principal es optimizar el sistema de planeación de la producción y el grado de utilización de los recursos.

b. Distribuidores: pretenden crear un balance entre aprovisionamiento y demanda, negociar con los agentes de aprovisionamiento y demanda, y minimizar los inventarios y el agotamiento de existencias en toda la cadena de suministro.

c. Proveedores y minoristas: pretenden maximizar la satisfacción de sus clientes mediante la minimización de sus propios niveles de agotamiento de existencias.

d. Proveedores de servicio (terceros): buscan cumplir con la demanda de los fabricantes, distribuidores, proveedores y minoristas.

En tanto, estos actores buscan mejorar sus resultados en cuanto a aprovisionamiento, producción y distribución, se ha generado la necesidad de aplicar estrategias de coordinación a lo largo de la cadena de suministro. En este contexto se puede entender por coordinación a "la acción de administrar las dependencias entre entidades y el esfuerzo conjunto de las mismas, trabajando hacia el cumplimiento de unas metas conjuntamente establecidas"[31].

Los procesos logísticos han sido observados desde dos perspectivas diferentes; la gestión de procesos, y la aplicación de modelos matemáticos explicativos. A continuación se encuentra una breve definición de los procesos, así como de los principales modelos desarrollados en la integración y coordinación de los mismos.

Perspectivas sobre coordinación de inventarios. Arshinder et al.[32] expone un completo resumen de las diferentes perspectivas que se han construido alrededor de la coordinación de inventarios durante los últimos 14 años, tales como: Narus et al. (1996)[33]. La coordinación como cooperación entre firmas independientes para cumplir demanda; Lambert et al.[34]. Relación entre firmas para administrar riesgo y compartir recompensas; Ballou et al. (2000)[35]. Integración a través de grados de autoridad y responsabilidad organizacional; Lee.[36]. Medios para rediseñar el

flujo de recursos en la cadena para alcanzar un mejor desempeño; Simatupang et al.[37]. Relaciones de mutualismo para alcanzar objetivos de la cadena, y particulares; Larsen et al.[38]. Desarrollo de actividades conjuntas y sincronizadas para determinar procesos de producción y aprovisionamiento.

2.2 Internet: la vía de integración por excelencia

Granja Florez [39], define que la World Wide Web está planteada y configurada "como una combinación de hardware (ordenadores interconectados por vía telefónica o digital) y software (protocolos lenguajes que hacen que todo funcione) es una infraestructura de redes a escala mundial que conecta a la vez a todos los tipos de ordenadores".

Así mismo [39] comenta que el internet para las PyMes es la llave que descubre y enfoca los criterios de equilibrio y solidez en la creciente globalización, resaltando el comercio constante, la reducción de costos y el encuentro con nuevos clientes y potenciales socios. Tal como explica el estudio de mercado realizado por la Red Global de Exportación y presentado en el documento Internet y las nuevas Tecnologías como herramientas para las PyMes exportadoras (2009) "Las PyMEs perciben que Internet ha impactado decidida y positivamente en la productividad de sus negocios, así como en la reducción de costos y gastos."[40].

El articulo Internet y la economía real (2007) escrito por Sukhinder Cassidy en el cual recopila lo expuesto por el Consejo de Asesores e la Casa Blanca enfoca a "las empresas que invierten más del promedio en IT logran aumentos de productividad hasta 4 veces más que empresas que invierten menos del promedio en esta área [41]. En el mismo artículo, se explica que "hoy el uso de Internet en las PyMEs estimula los canales de comunicación en el ecosistema de negocios -compradores, vendedores y socios- y aporta aplicaciones como correo electrónico, manejo de documentos, hojas de cálculo, sistemas de búsquedas corporativas y otras herramientas de comunicación y colaboración que impactan en la productividad de los negocios". Se afirma también, que el internet "está facilitando a los países emergentes la posibilidad de exportar sus productos, servicios y conocimiento hacia el resto del mundo y es también un espacio que integra la cadena de valor, a partir del trabajo colaborativo y el conocimiento compartido" [41].

Paola Morales en su artículo ¿Por qué usar Internet en su empresa? (2007) manifiesta que el progreso de las PyMes acompañado del Internet han experimentado el "desarrollo de nuevos productos, el mercadeo, la compra, la distribución" a la mismas vez que "el servicio al cliente se han visto mejorados, sin importar las distancias geográficas. Además, los negocios por outsourcing vía la Internet, han reducido el costo tanto dentro de las empresas como entre las empresas[42].

2.3 La Agrónica: Una herramienta para la explotación racional de los recursos

Según el profesor Mauro Florez Calderón de la Universidad Nacional de Colombia, existe un gran consenso entre los economistas, los científicos y los políticos en que la forma de producción industrial ha hecho crisis permitiendo la aparición de la posmodernidad, era posindustrial o era informacional; crisis que se manifiesta en diferentes formas, como los problemas energéticos pues es evidente que la sociedad de consumo y de despilfarro de bienes industriales requiere de grandes cantidades de energía, de hidrocarburos y éstos se hallan en los países de la periferia en cantidades limitadas como en el Golfo Pérsico, México y Venezuela entre otros, generando conflictos y tensiones internacionales permanentes[43].

El mismo profesor anota que, la Agrónica tiene enormes posibilidades en la explotación racional de los recursos naturales, marítimos, terrestres y atmosféricos, rebasando ampliamente el campo de acción de la informática agropecuaria, pues ésta se reduce al procesamiento de información en las actividades agrícolas y pecuarias, es decir, la informática agropecuaria es una manifestación de la Agrónica pero no su sinónimo.

La Agrónica es ampliamente empleada en aplicaciones agropecuarias, bosques, hidrología, recursos energéticos, oceanografía, geología, cartografía, determinación de la vocación de la tierra, predicción de los fenómenos meteorológicos y por lo tanto su comprensión, predicción de las calamidades naturales como inundaciones y desplazamientos de ciclones, lo cual permite evacuar la población y evitar pérdidas de vidas humanas, la Agrónica ayuda a proteger nuestro patrimonio común, la Tierra, con toda su diversidad, el espacio ultra-terrestre y el patrimonio cultural.

Puntualiza también el profesor Florez que, en la agricultura, la existencia de bases de datos y de redes le permiten al agricultor determinar de una manera más racional el tipo de cultivo por el que ha de decidirse, pues, estará más enterado de la cantidad de tierra sembrada en el país con este producto, de los precios locales e internacionales del mismo, evitándose de esta forma la sobre oferta y por lo tanto la quiebra de muchos agricultores. La Agrónica nos proporciona las herramientas para determinar la composición de la tierra y los cultivos más aptos, se puede atenuar los efectos de las heladas y de las granizadas, proteger los cultivos contra las plagas, por ejemplo, usando señales electromagnéticas moduladas en amplitud o frecuencia con una portadora ultrasónica. La costosa maquinaria agrícola se está sirviendo de las tecnologías informacionales, así no es raro que en un tractor se incorporen sensores que midan sus diferentes variables para garantizar el buen estado del mismo, también se usan computadores y redes de comunicación para automatizar el trabajo de la maquinaria agrícola de manera que ésta funcione las 24 horas diarias sin intervención humana. Los microprocesadores controlan en los invernaderos todas las magnitudes como la temperatura, humedad y nutrientes. Los bancos de datos permiten la correcta distribución de los productos en todos los mercados locales evitando que se presente sobre oferta en unos y déficit en otros, afectando tanto a consumidores como productores.

"Gracias, en parte, a las telecomunicaciones, la informática y la electrónica el campesino raso y el industrial agropecuario se están integrando cada día más y más a la vida política, económica y cultural del país" [43].

2.4 El concepto de computación Ubicua

Pomares (2009) como parte de su participación en el en el Taller: Grandes Retos de Investigación Científica y Tecnológica en Tecnologías de Información y Comunicaciones en México Mayo 21 y 22, defiende que, "gracias a los grandes avances en telecomunicaciones, redes de computadoras, microprocesadores, dispositivos de almacenamiento, sensores, y la democratización del uso del Internet, entre otros, es posible el día de hoy incursionar en el desarrollo de servicios que antes sólo existían en el mundo de la ciencia-ficción. Bajo el cómputo ubicuo, los usuarios podrán acceder a servicios de información adecuados a la situación en la que se encuentran, adonde sea y cuando sea (at anywhere and anytime)" [44].

Por otra parte, Hernández et. Al [45], define que, el desarrollo de sistemas informáticos integrales que permitan el alcance de objetivos específicos en determinadas áreas sociales, llevan a desarrollar, crear o buscar nuevas y mejores herramientas computacionales que otorgan alternativas de apoyo, así como dar soluciones en diferentes procesos. La aplicación del cómputo ubicuo involucra varias ramas de las ciencias de la computación como el desarrollo de sistemas embebidos (sistemas que se pueden desarrollar mediante interfaces), la interacción hombre-máquina, los sistemas sensibles al contexto, las redes de computadoras, los sistemas distribuidos y los sistemas integrales. La Computación Ubicua según (Weiser 1991) supone la diseminación del sistema informático de manera que éste pasa a formar parte de productos cotidianos convertidos en objetos auxiliares e "inteligentes" gracias a las tecnologías ubicuas. Las tecnologías más relevantes para hacer realidad este paradigma son la identificación automática (Auto-ID), la localización y los sensores. Según [46] las etiquetas de código de barras, etiquetas de identificación por radiofrecuencia (RFID), etiquetas inteligentes (Smart Cards) o sistemas biométricos representan los mecanismos más utilizados para la identificación. La tecnología de código de barras está ampliamente extendida, no obstante la detección requiere visión directa entre el receptor y la etiqueta. La identificación por radiofrecuencia (RFID) no tiene esta limitación. Aunque se trata de una solución más sofisticada su uso aumenta a la par que sus costos de producción se reducen.

El uso de dispositivos móviles se encuentra en uno de sus puntos máximos, debido a que en la actualidad el desarrollo de aplicaciones ha evolucionado la operatividad de los mismos, y esta es realizada de una manera sencilla utilizando un lenguaje de programación, lo que permite el fácil manejo de sistemas embebidos para el intercambio y actualización de información con Bases de Datos y tecnologías WEB. En la estructura del sistema SISCA los dispositivos móviles son utilizados como una herramienta de comunicación y alimentación de información [45].

Es así como según Caballero et al., 2012, la informática y las comunicaciones han propiciado el desarrollo de procesos de gestión de información, disminuyendo el tiempo necesario de procesamiento y el costo de operaciones al aprovechar las ventajas de la automatización por medio de sistemas informáticos [47].

2.5 Redes Móviles Ad-Hoc

Una red móvil ad-hoc o MANET, del inglés Mobile Ad-hoc Networks [22] es una colección de nodos inalámbricos móviles que se comunican de manera espontánea y auto organizada constituyendo una red temporal sin la ayuda de ninguna infraestructura preestablecida (como puntos de acceso WiFi o torres de estaciones base celulares con antenas 2G, 3G o 4G) ni administración centralizada [22]. Una de las principales ventajas de una MANET es la posibilidad de integrarla a una red de infraestructura con diferentes fines, tal como el acceso a aplicaciones de una organización desde un dispositivo móvil [23].

Las principales características de una red MANET son:

- Terminales autónomos: Cada Terminal se comporta como un nodo autónomo que puede funcionar como emisor, receptor o enrutador.
- Funcionamiento distribuido: No existe ningún elemento central que se encargue de la gestión y el control de la red, todos los nodos son iguales y por lo tanto la gestión está distribuida.
- Enrutamiento multi-salto: Los paquetes realizan uno o varios saltos para llegar de la fuente al destino.
- Topología dinámica: Como no existe ninguna infraestructura fija y además los nodos son móviles, la topología de la red puede ser altamente cambiante, introduciendo una cierta complejidad al realizar el encaminamiento.
- Conexiones inalámbricas: Al no existir ningún tipo de infraestructura fija, los terminales usan el aire como canal de comunicación.
- Consumo de energía: Los nodos son móviles por lo tanto funcionan con baterías de vida limitada, por esa razón es muy importante que el consumo de energía se reduzca lo máximo posible.

2.6 Topologías de red

En su mayoría, las tecnologías previamente mencionadas pueden ser configuradas de distintas formas, en particular, lo que a su forma o cadena de comunicación con los demás equipos de la red se

refiere, se le conoce como la topología de red.

A la fecha se conocen varias topologías, entre las cuales vale la pena destacar para el presente estudio la red tipo malla [48] por ser un caso particular del estudio en redes Ad-Hoc. En ella podemos rescatar las siguientes características:

- Flexibilidad: Un nodo puede adherirse a la red si puede ver a uno de los nodos vecinos. Las zonas rurales aisladas normalmente no siguen una distribución geométrica ordenada alrededor de un punto central.

- Estructura descentralizada auto configurable: En las zonas rurales, donde gran parte del tiempo puede emplearse en desplazamientos, resulta fundamental que la tecnología minimice la instalación, administración y el mantenimiento de la red.

- Alimentación e integración: A diferencia de otras tecnologías inalámbricas donde sus nodos exigen grandes cantidades de energía, los nodos *Mesh* solo necesitan de una mínima energía, dotándoles de autonomía con una energía natural como pueda ser la solar. Además el hardware *Mesh* es fácilmente integrable en un sistema impermeable que soporte condiciones meteorológicas adversas.

2.7 Topología dinámica en las redes Ad-Hoc: El problema del enrutamiento

La constante movilidad de los terminales en una red Ad-Hoc supone un continuo cambio de la topología de la red e implica una nueva configuración de las tablas de enrutamiento de los nodos. Actualmente se investiga activamente en protocolos de enrutamiento para mejorar los resultados obtenidos hasta el momento.

En las redes MANETs el enrutamiento de paquetes entre cualquier par de nodos llega a convertirse en una tarea compleja, en contraste con las redes cableadas, las redes ad-hoc presentan cambios de topología frecuentes e impredecibles debido a la movilidad de sus terminales. Estas características impiden la utilización de protocolos de enrutamiento desarrollados para redes cableadas. Una característica especialmente importante de los protocolos de enrutamiento para redes ad hoc es que deben poder

adaptarse rápidamente a los cambios dinámicos en la topología de la red y a la baja confiabilidad del medio. Al mismo tiempo, tales protocolos son los responsables del descubrimiento, establecimiento y mantenimiento de las rutas de comunicación de la red [49].

2.7.1 Protocolos Proactivos

Los protocolos proactivos tratan de mantener la información necesaria para el encaminamiento continuamente actualizado mediante el uso de tablas. Cada nodo tiene una o varias tablas en las que guarda la ruta que debe utilizar para llegar a cualquier otro nodo de la red. Cuando la topología sufre una modificación (un nodo se incorpora, deja de formar parte o cambia de posición) se inunda la red de mensajes tipo *broadcast* para actualizar las rutas de todas las tablas [50].

2.7.2 Protocolos Reactivos

A diferencia de los protocolos proactivos, los reactivos buscan como llegar al nodo destino cuando quieren iniciar una comunicación. Estos van descubriendo la ruta para cada comunicación entre un nodo fuente y un nodo destino; esto es lo que les ha llevado a conocerse como protocolos bajo demanda. Estos protocolos optimizan los recursos evitando el envío de paquetes de forma innecesaria. Como contrapartida sufren una pérdida de tiempo cada vez que realizan el descubrimiento de la ruta[50].

2.7.3 Protocolos Híbridos

Como es común, se suelen implementar técnicas que permiten combinar elementos proactivos y reactivos en protocolos para dominios donde simultáneamente se gestionan comunicaciones intra e interdominio. Donde por lo general el enrutamiento intradominio es asumido por el rol proactivo y la función de enrutamiento en la comunicación interdominio se asume de manera reactiva[51]

Es claro sin embargo, que las clasificaciones aquí anunciadas pueden ser complementadas por propuestas que detallen otros tipos de comportamiento deseables en los protocolos de enrutamiento de las MANET, como el rol de cada nodo frente al mantenimiento de la información topológica de la red, gestión de la movilidad, la interdependencia informática entre nodos, el conocimiento de la posición, la organización de la red, entre otras [52].

2.7.4 Optimized Link State Routing (OLSR)

OLSR es un protocolo pensado para MANETs cuyo estándar es el RFC 3626 [53]. Este protocolo se basa en tablas de enrutamiento. Una de sus características más importantes es que es proactivo, lo que quiere decir que cada nodo dispone en cada momento de toda la información del estado y disposición de los nodos de la red. OLSR funciona bien en redes con alto número de usuarios (nodos) y con una topología cambiante. Para llevar un control, se intercambian periódicamente mensajes de tal forma que se va aprendiendo la topología de la red y el estado de los nodos vecinos. El protocolo reduce el tamaño de los paquetes de control, ya que en lugar de declarar todos los enlaces de un nodo con sus nodos vecinos, declara solamente un conjunto de estos denominados Multipoint Relay Selectors. También utiliza únicamente nodos seleccionados (Multipoint Relays, MPRs) para difundir los paquetes en la red [51].

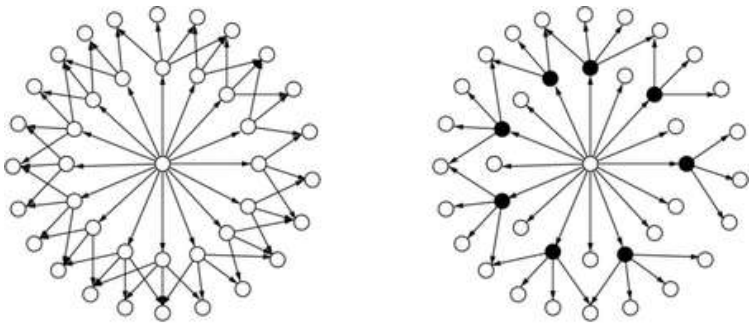


Figura 1. Inundación tradicional vs. Inundación por MPR OLSR [54]

OLSR soporta una movilidad nodal que puede ser seguida a través de sus mensajes de control locales, y que depende de la frecuencia de envío de estos [54].

- Mensajes OLSR

OLSR utiliza un único formato para todos los mensajes relacionados con este protocolo, de esta forma se facilita la extensión del protocolo sin problemas de compatibilidad con versiones anteriores. Los paquetes se transmiten por la red, encapsulados en paquetes UDP utilizando el puerto 698 asignado por IANA.

Los tres tipos de mensajes que hacen posible el funcionamiento del protocolo son los siguientes:

1. Mensajes HELLO: Realizan las funciones de descubrimiento de nodos, de detección de estado de enlaces y de señalización y elección de MPRs.

2. Mensajes TC (Topology Control): Realizan la función de declaración de la topología de la red. Cada nodo guarda información de la topología de la red para poder realizar cálculos de tablas de encaminamiento.

3. Mensajes MID: Realizan la función de descubrimiento de la presencia de múltiples interfaces en un nodo.

2.7.5 Better Approach to Mobile Ad-Hoc Networking (B.A.T.M.A.N.)

Derivado del protocolo de enrutamiento OSLR, B.A.T.M.A.N. surge en Berlín en el año 2006 como propuesta frente al mismo que busca superar algunas de sus dificultades. En este protocolo, solo se conocen las rutas hacia los vecinos más próximos (a un solo salto) en cada nodo y remplaza los mensajes OSLR por un único tipo de mensaje denominados Originator Messages (OGMs) [55].

La estrategia básica del protocolo reside en la comprobación del campo Originator Address, de tal manera que si el campo coincide con las cabeceras del IP mensaje, los dos nodos (receptor y transmisor del mensaje) son vecinos directos.

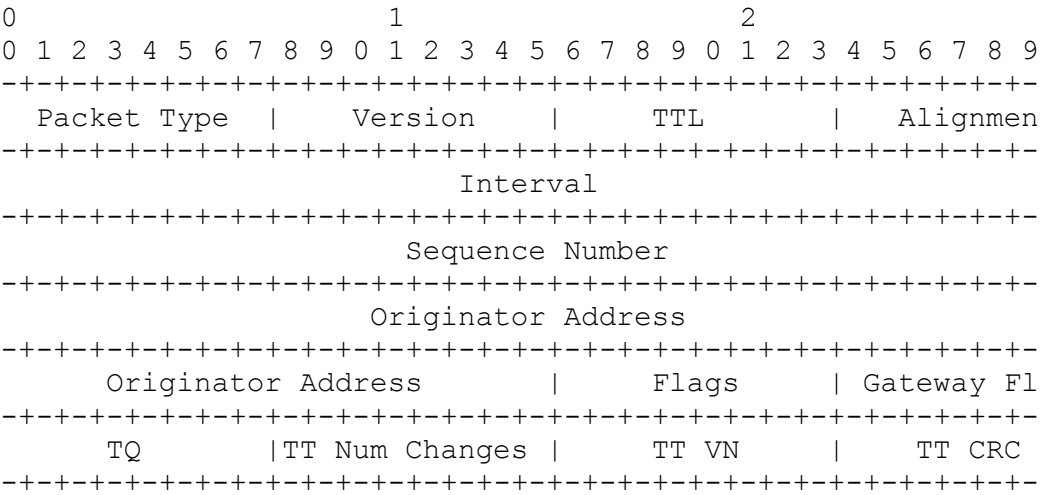


Figura 2. Estructura de un paquete OGM

De manera similar al protocolo OLSR, los mensajes OGM son difundidos a través de la red bajo el mecanismo de inundación hasta que el valor TTL se haga 0 o hasta que el paquete OMG haya sido recibido por el mismo nodo que lo envió, de esta manera,

todos los nodos pueden conocer la existencia de los demás nodos y las rutas hacia los nodos más cercanos (a un salto de sí mismos).

La última versión del protocolo, conocida como BATMAN-adv, introduce una de las mayores características con las que cuenta actualmente el protocolo. Esta es su operación en la capa de enlace (capa 2 del modelo OSI), de tal forma que la información de enrutamiento se inserta en las tramas respectivas a dicha capa (Ethernet, Wi-Fi, entre otras) mediante encapsulamiento hasta el nodo destino, emulando un switch virtual entre todos los nodos de la red. Esto hace que todos los nodos parezcan estar conectados con un enlace local de 1 salto desde el punto de vista de IP, ocultándoles la verdadera topología de la red así como de los cambios en la misma.

Las últimas revisiones del protocolo B.A.T.M.A.N ha introducido un nuevo paquete llamado ELP(Echo Location Protocol). Los mensajes ELP son paquetes que se envían con alta periodicidad para comprobar rápidamente el estado de los enlaces locales, de manera que no se propagan por toda la red. Los mensajes OGM siguen cumpliendo la misma tarea que tenían asignados en anteriores versiones. Con esta división de tareas entre ELP y OGM podemos reducir ampliamente el intervalo de envío de los OGM (reduciendo el ancho de banda usado y la capacidad de procesamiento), mejorando los tiempos de convergencia y simplificando notablemente el núcleo del protocolo de encaminamiento de B.A.T.M.A.N.

2.7.6 Métricas de enrutamiento

Las métricas son conjuntos de parámetros que representan las condiciones de operación de un sistema y las hacen comparables con otras, permitiendo la elección de la alternativa más conveniente. Para el caso del problema del enrutamiento, dicha alternativa se refiere a la mejor ruta para transmitir los datos de un nodo cualquiera a otro, es decir aquella ruta que mejor satisfaga las siguientes condiciones:

- Mantener acotado el retardo entre pares de nodos de la red.
- Ofrecer altas cadencias efectivas independientemente del retardo medio de tránsito
- Ofrecer el menor costo.

Las métricas más utilizadas se resumen en la Tabla 1.

Tabla 1 Taxonomía de las métricas de enrutamiento [56]

Métrica	**Objetivos de optimización**	**Método de cómputo de la métrica**	**Función de evaluación**
Basados en la topología de la red: 1. Numero de saltos	- Minimizar el retraso	-Uso de información disponible localmente	- Sumatoria
Basados en la intensidad de la señal: 1. Preemptive routing [57] 2. SSAR [58] 3. Link quality factor [59]	- Mayor tiempo de vida esperado de ruta	- Uso de información disponible localmente	- No definido, decisión del algoritmo
Sondeo Activo: 1. Per hop RTT [60] 2. Per hop PktPair [60] 3. ETX [61] 4. ETT [62] 5. MTM [63] 6. WCETT [62] 7. MCR [64] 8. Modified ETX [65] 9. ENT [65] 10. MIC[66]	- Minimizar el retraso - Maximizar la probabilidad de entrega de paquetes - Balanceo de carga	- Sondeo activo	-Sumatoria
Interferencia 1.Interference-aware[67]	- Minimizar el retraso - Maximizar la probabilidad de entrega de paquetes	- Uso de información disponible localmente	- Sumatoria
Movilidad 1. ABR [68] 2. Link affinity metric [69]	- Mayor tiempo de vida esperado de ruta	- Sondeo activo - Métricas sobre los paquetes de descubrimiento de rutas	- No definido, decisión del algoritmo
Energía 1. MTPR [70] 2. MBCR [71] 3. CMMBCR [72] 4. MREP [73] 5. PIM [74] 6. MMBCR [75]	- Minimizar el consumo energético	- Uso de información disponible localmente	- Sumatoria - Estadísticas (Min-máx.)
Normalización 1. AirTime [76]	- Minimizar el retraso	- Sondeo activo - Uso de información disponible localmente	- No definido, decisión del algoritmo

3. DESARROLLO DEL MODELO DE RED AD-HOC

3.1 Modelo Propuesto

Para materializar los conceptos hasta aquí estudiados, se propone el siguiente modelo de redes superpuestas. El presente trabajo se enfoca en el desarrollo de la red Ad-Hoc representada mediante enlaces de línea discontinua (--) dentro del diagrama, las cuales son de naturaleza estocástica, por lo que más adelante serán objeto de simulación.

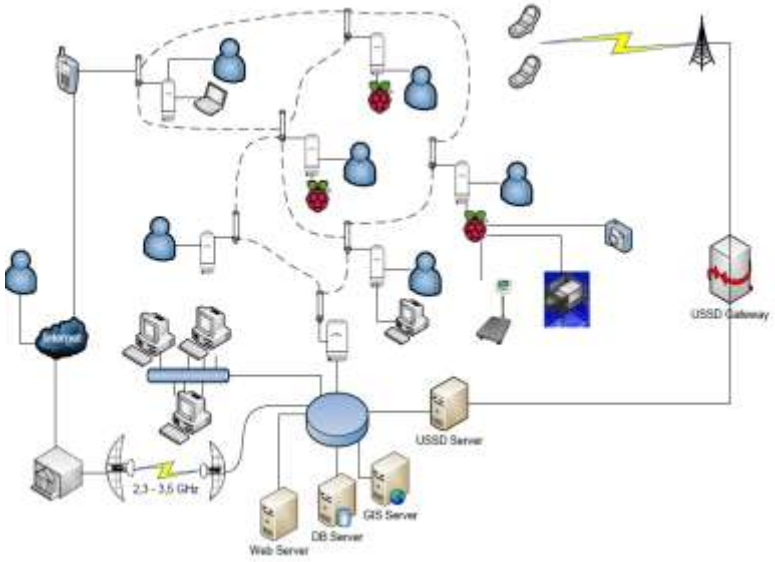


Figura 3. Diagrama general del modelo propuesto

3.2 Nodo Ad-Hoc: Concepto

Las condiciones dinámicas en una red Ad Hoc también hacen compleja la definición y comportamiento de un nodo dentro de una red de este estilo, por ello partiremos del modelo de un nodo Ad Hoc en un ambiente simulado como se observa en la Fig. 1. En ella se puede apreciar una adición de parámetros a la de un nodo regular o un host cualesquiera dentro de una red convencional.

Dentro de estos elementos podemos adicionar, fuera de los ya conocidos, parámetros de movilidad y consumo energético, los cuales limitan el desempeño de la red, procedimientos específicos para el manejo de colas, la resolución de direcciones, los modelos

de radio propagación y aún más complejo su dualidad de roles, tanto enrutador como cliente dentro de una red Ad Hoc.

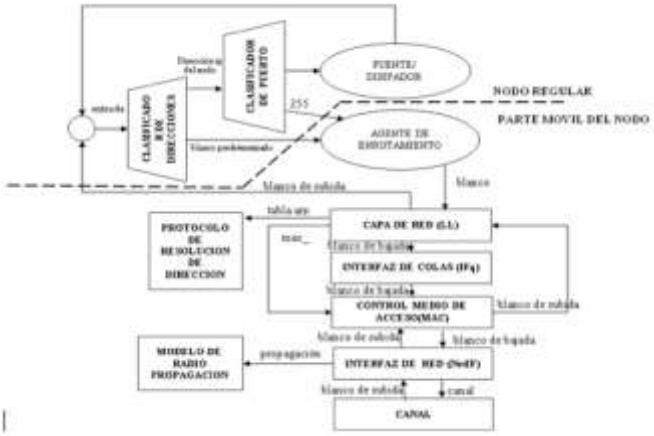


Figura 4. Nodo Móvil – Estructura simulación

Dentro del modelo simulado los nodos actúan como objetos que poseen parámetros definidos dentro de su arquitectura, estos parámetros generan primitivas que se comunican con las abstracciones de las capas de red propias del simulador de redes, en este caso Network Simulator 2 NS-2, (La abstracción se conserva en la versión posterior NS-3) dentro del objeto nodo se definen dos instancias principales las cuales son el puerto y el clasificador de direcciones, las instancias indican los atributos del nodo para convertirse en un ente Ad hoc, para ello el nodo asume tres roles importantes los cuales son Nodo Fuente, Nodo Destino o Nodo de Paso.

Dentro del modelo del nodo es vital definir las instancias de enrutamiento, en este caso los protocolos de enrutamiento Ad hoc como lo son el protocolo AODV (Ad Hoc On-Demand distance Vector) o el OLSR (Optimized Link State Routing), se convierten en atributos propios del nodo, al igual que la caracterización de sus recursos vital para definir el comportamiento del nodo y validarlo sobre un modelo de red Ad Hoc, adicionalmente el protocolo de enrutamiento permite la interacción entre las diferentes capas de Red y el manejo de los paquetes generados por los diferentes nodos en cualquiera de sus roles.

Finalmente la movilidad es un parámetro que hace vulnerable al nodo tanto en conectividad como en seguridad, sin contar las variaciones de los protocolos TCP/IP, ICMP o HTTP, dentro del medio de transmisión y aún más dinámico al evaluar los niveles de potencia de las redes en canales aledaños de transmisión e incluso

dentro de la misma frecuencia de transmisión, algo que hace aún más complejas las redes inalámbricas.

El modelo formal del nodo está ligado tanto a la cantidad de recursos disponibles como a su consumo energético y a su modelo de movilidad. En este sentido tenemos que el nodo basado en recursos lo podemos definir como:

$$\sum_{i=0}^{n} \delta m_n$$

Donde delta es un factor de ajuste del recurso el cual oscila entre 0 y 1 y m_n el recurso evaluado, para los efectos de este trabajo el recurso es el tráfico soportado por el nodo, lo cual redunda en la capacidad para recibir y proveer servicios dentro de la red. Al validarlo dentro del modelo de movilidad podemos tener al nodo como

$$\sum_{i=0}^{n} \alpha\delta m_n$$

En este caso alfa representa la distribución probabilística del movimiento del nodo, delta el factor de corrección la cual depende del escenario propio de configuración del nodo, la cual debe ser no determinística por la naturaleza misma del nodo Ad Hoc. El parámetro m únicamente representa un recurso, para tomar la totalidad del nodo se debe validar el escenario especifico a evaluar del nodo. Finalmente el nodo en este caso el dispositivo Raspberry Pi, debe responder al siguiente esquema dentro de la implementación propuesta, como se ve en la Figura 5.

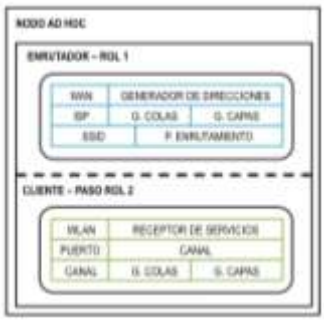


Figura 5. Nodo Móvil Implementación

4. SIMULACIÓN

Como parte del desarrollo del modelo de red Ad-Hoc para la adquisición de datos en la coordinación de redes logísticas, se ha implementado una simulación usando el software Network Simulator 3 (NS-3). Dicha simulación busca validar el comportamiento de la red dados tres escenarios, a saber:

1. Tres nodos semi-estáticos.
2. Cuatro nodos con movilidad aleatoria.
3. n-nodos y m-clústers con movilidad aleatoria.

Para estas simulaciones, se implementó el modelo de propagación de Friis, el cual basándose en el modelo de pérdida de señal en espacio libre (FSPL), propone que dado un par de antenas (transmisora y receptora), el cociente entre la potencia en la entrada de la antena receptora P_R y la potencia en la salida de la antena transmisora P_T, cumple con la siguiente relación:

$$\frac{P_R}{P_T} = G_t G_r \left(\frac{\lambda}{4\pi R} \right)^2 \qquad (1.1)$$

Donde: G_t y G_r son las ganancias de las antenas transmisora y receptora respectivamente, λ es la longitud de onda de la señal transmitida y R es la distancia entre las antenas. Igualmente para los dos últimos escenarios, fue implementado un modelo de movilidad aleatoria en cada uno de los nodos simulados de tal forma que lo que se busca verificar es la conectividad entre los nodos bajo condiciones de topología dinámica.

Los dos modelos implementados en la simulación (de propagación y movilidad) pretenden acercarse al escenario de operación real de la red, el cual en el caso de estudio resulta cercano dadas las condiciones de baja interferencia radioeléctrica en las zonas rurales del país y alta movilidad de los actores logísticos. En particular el último escenario permitió

comprobar el correcto funcionamiento de la red y la prestación de los servicios informáticos entre sus nodos de forma descentralizada.

El modelo de movilidad usado fue un modelo de movilidad sintético, basado en pausas siempre que se producen cambios de dirección o velocidad, este modelo es conocido como *Random Waypoint* de Johnson y Maltz que, entre sus características más resaltantes, cuenta con una amplia simplicidad y disponibilidad.

En los modelos de movilidad aleatoria los nodos puede ejercer desplazamientos de manera libre, estocástica e independiente entre los demás nodos de la red, la **Figura 6** muestra el desplazamiento de dos nodos distintos en un área rectangular de 2000m x 2000m.

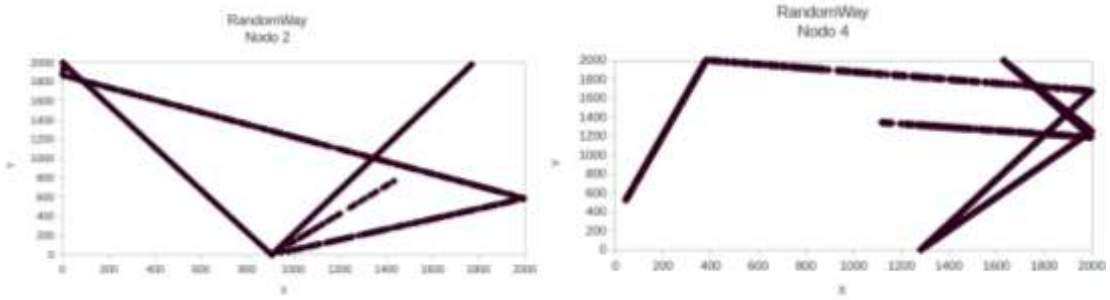


Figura 6. Movimiento del nodo 2 y 4

4.1 Primer Escenario

En este escenario, se dispuso de tres nodos que implementan el protocolo de enrutamiento OSLR, configurados geoespacialmente de tal forma que ninguno es capaz de comunicarse con su nodo más próximo pues sus interfaces inalámbricas se han configurado para tener un enlace efectivo de aproximadamente 250 metros.

El nodo ubicado más a la izquierda (nodo 3) (**Figura 7**), pretende enviar un mensaje al nodo ubicado en la posición (450,0,0) (nodo 2) pero como se mencionó, dicho mensaje no llega a su destino hasta que el nodo inicialmente más a la derecha (1000,0,0) (nodo 1) cambia su posición de manera espontanea en el segundo 20.0. Momento en el cual dicho nodo actúa como puente entre la comunicación de los nodos que buscan comunicarse. Seguido a ello, el nodo vuelve a su posición original en el segundo 35.0 y la comunicación desaparece.

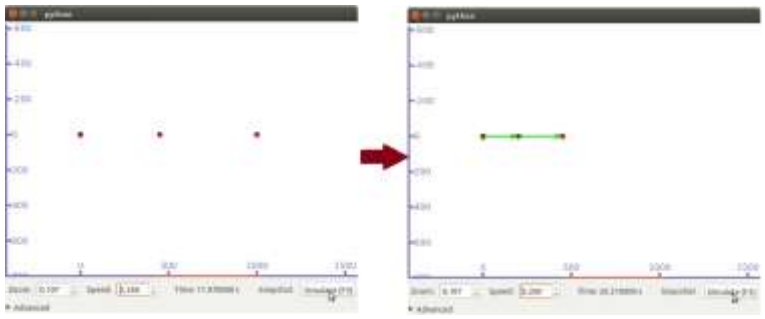


Figura 7. Visualización grafica del primer escenario de simulación

El objetivo primordial en esta simulación fue el de validar el comportamiento multi-salto en la transmisión de paquetes, lo cual se comprueba en la tabla de enrutamiento del nodo 3 (Figura 8) y por consiguiente comprobar el correcto funcionamiento del protocolo de enrutamiento instalado (OSLR), dicho comportamiento se evidencia en la Figura 9, donde se observa el intercambio de paquetes entre los nodos, en particular en el segundo 20.0 cuando el nodo 1 aparece en la posición necesaria para habilitar la comunicación.

Figura 8. Tabla de enrutamiento del nodo 3

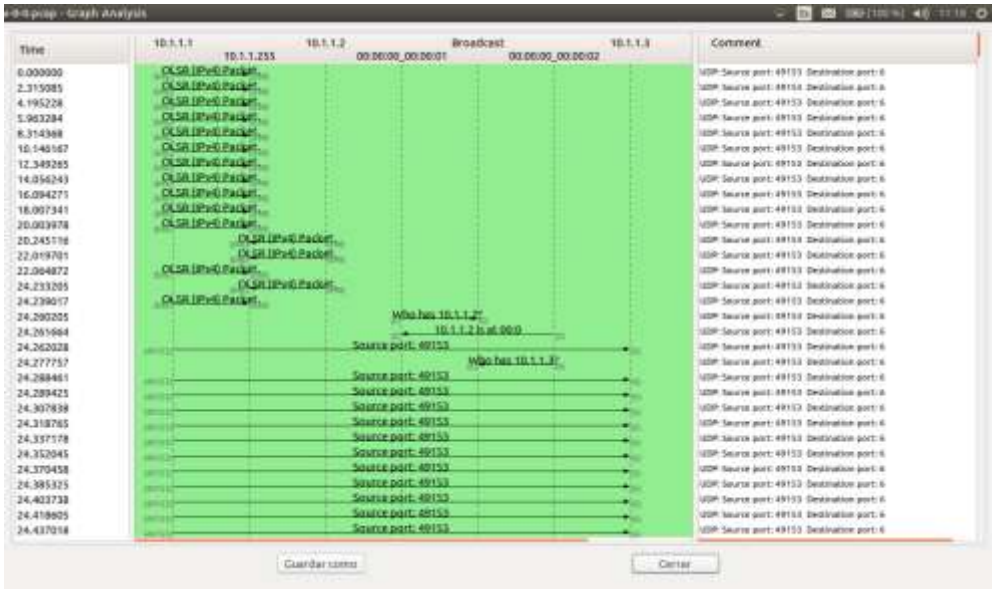


Figura 9. Intercambio de mensajes entre nodos en el segundo 20.0

Se ha hecho uso de este escenario de simulación para analizar brevemente la carga que supone la implementación del protocolo de enrutamiento para el trafico de la red, en este sentido, se analizaron las trazar generadas por el simulador en la herramienta Wireshark[77] de tal manera que se logró disgregar el trafico de la red según el protocolo asociado a cada uno de los paquetes (**Figura 10**).

Figura 10. Porcentajes de tráfico según protocolo de comunicación

Allí se observa la baja carga que supone para la comunicación el protocolo de enrutamiento (2.52% y 0,27% del total de paquetes y Bytes transmitidos respectivamente).

4.2 Segundo escenario

De forma similar al primer escenario de simulación, se instaló en cuatro nodos de red el protocolo de enrutamiento OSLR. Seguido a ello, se implemento el llamado a una función de generación de tráfico que sigue una distribución de Poisson entre los nodos de la red. Adicionalmente, se corrió la simulación bajo el modelo de propagación de Friis y un modelo de propagación aleatoria.

Cada uno de los nodos, se posiciona inicialmente en línea recta respecto a los demás e implementa un modelo de movilidad aleatorio delimitado por un espacio rectangular de tal forma que la disponibilidad de los enlaces de comunicación entre un par de nodos cualquiera de red se desarrolla de manera estocástica (**Figura 11**). Este comportamiento se evidencia en las tablas de enrutamiento de cada uno de los nodos.

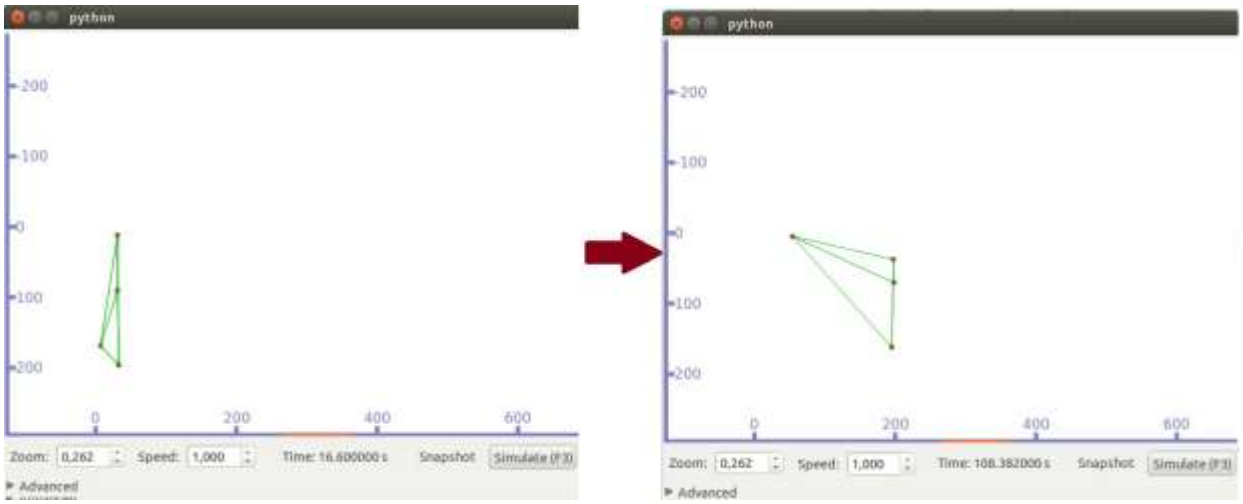


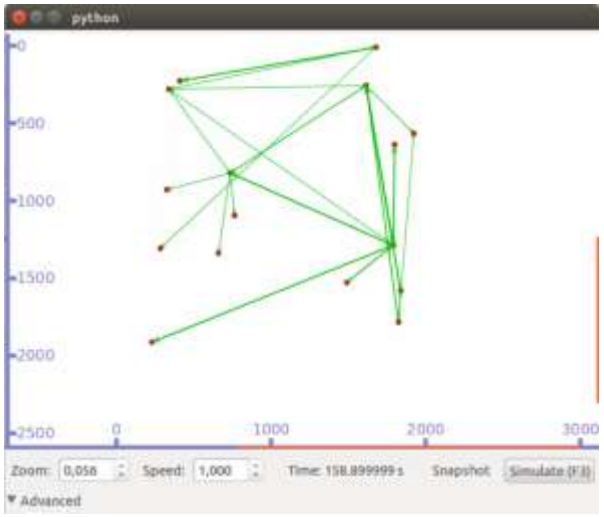
Figura 11. Visualización grafica del segundo escenario de simulación

Al variar las condiciones de movilidad de los nodos se obtienen distintos resultados en la estructura de las tablas de enrutamiento teniendo en cuenta que según la configuración de la potencia de transmisión en cada nodo permite la comunicación exitosa entre dos nodos en un rango de aproximadamente 250 metros. En las simulaciones realizadas, se evidenció que los nodos de la red pueden mantener comunicación con todos los demás siempre y cuando los límites del rectángulo se mantengan acotados.

4.3 Tercer escenario de simulación

Este último escenario resulta ser una generalización del anterior, donde se crean n nodos que pertenecen a m clústers de red y se alistan de igual manera que en el segundo escenario de simulación. Se realizaron pruebas con n=4 y m=5. Es decir cinco clústers de red cada uno con 4 nodos. La topología de la red para un instante dado se observa en la **Figura 12** así como la tabla de enrutamiento para un nodo elegido al azar.

La **Figura 13** muestra como está constituida la topología de red en este escenario, si bien los nodos independientes puedes moverse libremente, hacen parte de un grupo *"liderados"* por su *cluster head* y las comunicaciones con el resto de la red deben tener este nodo líder como salto obligatorio.



OLSR routing table for node 9

Destination	Next hop	Interface	Num. Hops
10.1.1.1	10.1.3.1	(interface 1)	2
10.1.1.2	10.1.3.1	(interface 1)	1
10.1.1.3	10.1.3.1	(interface 1)	2
10.1.1.4	10.1.3.1	(interface 1)	2
10.1.2.1	10.1.3.1	(interface 1)	2
10.1.2.2	10.1.3.1	(interface 1)	3
10.1.2.3	10.1.3.1	(interface 1)	3
10.1.2.4	10.1.3.1	(interface 1)	3
10.1.3.1	10.1.3.1	(interface 1)	1
10.1.3.2	10.1.3.2	(interface 1)	1
10.1.3.3	10.1.3.3	(interface 1)	1
10.1.4.1	10.1.3.1	(interface 1)	2
10.1.4.2	10.1.3.1	(interface 1)	3
10.1.4.3	10.1.3.1	(interface 1)	3
10.1.4.4	10.1.3.1	(interface 1)	3
10.1.5.1	10.1.3.1	(interface 1)	2
10.1.5.2	10.1.3.1	(interface 1)	3
10.1.5.3	10.1.3.1	(interface 1)	3
10.1.5.4	10.1.3.1	(interface 1)	3

Figura 12. Visualización grafica del tercer escenario de simulación

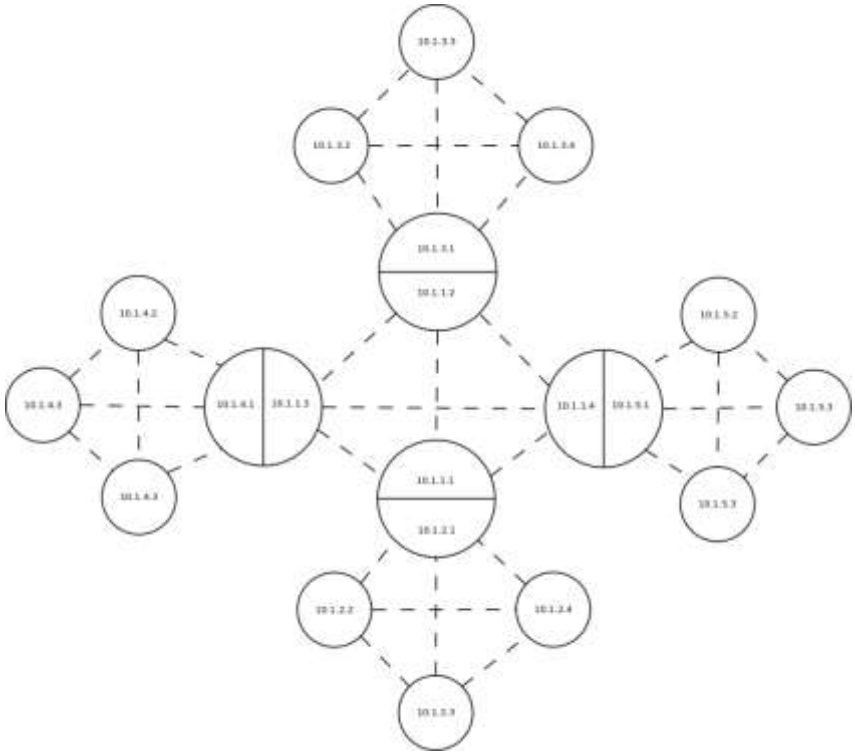


Figura 13. *Enlaces y topología estocástica de la red*

5. IMPLEMENTACIÓN DE LA RED AD-HOC

5.1 Raspberry Pi como Nodo Ad-Hoc

Se ha elegido como prototipo de implementación, al Ordenador de Placa Reducida (SBC por las siglas en ingles de la expresión Single Board Computer), Raspberry Pi modelo B+. Este es un dispositivo de bajo costo y en concreto, se trata de una placa base de 85 x 54 milímetros en la que se aloja un chip Broadcom BCM2835 con procesador ARM1176JZF-S que trabaja hasta a 1 GHz usando overclock, GPU VideoCore IV y 512 Mbytes de memoria RAM.

Figura 14. Raspberry Pi modelo B+

- Sistema Operativo en el Raspberry Pi

De fábrica el Raspberry Pi viene sin ningún tipo de sistema operativo preinstalado, su mecanismo de almacenamiento persistente resulta una tarjeta micro SD que debe adquirirse por separado. Para este SBC, se han desarrollado una serie de sistemas operativos, en su mayoría basados en el Kernel GNU-Linux. Principalmente por las características que se explicarán más adelante respecto a la integración del protocolo B.A.T.M.A.N. con el Kernel 2.6 de Linux, se ha elegido implementar el sistema operativo Raspbian en cada uno de los nodos de la red, el procedimiento se detalla a continuación.

1. Descargar la imagen del SO de la dirección
http://www.raspberrypi.org/downloads/

En el presente documento se supone la utilización del sistema operativo basado en Debian, Raspbian Wheezy.

2. Escribir la imagen en una memoria Micro SD de mínimo 2GB

   a. En Windows descargar y usar la herramienta Image Writer: http://sourceforge.net/projects/win32diskimager/

   b. En Linux o Mac usar el comando:

      sudo dd bs=4M if=<nombre de la imagen>.img of=/dev/xxx

3. Una vez escrita la memoria micro SD, el Sistema Operativo está listo para correr en el Raspberry Pi e iniciará automáticamente el proceso de configuración inicial en su primer inicio. Para ello, mediante el uso bien sea de una pantalla con soporte para HDMI, un mouse y teclado USB o una conexión serial a 115200 baudios mediante el uso de un cable convertidor USB-TTL (Ver Numeral 3.1), se procede a realizar la configuración inicial del SO, ésta en todo caso puede ser realizada en cualquier otro momento ejecutando el comando: sudo raspi-config

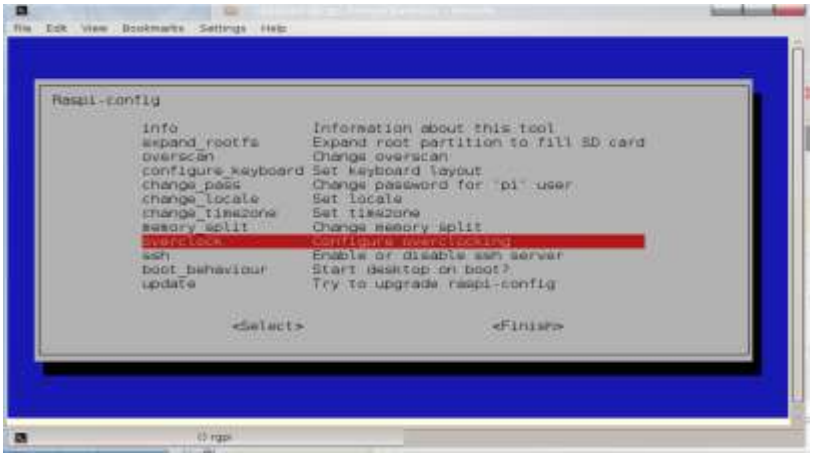


Figura 15. Interface de usuario del comando de raspi-config

Ejecutar la opción "expand_root fs" para permitir al sistema operativo usar la totalidad de la capacidad de la memoria micro SD. Habilitar el servidor ssh y configurar correctamente la zona horaria. Al terminar, seleccionar la opción "Finish" lo cual iniciará la ejecución del SO y según se haya elegido se iniciará un ambiente grafico o sencillamente la consola de Linux.

Según el sistema operativo utilizado en la maquina que se utilice para acceder al sistema embebido Raspberry Pi, el procedimiento

puede cambiar a propósito de la instalación de los controladores del convertidor usado (Se describe a continuación, el procedimiento para la utilización del cable USB-TTL con chip Prolific Pl 2303).

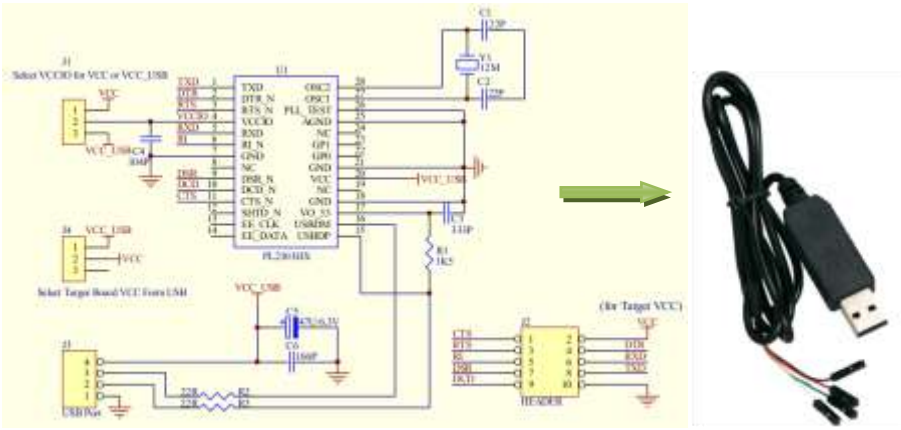


Figura 16. Circuito esquemático y Cable USB-TTL con chip Prolific PL-2303

a. Windows: En primer lugar se deben descargar e instalar los controladores del convertidor Prolific Pl 2303 del siguiente enlace:

   http://www.prolific.com.tw/UserFiles/files/PL2303_Prolific_DriverInstaller_v1_10_0.zip

   Seguido a esto, se necesitará una consola virtual para acceder al Raspberry Pi. Para ello se recomienda la utilización del software Putty:

   http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

   O hyperterminal:

   http://nksistemas.com/wp-content/uploads/2011/04/hyperterminal.rar

   En cualquiera de estos últimos, iniciar una comunicación serial a 115200 baudios con el puerto COMx asignado al cable USB-TTL luego de su instalación.

b. Linux: En el kernel de Linux los controladores vienen pre-instalados, así que luego de conectar el cable UBS-TTL, únicamente quedará pendiente verificar la existencia de una consola serial para probar la conexión. Para este fin, se utilizará el paquete screen que permite virtualizar una consola local o bien conectarnos a alguna consola externa indicando su ubicación. Para instalar el paquete basta correr el comando: sudo apt-get install screen.

Es claro entonces, que podremos conectarnos a la consola del Raspberry Pi usando este comando. La siguiente sintaxis se debe usar para establecer una conexión serial:

sudo screen <dirección> <velocidad de conexión en baudios>

Para encontrar la dirección donde ha sido conectado el cable USB-TTL, podemos ejecutar el comando cat /proc/tty/drivers el cual nos dará una lista de los puertos de conexión serial disponibles, dicha dirección es por defecto "/dev/ttyUSB0" y la velocidad de conexión pre configurada en Raspbian SO es de 115200 baudios, así pues el comando que se debe ejecutar para establecer la conexión debe ser similar al siguiente:

sudo screen /dev/ttyUSB0 115200

5.2 Linux Kernel 2.6 + B.AT.M.A.N-adv

Como una característica de alta importancia para la presente investigación, se resalta que desde 2007 y la versión 2.6 del *kernel* de Linux, el protocolo B.A.T.M.A.N-adv ha sido incorporado como modulo del espacio de k*ernel*, esto ha permitido mejorar su desempeño y garantizar de manera más simple, la implementación del estándar IEEE 802.11s [76] en dispositivos con sistema operativo basado en Linux. El sistema operativo implementado en los nodos es Raspbian Wheezy (2015-05-05) con versión de Linux *kernel* 3.18.

5.3 Raspberry Pi como nodo sensor

- Circuito sensor

El subsistema de control y comunicación, cumple la labor en primera instancia de sensar constantemente las variables de temperatura, humedad y peso, las dos primeras mediciones se realizan mediante sensores LM35 y HIH-4000-002. El presente subsistema se conectará en configuración serial con el subsistema de pesaje con el fin de acoplar sus respectivas mediciones en una tabla conjunta (peso, temperatura y humedad) que será transmitida de forma serial al computador. Tal objetivo será logrado mediante el uso del siguiente circuito electrónico donde los microcontroladores observados son de izquierda a derecha: MSP430G2553, MSP430G2553 y MSP430F2013. Los primeros 5 sensores LM35 representan para fines de la simulación y simplificación del esquema a los 5 sensores HIH-4000-002, los siguientes 5 sensores

representan los sensores de temperatura y el último de ellos representa la celda de carga.

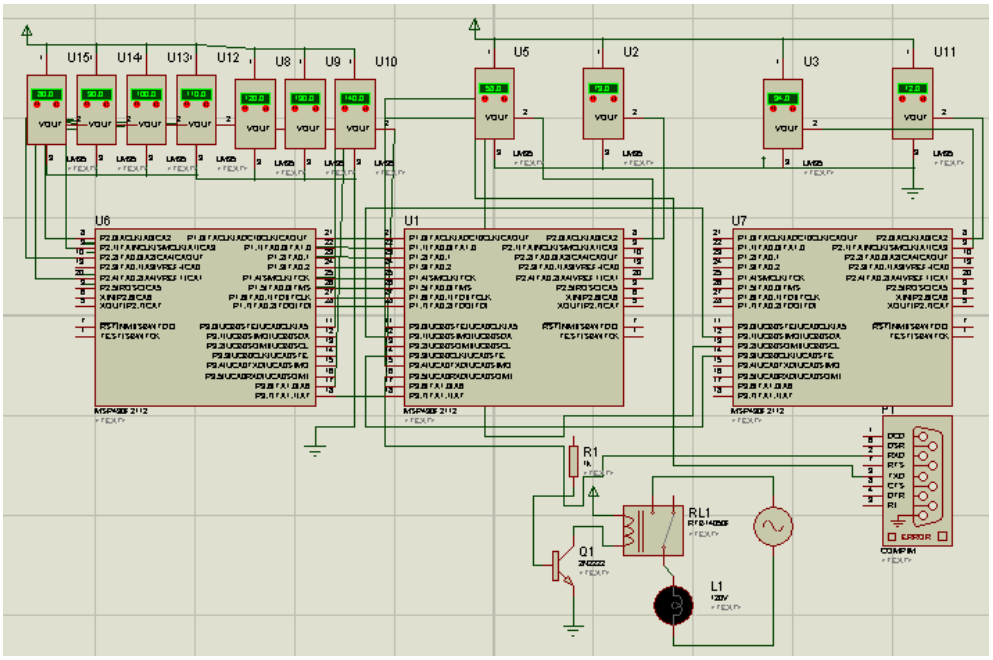


Figura 17. Circuito de Control y Comunicación Serial

Este circuito a su vez puede ser analizado en términos de sus funciones de la siguiente manera:

- Lectura de sensores

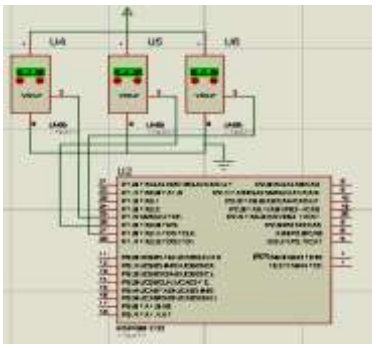


Figura 18. Esquema de adquisición de datos

La información recogida por los sensores de temperatura y humedad, debe ser convertida de su naturaleza analógica a su representación digital mediante el uso de un circuito de conversión análogo-digital (ADC), los cuales han sido programados por software dentro de los microcontroladores, las dos primeras variables gracias a su escala (0-100° y 0-100% respectivamente) y precisión requerida, pueden ser leídas con suficiente resolución por el circuito ADC de 10 bits, presente en el microcontrolador MSP430G2553 mientras que la variable correspondiente al peso para poder ser leída con una resolución aceptable será leída por el ADC de 16 bits presente en el MSP430F2013

- Sistema de comunicación con el Raspberry Pi

Se utilizó una conexión de tipo serial dúplex entre el Raspberry Pi y el micro controlador, a fin de recibir en él las variables recolectadas en los puntos anteriores y transmitir al circuito las variables de control necesarias como la temperatura de referencia y el momento de inicio del proceso, entre otras. La información generada por el micro controlador será recibida finalmente en un nodo Rapberry Pi, mediante una conexión de tipo serial por medio de su modulo UART. A continuación se presenta un análisis más detallado del desarrollo del subsistema realizando las consideraciones pertinentes para la implementación real del circuito anteriormente descrito. Los detalles de la implementación se pueden observar en el Anexo G.

5.4 Scripts de auto-configuración

Dada la naturaleza de una red Ad-hoc, sus nodos integrantes, deben contar con la capacidad de levantar su conexión con la red de manera autónoma y automática, para ello resulta de alto interés conocer las técnicas y pasos que sigue el sistema operativo del nodo para su inicialización (en nuestro caso de estudio el sistema operativo Raspbian). La siguiente es la lista de sucesos en el inicio de Linux:

1. El Basic Input/Output System (BIOS) o un gestor de arranque (como lilo, zlilo, grub, etc) carga el Kernel de Linux Kernel del disco a la memoria ram, con algunos parámetros de configuración definidos en el gestor de arranque. El Kernel reside en el directorio /boot y solo es accedido en este momento.
2. En la memoria, el código del Kernel inicia su ejecución detectando una serie de dispositivos vitales para el funcionamiento de la maquina como particiones de disco, etc.
3. Una de las últimas tareas del Kernel es la de montar el sistema operativo en la ruta /, la cual obligatoriamente debe contener los directorios /etc, /sbin, /bin y /lib.
4. Inmediatamente después, se invoca al comando init (/sbin/init) y se le delega el control del sistema al mismo.
5. El comando init leerá su archivo de configuración (/etc/inittab) el cual define el runlevel del sistema, y algunos scripts que deben ser ejecutados.

6. Estos scripts continuarán con la configuración de la infraestructura mínima del sistema, montando otros sistemas de archivos (según el contenido de /etc/fstab), activando el espacio swap (memoria virtual), etc.
7. El último paso, es la ejecución del script especial /etc/rc.d/rc, el cual inicializa los subsistemas de acuerdo a la estructura de directorios definida en /etc/rc.d. El nombre rc viene de la expresión en inglés "RUN COMMANDS".

- Runlevels

El mecanismo de runlevels, le permite a Linux inicializarse de distintas maneras. Y le permite al usuario cambiar de un perfil (runlevel) a otro, sin reiniciar el sistema. El runlevel por omisión se define en el archivo /etc/inittab con una linea como la siguiente:

```
id:2:initdefault:
```

Los runlevels se definen mediante números del 1 al 6 según las siguientes características:

- 0: Detiene el sistema. Todos los subsistemas son desactivados (por software) antes del apagado. Este nivel no debe ser usado como valor en la línea initdefault del fichero /etc/inittab.

- 1: Modo Mono-usuario. Solo subsistemas vitales son inicializados. No requiere autenticación (login) de usuario. Una terminal de línea de comandos es devuelta al usuario.

- 3, 2: 3 es usado cuando el sistema está en producción. Se puede entender como el runlevel en el que las aplicaciones de usuario correrán. 2 es similar a 3, pero sin NFS.

- 4: No se usa. Puede ser definido como se desee, pero es poco usual.

- 5: Similar a 3, con una interfaz grafica para inicio de sesión (login).

- 6: Similar a 0, pero luego del completo detenimiento del sistema, la maquina es reiniciada. Este nivel tampoco debe ser usado como valor en la línea initdefault del fichero /etc/inittab.

- Subsistemas

Linux provee una forma modular para organizar la inicialización de subsistemas. Un hecho importante al respecto es pensar en la interdependencia de los subsistemas. Por ejemplo, no tiene sentido iniciar un servidor web antes de iniciar el subsistema de navegación en la red. Los subsistemas están organizados en los directorios /etc/init.d y /etc/rcN.d (donde N represente el numero de runlevel).

- /etc/init.d
  Todos los subsistemas instalados incluyen en este directorio un programa de control a manera de script que sigue un estándar simple (el cual se describe mas adelante). La siguiente es una lista de ejemplo de los subsistemas instalados en Raspbian:

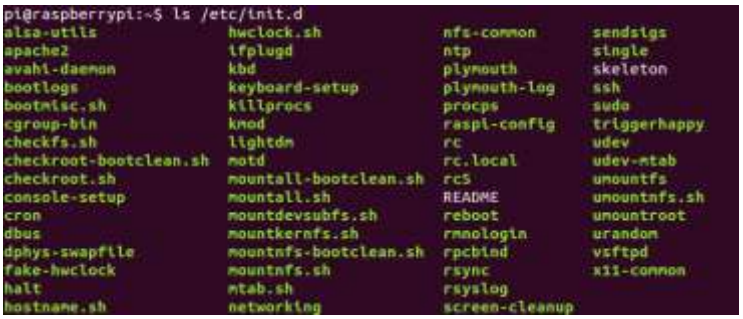


Figura 19. Lista de servicios registrados en /etc/init.d

- /etc/rc.d/rcN.d
  Estos directorios deben contener únicamente vínculos simbólicos a los scripts alojados en /etc/init.d. El siguiente es un ejemplo de ello en Raspbian:

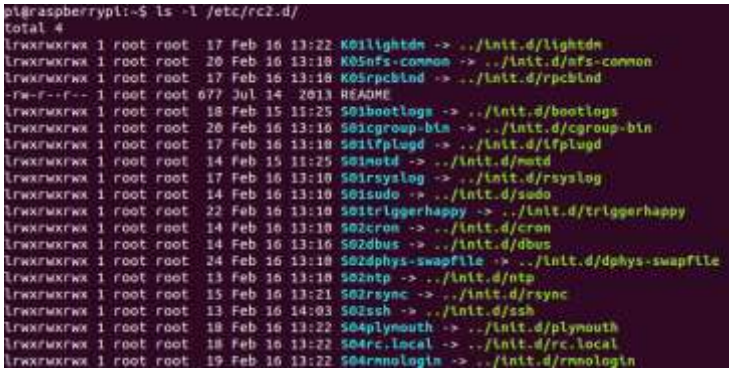


Figura 20. Lista de servicios para el runlevel 2

Como se observa, todos los nombres de los vínculos tienen un prefijo que empieza bien sea con la letra K (de Kill, para desactivar) o S (de Start, para activar), y un numero de 2 dígitos que define la prioridad en el arranque. En la lista mostrada se tiene por ejemplo que bootlogs (prioridad 01) inicia antes que otros subsistemas tales como ssh (prioridad 02) y el subsistema nfs desactivado (prefijo K) para este runlevel (2).

- Script Bootstrap.sh

Así que si se desea ejecutar un programa (para nuestro caso el programa que gestionará el comportamiento del nodo en la red Ad-Hoc) en el proceso de arranque del sistema, éste debe ser un subsistema. Para ello definiremos un subsistema con identificador "Ad-hoc" con el fin de iniciar la ejecución del servicio de auto-levantamiento de la red (Bootstrap.sh), el cual se alojará en /etc/init.d según el esqueleto provisto por el sistema operativo en /etc/init.d/skeleton (Anexo A).

El script "bootstrap.sh" será el encargado de iniciar los diferentes servicios requeridos para la conexión a la red Ad-hoc, entre los cuales cabe resaltar los siguientes:

1. Conectarse a la red y actuar como nodo Ad-Hoc dentro de ella.
2. Definición de puentes entre interfaces y pasarela a internet.
3. Encontrar otros nodos de la red.
4. Compartir la llave secreta para acceder a la red de manera segura.

El presente trabajo, se centrara en el numeral 1, esto sin olvidar que es necesario y de la mayor relevancia abordar los demás numerales en futuros trabajos. Para ello se asume una red Ad-hoc sin seguridad y donde se conocen plenamente las direcciones de los demás nodos de la red, los pasos detallados para este numeral serán los siguientes:

1. Inicialización de protocolo de enrutamiento para una red sin infraestructura (MESH o Ad-Hoc) bajo capa 2 (B.A.T.M.A.N).
2. Configuración de interfaz inalámbrica para conexión a red sin infraestructura.

3. Delegación del control de paquetes enviados por la interfaz inalámbrica a B.A.T.M.A.N.

4. Ejecución de técnicas de auto-direccionamiento (Zero-conf).

Así pues la primera versión del script de "auto-levantamiento" bootstrap.sh se incluye en el Anexo B. Finalmente, para incluir estos archivos de manera automática en el sistema operativo, se ha elaborado también un siguiente script de instalación del nodo (Anexo C).

Realizado lo anterior, al reiniciar el dispositivo, el dispositivo estará automáticamente conectado a la red Ad-Hoc con nombre "TLON_Adhoc"!!. Los siguientes son los resultados de ejecutar los comandos ifconfig e iwconfig en dos dispositivos pertenecientes a la red "TLON_Adhoc":

1. **Comando: iwconfig wlan0**

RPi 1:

```
wlan0     IEEE 802.11bgn  ESSID:"TLON-Adhoc"
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: 02:1B:55:AD:0C:02
          Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

RPi 2:

```
wlan0     IEEE 802.11bgn  ESSID:"TLON-Adhoc"
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: 02:1B:55:AD:0C:02
          Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

2. **Comando: ifconfig bat0:avahi**

RPi 1:

```
bat0:avahi Link encap:Ethernet  HWaddr be:db:cb:61:dc:c7
          inet addr:169.254.11.232  Bcast:169.254.255.255
Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

RPi 2:

```
bat0:avahi Link encap:Ethernet  HWaddr 1a:43:2a:30:f1:e6
          inet addr:169.254.10.106  Bcast:169.254.255.255
Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

En los resultados, se puede observar que los dos dispositivos comparten la configuración de red haciendo posible su comunicación y ejecutan el protocolo de enrutamiento B.A.T.M.A.N. para la gestión del enrutamiento de paquetes de información dentro de la red. Así mismo, han resuelto de manera autónoma su configuración de direccionamiento IPV4 mediante el uso del sistema Avahi [78] haciéndose así parte de la subred definida por la dirección de multidifusión 169.254.255.255/16 la cual permite un total de 65.534 nodos de red en el clúster generado.

Finalmente, se implementó el servicio de nombre "port_publisher" (Anexo D) en el nodo, el cual se encarga de llamar en su ejecución a un script desarrollado en Python como parte de los ejemplos de uso de las librerías PySerial. Su código puede ser consultado en el Anexo E. Este último se encarga de crear un socket de conexión direccionado al modulo UART del nodo Raspberry Pi y publicar mediante el uso de la librería Python-Avahi dicho servicio en la red, con el uso de mensajes *Broadcast*.

6. PRUEBAS DE DESEMPEÑO

Una vez establecido el clúster de red con los procedimientos detallados en los capítulos anteriores, se procedió a tomar una serie de medidas para evaluar el desempeño de la red. Para ello, se determinaron las siguientes medidas:

- Ancho de banda
- Distancia máxima de comunicación
- Razón de paquetes enviados sobre paquetes recibidos
- Tiempo de demora de trasmisión de paquetes

Se realizaron también pruebas de enrutamiento múlti-salto de la red para comprobar el buen funcionamiento del protocolo de comunicaciones, así como para validar los resultados de simulación. La gráfica que caracterizó el ambiente en términos de la ocupación del espacio radioeléctrico se observa en la Figura 21.

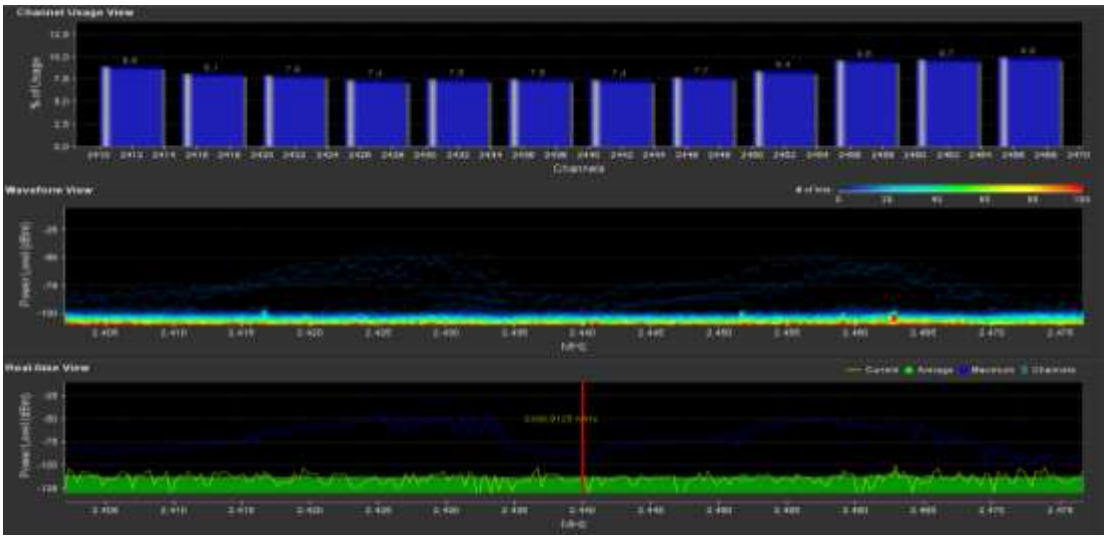


Figura 21. Ocupación del espacio radioeléctrico entre las bandas de 2,41 y 2,47 MHz durante las mediciones en campo

A continuación se detallan los resultados de cada una de las mediciones.

6.1 Ancho de banda

Para la medición del ancho de banda, se usó la herramienta Iperf, la cual permite medir el máximo desempeño TCP y UDP del ancho de banda de una conexión [79]. Las mediciones con esta herramienta consistieron en el envió de paquetes TCP de 43.8 Kb bajo cinco regímenes de transmisión durante 15 segundos:

1. Transmisión unidireccional de un solo hilo
2. Transmisión bidireccional secuencial(semi-dúplex) de un solo hilo
3. Transmisión bidireccional simultanea(full-duplex) de un solo hilo
4. Transmisión unidireccional de cinco hilos paralelos
5. Transmisión bidireccional secuencial(semi-dúplex) de cinco hilos paralelos

La Figura 22 muestra de forma gráfica los resultados de medición del ancho de banda bajo los regímenes 1 y 4 entre dos nodos de la red Ad-Hoc. Los resultados agregados, se observan en la Tabla 2.

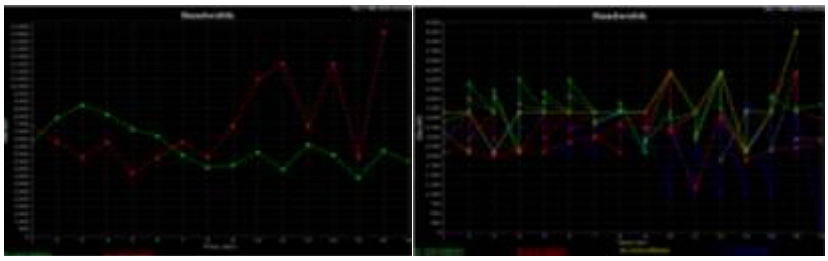


Figura 22. Ancho de banda en transmisión bidireccional dúplex de un hilo y bidireccional semi-dúplex de cinco hilos

Tabla 2. Resultados agregados de medición de ancho de banda entre dos nodos A y B

Régimen	Ancho de banda medido (Kbits/seg)		Promedio (Σ en Rg. 2)	Total
	A->B	A<-B		
1	14259		xx	14259
2	7490	6000	13490	13490
3	14022	14407	14214,5	14214,5
4	14407		xx	14407
5	13590	13985	13787,5	13787,5
Promedio total				**14031,6**

De estos resultados podemos promediar un ancho de banda de aproximadamente **14 Mbits/segundo** para la conexión entre dos nodos de la red, lo cual representa una muy buena medida para las aplicaciones propuestas en la red. Los resultados detallados de las mediciones con la herramienta Iperf se pueden encontrar en el Anexo G.

6.2 Distancia máxima de comunicación.

Figura 23. Medición de máxima distancia de transmisión

Para su medición se recreó la red con únicamente dos nodos para garantizar que no se presentarán errores en la medición a causa de un enlace múlti-salto, el experimento consistió en generar una transmisión tipo *ping sostenido* desde un nodo a otro y medir la distancia en la cual se reportó la perdida de la ruta hacia el nodo destino (**Figura 23**). Se realizaron 10 repeticiones de forma radial en torno al nodo origen, los siguientes fueron los resultados obtenidos:

Tabla 3. Mediciones de distancia máxima de conexión entre dos nodos

Repetición	Distancia máxima de conexión (m)
1	192
2	187
3	201
4	181
5	179
6	215
7	196
8	187
9	193
10	182
Promedio	**191,3**

6.3 Razón de paquetes enviados sobre paquetes recibidos y tiempo de demora de trasmisión de paquetes

Para evaluar estas dos medidas, se realizó un experimento en el que se transmitió una cantidad fija de 200 paquetes de 64 Kb entre dos nodos elegidos al azar. Dicha medición se repitió a lo largo de dos caminos: uno con obstáculos y otro a campo abierto hasta perder la comunicación entre los nodos. Los resultados se sintetizan en las siguientes gráficas según cada uno de los caminos mencionados (campo abierto y con obstáculos).

- **Escenario de campo abierto**

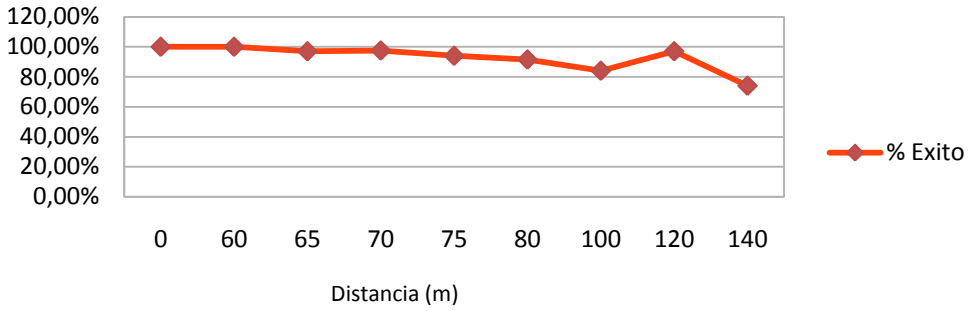


Figura 24. Razón de paquetes enviados sobre paquetes recibidos en campo abierto

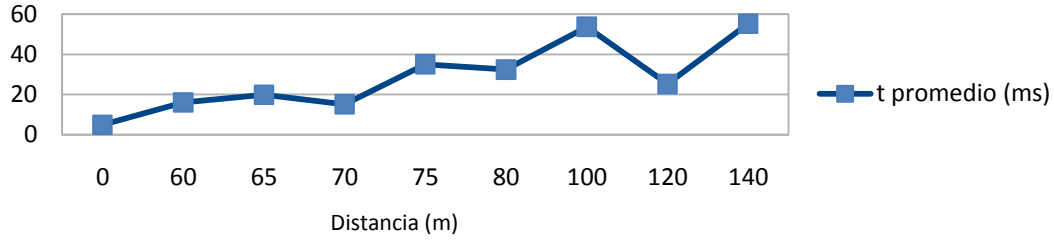


Figura 25. Latencia promedio en la transmisión de un paquete en campo abierto

- **Escenario con obstáculos**

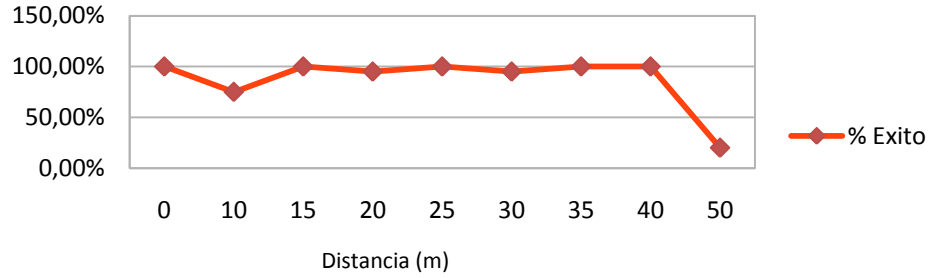


Figura 26. Razón de paquetes enviados sobre paquetes recibidos en medio de obstáculos

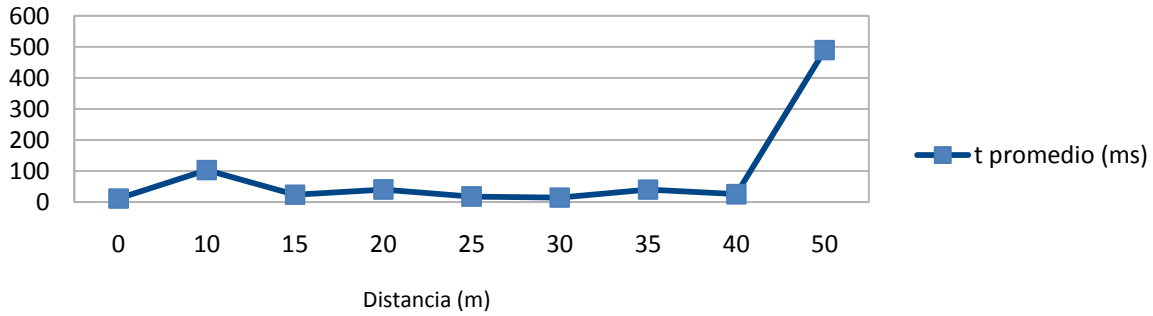


Figura 27. Latencia promedio en la transmisión de un paquete en medio de obstáculo

7. CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Los recientes avances en las ciencias de la computación así como en nuevas metodologías analíticas para la coordinación de la actividad logística tanto en cadenas de suministro como productivas del sector agropecuario, han despertado el interés por modelar e implementar sistemas integrados de operación lógica y física, valiéndose de la informática y las telecomunicaciones como medio para la adquisición y gestión de los servicios requeridos en tales procesos. El desarrollo de redes inalámbricas (WSN en particular) que soporten dicha operación, es de gran interés por sus altas prestaciones y bajos costos de implementación, más requieren de la posibilidad para gestionar la información a través de sistemas de red heterogéneos y con altos niveles de aleatoriedad en su operación y mantenimiento de los caminos de comunicación y enrutamiento de paquetes informáticos.

Mediante el presente trabajo se logró establecer una base conceptual y técnica para la implementación de sistemas expertos enfocados en la gestión descentralizada de la información en contextos ausentes de una sólida infraestructura de red. Dicha base tecnológica deberá contar además, con la capacidad de incluir técnicas de gestión de los flujos informáticos que inundan las cadenas logísticas y permitan la trazabilidad sobre los datos relacionados con los procesos de toma de decisiones en su interior. Se implementó para ello un ejercicio de sensorica como servicio (S2aaS) a través de un clúster de red Ad-hoc observándose niveles de calidad del servicio suficientes para su correcta prestación.

7.2 Recomendaciones

Se evidencia la necesidad de incluir en la capa de aplicación de la red propuesta, mecanismos de coordinación para la gestión de los recursos y servicios prestados en la misma, de tal manera que el modelamiento y simulación del comportamiento estocástico de la red resulten siendo la base para la una prestación efectiva y de alta calidad de tales servicios a los distintos agentes que componen la red.

A. Anexo: Cronograma

Tabla 4 Cronograma de actividades

#	Actividades\Semanas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1.	Identificación del problema y alcance																
2.	Estado del arte y marco teórico																
3.	Desarrollo de etapa de comunicación sensor-raspberry																
4.	Implementación de la red Ah-hoc																
5.	Modelamiento y simulación																
6.	Desarrollo de productos académicos																
7.	Pruebas de desempeño “en campo”																
8.	Análisis de Resultados y Conclusiones																
9.	Socialización																

B. Anexo: Archivo /etc/init.d/skeleton

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          skeleton
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO

# Author: Foo Bar <foobar@baz.org>
#
# Please remove the "Author" lines above and replace them
# with your own name if you copy and modify this script.

# Do NOT "set -e"

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Description of the service"
NAME=daemonexecutablename
DAEMON=/usr/sbin/$NAME
DAEMON_ARGS="--options args"
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is present
# and status_of_proc is working.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
```

```
        # Return
        #   0 if daemon has been started
        #   1 if daemon was already running
        #   2 if daemon could not be started
        start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --
test > /dev/null \
                || return 1
        start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON -- \
                $DAEMON_ARGS \
                || return 2
        # Add code here, if necessary, that waits for the process to be ready
        # to handle requests from services started subsequently which depend
        # on this one.  As a last resort, sleep for some time.
}

#
# Function that stops the daemon/service
#
do_stop()
{
        # Return
        #   0 if daemon has been stopped
        #   1 if daemon was already stopped
        #   2 if daemon could not be stopped
        #   other if a failure occurred
        start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile
$PIDFILE --name $NAME
        RETVAL="$?"
        [ "$RETVAL" = 2 ] && return 2
        # Wait for children to finish too if this is a daemon that forks
        # and if the daemon is only ever run from this initscript.
        # If the above conditions are not satisfied then add some other code
        # that waits for the process to drop all resources that could be
        # needed by services started subsequently.  A last resort is to
        # sleep for some time.
        start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec
$DAEMON
        [ "$?" = 2 ] && return 2
        # Many daemons don't delete their pidfiles when they exit.
        rm -f $PIDFILE
        return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
        #
        # If the daemon can reload its configuration without
        # restarting (for example, when it is sent a SIGHUP),
        # then implement that here.
        #
        start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name
$NAME
        return 0
}

case "$1" in
  start)
        [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
        do_start
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
```

```
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  stop)
        [ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
        do_stop
        case "$?" in
                0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
                2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
        esac
        ;;
  status)
        status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
        ;;
  #reload|force-reload)
        #
        # If do_reload() is not implemented then leave this commented out
        # and leave 'force-reload' as an alias for 'restart'.
        #
        #log_daemon_msg "Reloading $DESC" "$NAME"
        #do_reload
        #log_end_msg $?
        #;;
  restart|force-reload)
        #
        # If the "reload" option is implemented then remove the
        # 'force-reload' alias
        #
        log_daemon_msg "Restarting $DESC" "$NAME"
        do_stop
        case "$?" in
          0|1)
                do_start
                case "$?" in
                        0) log_end_msg 0 ;;
                        1) log_end_msg 1 ;; # Old process is still running
                        *) log_end_msg 1 ;; # Failed to start
                esac
                ;;
          *)
                # Failed to stop
                log_end_msg 1
                ;;
        esac
        ;;
  *)
        #echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
        echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
        exit 3
        ;;
Esac
```

C.Anexo: Bootstrap.sh

```
modprobe batman-adv || (echo "Error al cargar B.A.T.M.A.N";exit 11)
e=$?;[ $e -ne 0 ] && echo "Error: ${e}" >> /opt/adhoc/logs/error.log &&
exit $e
INT=$(iwconfig 2> /dev/null | grep -v "^ " | awk '{print $1}' | head -1
| egrep "^[a-z]")
if [ $? -eq 0 ]
then
echo "Apagando interfaz ${INT}"
ip link set ${INT} down
fi
sleep 1
ifconfig ${INT} down; iwconfig ${INT} mode ad-hoc || (echo "Falló al
iniciar modo Ad-hoc";exit 22)
e=$?;[ $e -ne 0 ] && echo "Error: ${e}" >> /opt/adhoc/logs/error.log &&
exit $e
ifconfig ${INT} mtu 1532
iwconfig ${INT} mode ad-hoc essid TLON-Adhoc ap 02:1B:55:AD:0C:02
channel 1 || (echo "Error al entrar a red Ad-hoc";exit 23)
e=$?;[ $e -ne 0 ] && echo "Error: ${e}" >> /opt/adhoc/logs/error.log &&
exit $e
sleep 1
ip link set ${INT} up || (echo "Error al levantar interfaz";exit 24)
e=$?;[ $e -ne 0 ] && echo "Error: ${e}" >> /opt/adhoc/logs/error.log &&
exit $e
sleep 1
batctl if add ${INT}
echo "Levantando interfaz B.A.T.M.A.N"
ifconfig bat0 up
echo "Ejecutando Zero-Conf"
avahi-autoipd -D bat0
sleep 5
ifconfig bat0:avahi && echo "Verifique su conexión en la red Ad-hoc!"
```

D. Anexo: Script de Instalación

```
#!/bin/sh

modprobe batman-adv
[ $? -ne 0 ] && echo "FATAL: B.A.T.M.A.N no disponible en el Kernel" &&
exit 1
apt-get -y install batctl bridge-utils wireless-tools avahi-autoipd
python avahi-utils
[ $? -ne 0 ] && echo "FATAL: Falló al instalar paquetes requeridos" &&
exit 1
apt-get -y install talk iperf tightvncserver gcc screen apache2 vsftpd
[ $? -ne 0 ] && echo "WARNING: Falló al instalar paquetes opcionales"

echo "...Copiando script de inicio en /etc/init.d"
cp scripts/init.d_TLON_Adhoc /etc/init.d/TLON_Adhoc
[ $? -ne 0 ] && echo "Falló copia del script de inicio" && exit 0

mkdir -p /opt/adhoc/scripts
mkdir -p /opt/adhoc/logs

echo "...Copiando script de auto-configuración en /opt/adhoc/scripts/"
cp scripts/bootstrap.sh /opt/adhoc/scripts/bootstrap.sh
chmod +x /opt/adhoc/scripts/bootstrap.sh
[ $? -ne 0 ] && echo "Falló copia del script de auto-configuración" &&
exit 0

echo "Inicio del servicio Ad-hoc"
chmod +x /etc/init.d/TLON_Adhoc
update-rc.d TLON_Adhoc defaults
[ $? -ne 0 ] && echo "Falló levantando el servicio" && exit 0


flag=0;
while [ $flag -eq 0 ]
do
echo -n "Instalar librerias pySerial y pyServer (WARNING: Deshablita
ttyAMA0 como shell!)?(s/n):"; read ext
case $ext in
  's')
    cp -r scripts/PyServer /opt/adhoc/scripts/
    if [ ! -f /boot/cmdline_back ]; then
                        cp /boot/cmdline.txt /boot/cmdline_back
```

```
                        [ $? -ne 0 ] && echo "Falló copia de seguridad de
/boot/cmdline.txt" && exit 0
                        cp scripts/Serial/cmdline /boot/cmdline.txt
                        cp /etc/inittab  /etc/inittab_back
                        [ $? -ne 0 ] && echo "Falló copia de seguridad de
/etc/inittab" && exit 0
                        cp scripts/Serial/inittab  /etc/inittab
      fi
      mkdir -p /opt/adhoc/pyserial-2.6
      tar -zxvf scripts/Serial/pyserial-2.6.tar.gz -C
/opt/adhoc/pyserial-2.6
      cd /opt/adhoc/pyserial-2.6/
      python setup.py install
      [ $? -ne 0 ] && echo "WARNING: Falló instalación pySerial"

      apt-get -y install python-avahi
      [ $? -ne 0 ] && echo "FATAL: Falló al instalar python-avahi" &&
exit 1
      cd /opt/adhoc/
      cp scripts/PyServer/port_publisher.py
/usr/local/bin/port_publisher.py
      cp scripts/PyServer/port_publisher.sh /opt/adhoc/scripts/PyServer/
      cd scripts/PyServer/
      sh port_publisher.sh start
      [ $? -ne 0 ] && echo "WARNING: Falló la publicación del servicio"

      flag=1;
     ;;
  n)
      echo "No se instaló pySerial"
      flag=1;
     ;;
   *)
      echo "Opción invalida";
      flag=0;
     ;;
esac
done

echo "Instalación exitosa."
```

E.Anexo: Port_Publisher.sh

```
#! /bin/sh
# daemon starter script
# based on skeleton from Debian GNU/Linux
# cliechti at gmx.net

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/local/bin/port_publisher.py
NAME=port_publisher
DESC="serial port avahi device publisher"

test -f $DAEMON || exit 0
set -e

case "$1" in
    start)
        echo -n "Starting $DESC: "
        python $DAEMON --daemon --pidfile /var/run/$NAME.pid
        echo "$NAME."
        ;;
    stop)
        echo -n "Stopping $DESC: "
        start-stop-daemon --stop --quiet --pidfile /var/run/$NAME.pid
        # \     --exec $DAEMON
        echo "$NAME."
        ;;
    restart|force-reload)
        echo -n "Restarting $DESC: "
        start-stop-daemon --stop --quiet --pidfile \
                /var/run/$NAME.pid
                # --exec $DAEMON
        sleep 1
        $DAEMON --daemon --pidfile /var/run/$NAME.pid
        echo "$NAME."
        ;;
    *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop|restart|force-reload}" >&2
        exit 1
        ;;
esac

exit 0
```

F. Anexo: Port_Publisher.py

```
#! /usr/bin/env python
"""\
Multi-port serial<->TCP/IP forwarder.
- RFC 2217
- check existence of serial port periodically
- start/stop forwarders
- each forwarder creates a server socket and opens the serial port
- serial ports are opened only once. network connect/disconnect
  does not influence serial port
- only one client per connection
"""
import sys, os, time
import traceback
import socket
import select

import serial
import serial.rfc2217

import avahi
import dbus

class ZeroconfService:
    """\
    A simple class to publish a network service with zeroconf using
avahi.
    """

    def __init__(self, name, port, stype="_http._tcp",
                 domain="", host="", text=""):
        self.name = name
        self.stype = stype
        self.domain = domain
        self.host = host
        self.port = port
        self.text = text
        self.group = None

    def publish(self):
        bus = dbus.SystemBus()
        server = dbus.Interface(
```

```
            bus.get_object(
                avahi.DBUS_NAME,
                avahi.DBUS_PATH_SERVER
            ),
            avahi.DBUS_INTERFACE_SERVER
        )

        g = dbus.Interface(
            bus.get_object(
                avahi.DBUS_NAME,
                server.EntryGroupNew()
            ),
            avahi.DBUS_INTERFACE_ENTRY_GROUP
        )

        g.AddService(avahi.IF_UNSPEC, avahi.PROTO_UNSPEC,
dbus.UInt32(0),
                     self.name, self.stype, self.domain, self.host,
                     dbus.UInt16(self.port), self.text)

        g.Commit()
        self.group = g

    def unpublish(self):
        if self.group is not None:
            self.group.Reset()
            self.group = None

    def __str__(self):
        return "%r @ %s:%s (%s)" % (self.name, self.host, self.port,
self.stype)



class Forwarder(ZeroconfService):
    """\
    Single port serial<->TCP/IP forarder that depends on an external
select
    loop.
    - Buffers for serial -> network and network -> serial
    - RFC 2217 state
    - Zeroconf publish/unpublish on open/close.
    """

    def __init__(self, device, name, network_port, on_close=None):
        ZeroconfService.__init__(self, name, network_port,
stype='_serial_port._tcp')
        self.alive = False
        self.network_port = network_port
        self.on_close = on_close
        self.device = device
        self.serial = serial.Serial()
        self.serial.port = device
        self.serial.baudrate = 9600
        self.serial.timeout = 0
        self.socket = None
```

```
        self.server_socket = None
        self.rfc2217 = None # instantiate later, when connecting

    def __del__(self):
        try:
            if self.alive: self.close()
        except:
            pass # XXX errors on shutdown

    def open(self):
        """open serial port, start network server and publish service"""
        self.buffer_net2ser = ''
        self.buffer_ser2net = ''

        # open serial port
        try:
            self.serial.open()
            self.serial.setRTS(False)
        except Exception, msg:
            self.handle_serial_error(msg)

        self.serial_settings_backup = self.serial.getSettingsDict()

        # start the socket server
        self.server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.server_socket.setsockopt(
            socket.SOL_SOCKET,
            socket.SO_REUSEADDR,
            self.server_socket.getsockopt(
                socket.SOL_SOCKET,
                socket.SO_REUSEADDR
            ) | 1
        )
        self.server_socket.setblocking(0)
        try:
            self.server_socket.bind( ('', self.network_port) )
            self.server_socket.listen(1)
        except socket.error, msg:
            self.handle_server_error()
            #~ raise
        if not options.quiet:
            print "%s: Waiting for connection on %s..." % (self.device,
self.network_port)

        # zeroconfig
        self.publish()

        # now we are ready
        self.alive = True

    def close(self):
        """Close all resources and unpublish service"""
        if not options.quiet:
            print "%s: closing..." % (self.device, )
        self.alive = False
```

```
            self.unpublish()
            if self.server_socket: self.server_socket.close()
            if self.socket:
                self.handle_disconnect()
            self.serial.close()
            if self.on_close is not None:
                # ensure it is only called once
                callback = self.on_close
                self.on_close = None
                callback(self)

        def write(self, data):
            """the write method is used by serial.rfc2217.PortManager. it
has to
            write to the network."""
            self.buffer_ser2net += data

        def update_select_maps(self, read_map, write_map, error_map):
            """Update dictionaries for select call. insert fd->callback
mapping"""
            if self.alive:
                # always handle serial port reads
                read_map[self.serial] = self.handle_serial_read
                error_map[self.serial] = self.handle_serial_error
                # handle serial port writes if buffer is not empty
                if self.buffer_net2ser:
                    write_map[self.serial] = self.handle_serial_write
                # handle network
                if self.socket is not None:
                    # handle socket if connected
                    # only read from network if the internal buffer is not
                    # already filled. the TCP flow control will hold back
data
                    if len(self.buffer_net2ser) < 2048:
                        read_map[self.socket] = self.handle_socket_read
                    # only check for write readiness when there is data
                    if self.buffer_ser2net:
                        write_map[self.socket] = self.handle_socket_write
                    error_map[self.socket] = self.handle_socket_error
                else:
                    # no connection, ensure clear buffer
                    self.buffer_ser2net = ''
                # check the server socket
                read_map[self.server_socket] = self.handle_connect
                error_map[self.server_socket] = self.handle_server_error


        def handle_serial_read(self):
            """Reading from serial port"""
            try:
                data = os.read(self.serial.fileno(), 1024)
                if data:
                    # store data in buffer if there is a client connected
                    if self.socket is not None:
                        # escape outgoing data when needed (Telnet IAC
(0xff) character)
```

```
                    if self.rfc2217:
                        data =
serial.to_bytes(self.rfc2217.escape(data))
                    self.buffer_ser2net += data
            else:
                self.handle_serial_error()
        except Exception, msg:
            self.handle_serial_error(msg)

    def handle_serial_write(self):
        """Writing to serial port"""
        try:
            # write a chunk
            n = os.write(self.serial.fileno(), self.buffer_net2ser)
            # and see how large that chunk was, remove that from buffer
            self.buffer_net2ser = self.buffer_net2ser[n:]
        except Exception, msg:
            self.handle_serial_error(msg)

    def handle_serial_error(self, error=None):
        """Serial port error"""
        # terminate connection
        self.close()

    def handle_socket_read(self):
        """Read from socket"""
        try:
            # read a chunk from the serial port
            data = self.socket.recv(1024)
            if data:
                # Process RFC 2217 stuff when enabled
                if self.rfc2217:
                    data = serial.to_bytes(self.rfc2217.filter(data))
                # add data to buffer
                self.buffer_net2ser += data
            else:
                # empty read indicates disconnection
                self.handle_disconnect()
        except socket.error:
            self.handle_socket_error()

    def handle_socket_write(self):
        """Write to socket"""
        try:
            # write a chunk
            count = self.socket.send(self.buffer_ser2net)
            # and remove the sent data from the buffer
            self.buffer_ser2net = self.buffer_ser2net[count:]
        except socket.error:
            self.handle_socket_error()

    def handle_socket_error(self):
        """Socket connection fails"""
        self.handle_disconnect()

    def handle_connect(self):
```

```
        """Server socket gets a connection"""
        # accept a connection in any case, close connection
        # below if already busy
        connection, addr = self.server_socket.accept()
        if self.socket is None:
            self.socket = connection
            self.socket.setblocking(0)
            self.socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_NODELAY, 1)
            if not options.quiet:
                print '%s: Connected by %s:%s' % (self.device, addr[0],
addr[1])
            self.serial.setRTS(True)
            self.serial.setDTR(True)
            self.rfc2217 = serial.rfc2217.PortManager(self.serial, self)
        else:
            # reject connection if there is already one
            connection.close()
            if not options.quiet:
                print '%s: Rejecting connect from %s:%s' % (self.device,
addr[0], addr[1])

    def handle_server_error(self):
        """Socket server fails"""
        self.close()

    def handle_disconnect(self):
        """Socket gets disconnected"""
        # signal disconnected terminal with control lines
        try:
            self.serial.setRTS(False)
            self.serial.setDTR(False)
        finally:
            # restore original port configuration in case it was changed
            self.serial.applySettingsDict(self.serial_settings_backup)
            # stop RFC 2217 state machine
            self.rfc2217 = None
            # clear send buffer
            self.buffer_ser2net = ''
            # close network connection
            if self.socket is not None:
                self.socket.close()
                self.socket = None
                if not options.quiet:
                    print '%s: Disconnected' % self.device


def test():
    service = ZeroconfService(name="TestService", port=3000)
    service.publish()
    raw_input("Press any key to unpublish the service ")
    service.unpublish()


if __name__ == '__main__':
    import optparse
```

```
    parser = optparse.OptionParser(usage="""\
%prog [options]

Announce the existence of devices using zeroconf and provide
a TCP/IP <-> serial port gateway (implements RFC 2217).

Note that the TCP/IP server is not protected. Everyone can connect
to it!

If running as daemon, write to syslog. Otherwise write to stdout.
""")

    parser.add_option("-q", "--quiet", dest="quiet",
action="store_true",
        help="suppress non error messages", default=False)

    parser.add_option("-o", "--logfile", dest="log_file",
        help="write messages file instead of stdout", default=None,
metavar="FILE")

    parser.add_option("-d", "--daemon", dest="daemonize",
action="store_true",
            help="start as daemon", default=False)

    parser.add_option("", "--pidfile", dest="pid_file",
        help="specify a name for the PID file", default=None,
metavar="FILE")

    (options, args) = parser.parse_args()

    # redirect output if specified
    if options.log_file is not None:
        class WriteFlushed:
            def __init__(self, fileobj):
                self.fileobj = fileobj
            def write(self, s):
                self.fileobj.write(s)
                self.fileobj.flush()
            def close(self):
                self.fileobj.close()
                sys.stdout = sys.stderr =
WriteFlushed(open(options.log_file, 'a'))
        # atexit.register(lambda: sys.stdout.close())

    if options.daemonize:
        # if running as daemon is requested, do the fork magic
        # options.quiet = True
        import pwd
        # do the UNIX double-fork magic, see Stevens' "Advanced
        # Programming in the UNIX Environment" for details (ISBN
0201563177)
        try:
            pid = os.fork()
            if pid > 0:
                # exit first parent
```

```
                sys.exit(0)
        except OSError, e:
            sys.stderr.write("fork #1 failed: %d (%s)\n" % (e.errno,
e.strerror))
            sys.exit(1)

        # decouple from parent environment
        os.chdir("/")   # don't prevent unmounting....
        os.setsid()
        os.umask(0)

        # do second fork
        try:
            pid = os.fork()
            if pid > 0:
                # exit from second parent, print eventual PID before
                # print "Daemon PID %d" % pid
                if options.pid_file is not None:
                    open(options.pid_file,'w').write("%d"%pid)
                sys.exit(0)
        except OSError, e:
            sys.stderr.write("fork #2 failed: %d (%s)\n" % (e.errno,
e.strerror))
            sys.exit(1)

        if options.log_file is None:
            import syslog
            syslog.openlog("serial port publisher")
            # redirect output to syslog
            class WriteToSysLog:
                def __init__(self):
                    self.buffer = ''
                def write(self, s):
                    self.buffer += s
                    if '\n' in self.buffer:
                        output, self.buffer = self.buffer.split('\n', 1)
                        syslog.syslog(output)
                def flush(self):
                    syslog.syslog(self.buffer)
                    self.buffer = ''
                def close(self):
                    self.flush()
            sys.stdout = sys.stderr = WriteToSysLog()

            # ensure the that the daemon runs a normal user, if run as
root
        #if os.getuid() == 0:
            #    name, passwd, uid, gid, desc, home, shell =
pwd.getpwnam('someuser')
            #    os.setgid(gid)     # set group first
            #    os.setuid(uid)     # set user

    # keep the published stuff in a dictionary
    published = {}
    # prepare list of device names (hard coded)
    device_list = ['/dev/ttyAMA%d' % p for p in range(8)]
```

```
    # get a nice hostname
    hostname = socket.gethostname()

    def unpublish(forwarder):
        """when forwarders die, we need to unregister them"""
        try:
            del published[forwarder.device]
        except KeyError:
            pass
        else:
            if not options.quiet: print "unpublish: %s" % (forwarder)

    alive = True
    next_check = 0
    # main loop
    while alive:
        try:
            # if it is time, check for serial port devices
            now = time.time()
            if now > next_check:
                next_check = now + 5
                # check each device
                for device in device_list:
                    # if it appeared
                    if os.path.exists(device):
                        if device not in published:
                            num = int(device[-1])
                            published[device] = Forwarder(
                                device,
                                "%s on %s" % (device, hostname),
                                7000+num,
                                on_close=unpublish
                            )
                            if not options.quiet: print "publish: %s" %
(published[device])
                            published[device].open()
                    else:
                        # or when it disappeared
                        if device in published:
                            if not options.quiet: print "unpublish: %s"
% (published[device])
                            published[device].close()
                            try:
                                del published[device]
                            except KeyError:
                                pass

            # select_start = time.time()
            read_map = {}
            write_map = {}
            error_map = {}
            for publisher in published.values():
                publisher.update_select_maps(read_map, write_map,
error_map)
            try:
                readers, writers, errors = select.select(
```

```
                read_map.keys(),
                write_map.keys(),
                error_map.keys(),
                5
            )
        except select.error, err:
            if err[0] != EINTR:
                raise
        # select_end = time.time()
        # print "select used %.3f s" % (select_end - select_start)
        for reader in readers:
            read_map[reader]()
        for writer in writers:
            write_map[writer]()
        for error in errors:
            error_map[error]()
        # print "operation used %.3f s" % (time.time() - select_end)
    except KeyboardInterrupt:
        alive = False
    except SystemExit:
        raise
    except:
        #~ raise
        traceback.print_exc()
```

G. Anexo: Circuito Sensor

Para el subsistema de pesaje se usó una celda de carga con capacidad para 45 kilogramos, este elemento posee las siguientes características:

- Voltaje de alimentación: 5V – 12V DC.
- Sensibilidad: 2±0.2mV/V.
- Salida de voltaje lineal.

Estas características generan las siguientes necesidades:

- Alimentación entre los voltajes mencionados.
- Acondicionamiento de la señal de salida.
- Uso de un canal conversor análogo digital del microcontrolador.

Con el fin de suplir las necesidades mencionadas, hay que tener en cuenta que la señal de salida cumple la relación:

$$Señal = \frac{Carga\ x\ Sensibilidad\ x\ V_{alimentacion}}{Capacidad\ Maxima}$$

Donde, con un voltaje típico de alimentación de 10V, tendremos una señal máxima de salida de 20±2mV, la cual debe ser amplificada a 1,5 voltios por cuestiones de acoplabilidad con el sistema de conversión ADC implementado en el microcontrolador. Para este fin es necesario el uso de un amplificador operacional de instrumentación para el acondicionamiento de la señal. El amplificador INA128 cumple la siguiente relación de ganancia:

$$G = 1 + \frac{50k\Omega}{R_G}$$

Así pues para conseguir una salida máxima de 1.5V, es decir una ganancia en la señal de salida de 75 veces la señal de entrada, se debe cumplir que $R_G = \frac{50k\Omega}{\left(\frac{1500mV}{20mV}\right)-1}$. El valor comercial $R_G = 680\Omega$,

satisface dicha relación de tal forma que el voltaje de salida máximo tendrá un valor de 1.490V.

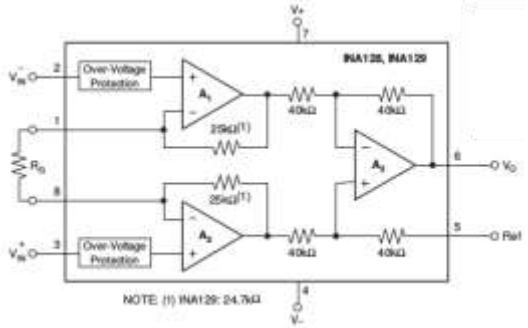


Figura 28. Amplificador de Instrumentación INA128

Teniendo en cuenta que la profundidad del conversor ADC del microcontrolador MSP430F2013 es de 16 bits, tendremos una resolución de 0,6866 g/bit para el caso de una celda de 45 Kg.

Adicionalmente, se han usado sensores de humedad relativa HIH-4000-002 (Figura 29) los cuales cuentan con las siguientes características:

- Voltaje de alimentacion:4 – 5.8 V DC
- Salida lineal de 0.8 a 3.8 V DC aprox. (**¡Error! No se encuentra el origen e la referencia.**Figura 29)

En el caso de estos sensores, el problema de acondicionamiento de la señal, será el opuesto al considerado para la celda de carga: la señal de salida debe ser atenuada. Para este fin se propone la implementación de un circuito de atenuación o división de voltaje como el observado en la Figura 30, el cual cumple la siguiente relación: $V_{out} = \frac{V_i}{1+\frac{R1}{R2}}$

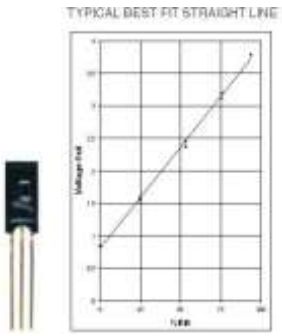


Figura 29. Sensor de Humedad HIH-4000-002 y su salida típica

Así pues, para conseguir un valor máximo de 1.5 V con una entrada de voltaje de 3.8, se debe cumplir que la relación $\frac{R_1}{R_2}= 1.53$ la cual se satisface suficientemente para los valores comerciales de $R_1= 10\,K\Omega$ y $R_1= 6.5\,K\Omega$.

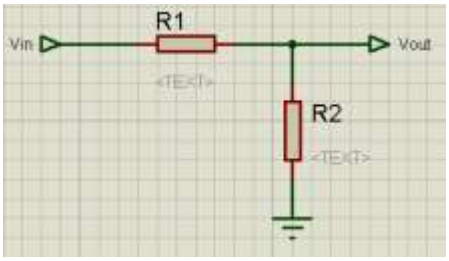


Figura 30. Circuito divisor de voltaje

Por un análisis similar al ya realizado con la celda de carga, con un conversor ADC de 10 bits, se tendrá una resolución de 0.097%RH/bit. Finalmente, respecto a los sensores de temperatura, se ha hecho uso de sensores LM35 (Figura 31), los cuales cuentan con un bajo costo y una alta resolución de medición de 0.1466°C/bit.

Dichos sensores entregan una salida de voltaje lineal de 0 a 1.5V ajustándose perfectamente a los niveles de referencia usados en el circuito ADC del microcontrolador por lo que no se hace necesaria una etapa previa de acondicionamiento de la señal.

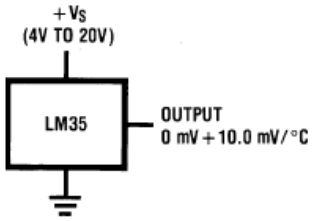


Figura 31. Diagrama del sensor LM35

- Programación del conversor ADC del MSP430

Las consideraciones planteadas con anterioridad, se han postulado teniendo como base las características del circuito ADC del microcontrolador MSP430G2553, para el cual se tienen tres opciones de valor de referencia preestablecida (1.5V, 2.5V y 3.3V). Se opta por el primero de ellos por razones de ajuste respecto a las señales entregadas por los sensores mencionados.

La configuración del valor de referencia en 1.5V, se realiza mediante el registro ADC10CTL0, el cual posee la siguiente estructura:

15	14	13	12	11	10	9	8
SREFx			ADC10SHTx		ADC10SR	REFOUT	REFBURST
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	REF2_5V	REFON	ADC10ON	ADC10IE	ADC10IFG	ENC	ADC10SC
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ENC = 0

Figura 32. Registro ADC10CTL0 del MSP430G2553

De tal forma que si el valor de REF2_5V es igual a 0, se generará una referencia interna de 1.5 V. Teniendo en cuenta que cada microcontrolador posee un máximo de 8 entradas análogas para ser usadas con el conversor ADC y que dos de ellas pueden ser reservadas para el uso de una referencia externa en posibles aplicaciones de gran utilidad, se evidencia la necesidad de usar dos microcontroladores para cubrir la totalidad de sensores requeridos. Tales microcontroladores deben comunicarse entre sí, de tal forma que sea uno de ellos quien recolecte la información de todos los sensores presentes y la transmita vía serial al computador donde se almacenaran, visualizaran y analizaran los datos obtenidos.

Para este fin se aprovecha la capacidad del microcontrolador MSP430G2553 el cual cuenta con dos módulos USCI, el primero se usara en modo UART para la comunicación asíncrona con el computador a 9600 baudios, caracteres de 8 bits, 1 bit de parada y sin paridad (9600-8N1) y el segundo modulo será utilizado para la comunicación entre los microcontroladores de forma síncrona en modo SPI. Para lograr esto, se debe modificar el registro UCAxCTL0 el cual tiene la siguiente estructura:

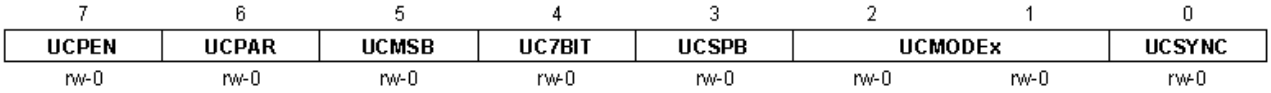


Figura 33. Registro UCAxCTL0 del MSP430G2553

Donde el valor de UCPEN, determina el uso de paridad (0=desactivada, 1=activada).

UCPAR determina el tipo de paridad en caso que UCPEN=1 (0=paridad impar, 1=paridad par).

UCMSB controla la forma de transmisión de los datos (0=LSB primero, 1=MSB primero).

UC7BIT controla la longitud de los datos (0=8 bits, 1=7 bits).

UCSPB controla la cantidad de bits de parada (0=1 bit, 1=2 bits).

UCMODEx determina el modo de conexión (00=UART, 01= multiprocesador Idle, 10=multiprocesador con bit de dirección, 11=UART con detección automática de baudios)

UCSYNC determina el tipo de conexión (0=asíncrona, 1=síncrona)

Este registro controlará el primer modulo USCI (USCI A), para el caso de la comunicación SPI ente los microcontroladores, se usara el modulo USCI B, el cual debe ser configurado mediante el registro UCBxCTL0, estableciendo un microcontrolador como maestro (quien manejara el reloj) y el otro como esclavo. A continuación se observa dicho registro:

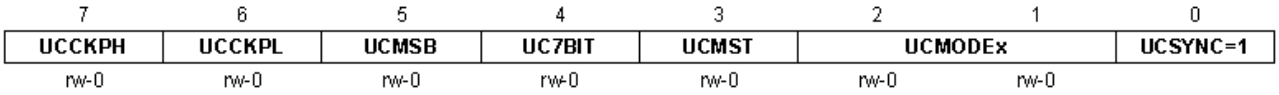


Figura 34. Registro UCBxCTL0 del MSP430G2553

UCCKPH determina la fase del reloj (0=primer flanco para cambio de dato, 1=primer flanco para captura)

UCCKPL controla la polaridad del reloj (0=estado inactivo bajo, 1= estado inactivo alto)

UCMSB controla la forma de transmisión de los datos (0=LSB primero, 1=MSB primero).

UC7BIT controla la longitud de los datos (0=8 bits, 1=7 bits).

UCMST selecciona el modo de operación (0=Esclavo, 1=Maestro)

UCMODEx determina el modo de conexión (00=3-pines SPI, 01= 4-pines SPI, 10=10 4-pin SPI, 11= I2C)

UCSYNC determina el tipo de conexión (0=asíncrona, 1=síncrona)

- Desarrollo del circuito impreso

Las conexiones necesarias para el funcionamiento del sitema modelado, se probaron en una protoboard para luego ser diagramadas mediante el uso de la herramienta ARES, tal diseño se observa en la Figura 17 donde el microcontrolador de la derecha funciona como maestro para la comunicación SPI síncrona y el microcontrolador central realiza la comunicación UART asíncrona con el Raspberry Pi.

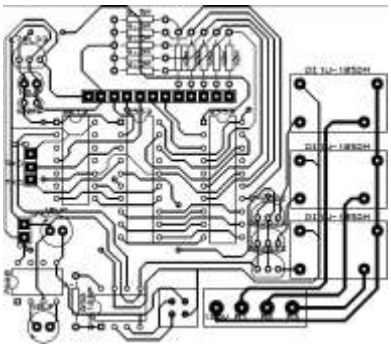


Figura 35. Diseño del circuito impreso para lectura de sensores

- Configuración del Raspberry PI para comunicación con Circuito sensor

Para habilitar el uso del modulo UART, se debe en primer lugar deshabilitar la opción que se tiene de fabrica para habilitar el acceso a la consola desde el puerto serial UART (/dev/ttyAMA0). Para esto se deben seguir los siguientes pasos:

Realizar una copia de seguridad de los archivos de configuración

```
sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt
sudo cp /etc/inittab  /etc/inittab_backup
```

Editar el archivo cmdline.txt

```
sudo nano /boot/cmdline.txt
```

buscar y eliminar los parametros relacionados con el modulo UART (ttyAMA0)

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Editar el archivo inittab (sudo nano /etc/inittab) para comentar la línea relacionada con el modulo UART:

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 11520 vt100
```

Al iniciar de Nuevo el SO, se tendrá la posibilidad de iniciar una conexión serial por el modulo UART.

Finalmente, para usar el modulo UART en nuestros códigos de Python, necesitaremos la librería pyserial [73]. Con ella, podemos crear fácilmente una conexión con el puerto serial del Raspberry, al seguir el siguiente esquema:

```
from serial import Serial      //Importar librerías
import time

serialPort = Serial("/dev/ttyAMA0", 9600, timeout=2) //Establecer
parámetros de conexión serial
if (serialPort.isOpen() == False): serialPort.open() //Abrir la conexión

outStr = 'String Out'  //Declarar variable de salida de datos
inStr = ''             //Declarar variable para recepción de datos

serialPort.flushInput()        //Limpiar el buffer de entrada
serialPort.flushOutput()       //Limpiar el buffer de salida

serialPort.write(outStr)       //Transmitir los datos de salida
time.sleep(0.05)               //Esperar mientras se completa la
Transmisión
inStr = serialPort.read(serialPort.inWaiting()) //Recibir datos de
entrada

serialPort.close()     //Cerrar la conexión
```

De esta manera, se logro establecer la comunicación simultánea entre los microcontroladores y un dispositivo Raspberry Pi. La Figura 36muestra las gráficas obtenidas de la medición de la celda de carga conectada a un nodo Raspberry Pi.

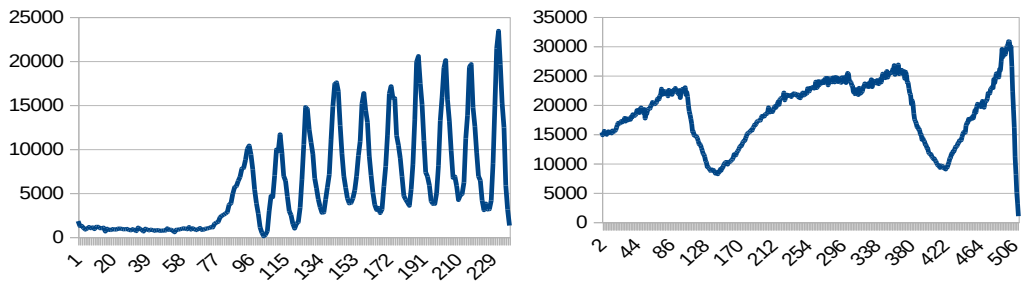


Figura 36. Señales adquiridas de celda de carga por nodo Raspberry Pi

H. Anexo: Detalle de mediciones de ancho de banda con Iperf

- Régimen 1: Transmisión unidireccional de un solo hilo

```
------------------------------------------------------------

Client connecting to 192.168.1.3, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.16 port 60805 connected with 192.168.1.3
port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  1.0- 2.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  2.0- 3.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  3.0- 4.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  4.0- 5.0 sec  1920 KBytes  15729 Kbits/sec
[  3]  5.0- 6.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  6.0- 7.0 sec  1536 KBytes  12583 Kbits/sec
[  3]  7.0- 8.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  8.0- 9.0 sec  1920 KBytes  15729 Kbits/sec
[  3]  9.0-10.0 sec  1792 KBytes  14680 Kbits/sec
[  3] 10.0-11.0 sec  1792 KBytes  14680 Kbits/sec
[  3] 11.0-12.0 sec  1408 KBytes  11534 Kbits/sec
[  3] 12.0-13.0 sec  1664 KBytes  13631 Kbits/sec
[  3] 13.0-14.0 sec  1792 KBytes  14680 Kbits/sec
[  3] 14.0-15.0 sec  1920 KBytes  15729 Kbits/sec
[  3]  0.0-15.1 sec  26368 KBytes  14259 Kbits/sec
Done.
```

- Régimen 2: Transmisión bidireccional secuencial (dúplex) de un solo hilo

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.3, TCP port 5001
```

```
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.16 port 60806 connected with 192.168.1.3
port 5001
[  5] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52347
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0- 1.0 sec   779 KBytes  6378 Kbits/sec
[  3]  0.0- 1.0 sec   896 KBytes  7340 Kbits/sec
[  5]  1.0- 2.0 sec   963 KBytes  7890 Kbits/sec
[  3]  1.0- 2.0 sec   768 KBytes  6291 Kbits/sec
[  5]  2.0- 3.0 sec  1068 KBytes  8748 Kbits/sec
[  3]  2.0- 3.0 sec   640 KBytes  5243 Kbits/sec
[  5]  3.0- 4.0 sec   994 KBytes  8144 Kbits/sec
[  3]  3.0- 4.0 sec   768 KBytes  6291 Kbits/sec
[  5]  4.0- 5.0 sec   874 KBytes  7159 Kbits/sec
[  3]  4.0- 5.0 sec   512 KBytes  4194 Kbits/sec
[  5]  5.0- 6.0 sec   819 KBytes  6707 Kbits/sec
[  3]  5.0- 6.0 sec   640 KBytes  5243 Kbits/sec
[  5]  6.0- 7.0 sec   660 KBytes  5410 Kbits/sec
[  3]  6.0- 7.0 sec   768 KBytes  6291 Kbits/sec
[  3]  7.0- 8.0 sec   640 KBytes  5243 Kbits/sec
[  5]  7.0- 8.0 sec   560 KBytes  4587 Kbits/sec
[  3]  8.0- 9.0 sec   896 KBytes  7340 Kbits/sec
[  5]  8.0- 9.0 sec   580 KBytes  4749 Kbits/sec
[  5]  9.0-10.0 sec   683 KBytes  5595 Kbits/sec
[  3]  9.0-10.0 sec  1280 KBytes  10486 Kbits/sec
[  5] 10.0-11.0 sec   543 KBytes  4448 Kbits/sec
[  3] 10.0-11.0 sec  1408 KBytes  11534 Kbits/sec
[  5] 11.0-12.0 sec   745 KBytes  6105 Kbits/sec
[  3] 11.0-12.0 sec   896 KBytes  7340 Kbits/sec
[  5] 12.0-13.0 sec   663 KBytes  5433 Kbits/sec
[  3] 12.0-13.0 sec  1408 KBytes  11534 Kbits/sec
[  3] 13.0-14.0 sec   640 KBytes  5243 Kbits/sec
[  5] 13.0-14.0 sec   475 KBytes  3892 Kbits/sec
[  5] 14.0-15.0 sec   697 KBytes  5711 Kbits/sec
[  3] 14.0-15.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  0.0-15.3 sec  13952 KBytes  7490 Kbits/sec
[  5] 15.0-16.0 sec   611 KBytes  5004 Kbits/sec
[  5]  0.0-16.3 sec  11904 KBytes  6000 Kbits/sec
Done.
```

- Régimen 3: Transmisión bidireccional simultanea(full-duplex) de un solo hilo

```
------------------------------------------------------------
Server listening on TCP port 5002
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

```
------------------------------------------------------------
Client connecting to 192.168.1.3, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.16 port 60809 connected with 192.168.1.3
port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  1.0- 2.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  2.0- 3.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  3.0- 4.0 sec  1920 KBytes  15729 Kbits/sec
[  3]  4.0- 5.0 sec  2176 KBytes  17826 Kbits/sec
[  3]  5.0- 6.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  6.0- 7.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  7.0- 8.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  8.0- 9.0 sec  1664 KBytes  13631 Kbits/sec
[  3]  9.0-10.0 sec  1152 KBytes  9437 Kbits/sec
[  3] 10.0-11.0 sec  1536 KBytes  12583 Kbits/sec
[  3] 11.0-12.0 sec  1664 KBytes  13631 Kbits/sec
[  3] 12.0-13.0 sec  1536 KBytes  12583 Kbits/sec
[  3] 13.0-14.0 sec  1408 KBytes  11534 Kbits/sec
[  3] 14.0-15.0 sec  2176 KBytes  17826 Kbits/sec
[  3]  0.0-15.0 sec  25728 KBytes  14022 Kbits/sec
[  5] local 192.168.1.16 port 5002 connected with 192.168.1.3 port
45832
[  5]  0.0- 1.0 sec  1662 KBytes  13614 Kbits/sec
[  5]  1.0- 2.0 sec  1765 KBytes  14457 Kbits/sec
[  5]  2.0- 3.0 sec  1595 KBytes  13067 Kbits/sec
[  5]  3.0- 4.0 sec  1695 KBytes  13889 Kbits/sec
[  5]  4.0- 5.0 sec  1878 KBytes  15384 Kbits/sec
[  5]  5.0- 6.0 sec  1786 KBytes  14631 Kbits/sec
[  5]  6.0- 7.0 sec  1700 KBytes  13924 Kbits/sec
[  5]  7.0- 8.0 sec  1828 KBytes  14978 Kbits/sec
[  5]  8.0- 9.0 sec  1711 KBytes  14017 Kbits/sec
[  5]  9.0-10.0 sec  1619 KBytes  13264 Kbits/sec
[  5] 10.0-11.0 sec  1783 KBytes  14607 Kbits/sec
[  5] 11.0-12.0 sec  1840 KBytes  15071 Kbits/sec
[  5] 12.0-13.0 sec  1823 KBytes  14932 Kbits/sec
[  5] 13.0-14.0 sec  1792 KBytes  14677 Kbits/sec
[  5] 14.0-15.0 sec  1843 KBytes  15094 Kbits/sec
[  5]  0.0-15.5 sec  27264 KBytes  14407 Kbits/sec
Done.
```

- Régimen 4: Transmisión unidireccional de cinco hilos paralelos

```
------------------------------------------------------------
Client connecting to 192.168.1.3, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
```

```
[  7] local 192.168.1.16 port 60799 connected with 192.168.1.3
port 5001
[  3] local 192.168.1.16 port 60795 connected with 192.168.1.3
port 5001
[  4] local 192.168.1.16 port 60796 connected with 192.168.1.3
port 5001
[  5] local 192.168.1.16 port 60797 connected with 192.168.1.3
port 5001
[  6] local 192.168.1.16 port 60798 connected with 192.168.1.3
port 5001
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0- 1.0 sec   384 KBytes  3146 Kbits/sec
[  5]  0.0- 1.0 sec   384 KBytes  3146 Kbits/sec
[  7]  0.0- 1.0 sec   512 KBytes  4194 Kbits/sec
[  4]  0.0- 1.0 sec   512 KBytes  4194 Kbits/sec
[  3]  0.0- 1.0 sec   512 KBytes  4194 Kbits/sec
[SUM]  0.0- 1.0 sec  2304 KBytes  18874 Kbits/sec
[  7]  1.0- 2.0 sec   256 KBytes  2097 Kbits/sec
[  4]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  6]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  3]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  5]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  1.0- 2.0 sec  1792 KBytes  14680 Kbits/sec
[  6]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[  3]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[  4]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[  7]  2.0- 3.0 sec   512 KBytes  4194 Kbits/sec
[  5]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  2.0- 3.0 sec  2048 KBytes  16777 Kbits/sec
[  3]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[  6]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[  5]  3.0- 4.0 sec   256 KBytes  2097 Kbits/sec
[  4]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[  7]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  3.0- 4.0 sec  1792 KBytes  14680 Kbits/sec
[  3]  4.0- 5.0 sec   256 KBytes  2097 Kbits/sec
[  5]  4.0- 5.0 sec   256 KBytes  2097 Kbits/sec
[  4]  4.0- 5.0 sec   256 KBytes  2097 Kbits/sec
[  7]  4.0- 5.0 sec   256 KBytes  2097 Kbits/sec
[  6]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  4.0- 5.0 sec  1408 KBytes  11534 Kbits/sec
[  6]  5.0- 6.0 sec   256 KBytes  2097 Kbits/sec
[  4]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[  3]  5.0- 6.0 sec   512 KBytes  4194 Kbits/sec
[  5]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[  7]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  5.0- 6.0 sec  1920 KBytes  15729 Kbits/sec
[  5]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[  6]  6.0- 7.0 sec   512 KBytes  4194 Kbits/sec
[  7]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
```

```
[  4]  6.0- 7.0 sec   512 KBytes  4194 Kbits/sec
[  3]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  6.0- 7.0 sec  2176 KBytes  17826 Kbits/sec
[  6]  7.0- 8.0 sec   256 KBytes  2097 Kbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1049 Kbits/sec
[  4]  7.0- 8.0 sec   256 KBytes  2097 Kbits/sec
[  5]  7.0- 8.0 sec   256 KBytes  2097 Kbits/sec
[  7]  7.0- 8.0 sec   256 KBytes  2097 Kbits/sec
[SUM]  7.0- 8.0 sec  1152 KBytes  9437 Kbits/sec
[  7]  8.0- 9.0 sec   512 KBytes  4194 Kbits/sec
[  5]  8.0- 9.0 sec   640 KBytes  5243 Kbits/sec
[  3]  8.0- 9.0 sec   768 KBytes  6291 Kbits/sec
[  4]  8.0- 9.0 sec   640 KBytes  5243 Kbits/sec
[  6]  8.0- 9.0 sec   640 KBytes  5243 Kbits/sec
[SUM]  8.0- 9.0 sec  3200 KBytes  26214 Kbits/sec
[  7]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  4]  9.0-10.0 sec   256 KBytes  2097 Kbits/sec
[  6]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  5]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  3]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[SUM]  9.0-10.0 sec  2304 KBytes  18874 Kbits/sec
[  7] 10.0-11.0 sec   256 KBytes  2097 Kbits/sec
[  4] 10.0-11.0 sec   384 KBytes  3146 Kbits/sec
[  6] 10.0-11.0 sec   256 KBytes  2097 Kbits/sec
[  5] 10.0-11.0 sec   256 KBytes  2097 Kbits/sec
[  3] 10.0-11.0 sec   256 KBytes  2097 Kbits/sec
[SUM] 10.0-11.0 sec  1408 KBytes  11534 Kbits/sec
[  4] 11.0-12.0 sec   384 KBytes  3146 Kbits/sec
[  6] 11.0-12.0 sec   256 KBytes  2097 Kbits/sec
[  3] 11.0-12.0 sec   256 KBytes  2097 Kbits/sec
[  5] 11.0-12.0 sec   256 KBytes  2097 Kbits/sec
[  7] 11.0-12.0 sec   512 KBytes  4194 Kbits/sec
[SUM] 11.0-12.0 sec  1664 KBytes  13631 Kbits/sec
[  7] 12.0-13.0 sec   256 KBytes  2097 Kbits/sec
[  4] 12.0-13.0 sec   512 KBytes  4194 Kbits/sec
[  6] 12.0-13.0 sec   512 KBytes  4194 Kbits/sec
[  3] 12.0-13.0 sec   512 KBytes  4194 Kbits/sec
[  5] 12.0-13.0 sec   512 KBytes  4194 Kbits/sec
[SUM] 12.0-13.0 sec  2304 KBytes  18874 Kbits/sec
[  7] 13.0-14.0 sec   256 KBytes  2097 Kbits/sec
[  6] 13.0-14.0 sec   256 KBytes  2097 Kbits/sec
[  4] 13.0-14.0 sec   256 KBytes  2097 Kbits/sec
[  5] 13.0-14.0 sec   256 KBytes  2097 Kbits/sec
[  3] 13.0-14.0 sec   256 KBytes  2097 Kbits/sec
[SUM] 13.0-14.0 sec  1280 KBytes  10486 Kbits/sec
[  3] 14.0-15.0 sec   256 KBytes  2097 Kbits/sec
[  3]  0.0-15.2 sec  5888 KBytes  3167 Kbits/sec
[  4] 14.0-15.0 sec   256 KBytes  2097 Kbits/sec
[  4]  0.0-15.3 sec  5888 KBytes  3158 Kbits/sec
[  6] 14.0-15.0 sec   512 KBytes  4194 Kbits/sec
[  6]  0.0-15.9 sec  6016 KBytes  3108 Kbits/sec
```

```
[  7] 14.0-15.0 sec   512 KBytes  4194 Kbits/sec
[  7]  0.0-15.9 sec  5888 KBytes  3041 Kbits/sec
[  5] 14.0-15.0 sec   384 KBytes  3146 Kbits/sec
[SUM] 14.0-15.0 sec  1920 KBytes  15729 Kbits/sec
[  5]  0.0-16.5 sec  5632 KBytes  2801 Kbits/sec
[SUM]  0.0-16.5 sec  29312 KBytes  14580 Kbits/sec
Done.
```

- Régimen 5: Transmisión bidireccional secuencial (dúplex) de cinco hilos paralelos

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.3, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  9] local 192.168.1.16 port 60820 connected with 192.168.1.3
port 5001
[  5] local 192.168.1.16 port 60816 connected with 192.168.1.3
port 5001
[  6] local 192.168.1.16 port 60817 connected with 192.168.1.3
port 5001
[  7] local 192.168.1.16 port 60818 connected with 192.168.1.3
port 5001
[  8] local 192.168.1.16 port 60819 connected with 192.168.1.3
port 5001
[ ID] Interval       Transfer     Bandwidth
[  7]  0.0- 1.0 sec   256 KBytes  2097 Kbits/sec
[  8]  0.0- 1.0 sec   256 KBytes  2097 Kbits/sec
[  9]  0.0- 1.0 sec   384 KBytes  3146 Kbits/sec
[  6]  0.0- 1.0 sec   256 KBytes  2097 Kbits/sec
[  5]  0.0- 1.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  0.0- 1.0 sec  1536 KBytes  12583 Kbits/sec
[  5]  1.0- 2.0 sec   256 KBytes  2097 Kbits/sec
[  7]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  9]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  6]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[  8]  1.0- 2.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  1.0- 2.0 sec  1792 KBytes  14680 Kbits/sec
[  9]  2.0- 3.0 sec   256 KBytes  2097 Kbits/sec
[  6]  2.0- 3.0 sec   256 KBytes  2097 Kbits/sec
[  5]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[  7]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[  8]  2.0- 3.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  2.0- 3.0 sec  1664 KBytes  13631 Kbits/sec
[  5]  3.0- 4.0 sec   256 KBytes  2097 Kbits/sec
[  8]  3.0- 4.0 sec   256 KBytes  2097 Kbits/sec
```

```
[  9]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[  7]  3.0- 4.0 sec   256 KBytes  2097 Kbits/sec
[  6]  3.0- 4.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  3.0- 4.0 sec  1536 KBytes  12583 Kbits/sec
[  9]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[  5]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[  7]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[  8]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[  6]  4.0- 5.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  4.0- 5.0 sec  1920 KBytes  15729 Kbits/sec
[  9]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[  6]  5.0- 6.0 sec   256 KBytes  2097 Kbits/sec
[  8]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[  5]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[  7]  5.0- 6.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  5.0- 6.0 sec  1792 KBytes  14680 Kbits/sec
[  6]  6.0- 7.0 sec   256 KBytes  2097 Kbits/sec
[  9]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[  5]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[  7]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[  8]  6.0- 7.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  6.0- 7.0 sec  1792 KBytes  14680 Kbits/sec
[  8]  7.0- 8.0 sec   256 KBytes  2097 Kbits/sec
[  5]  7.0- 8.0 sec   384 KBytes  3146 Kbits/sec
[  9]  7.0- 8.0 sec   384 KBytes  3146 Kbits/sec
[  7]  7.0- 8.0 sec   384 KBytes  3146 Kbits/sec
[  6]  7.0- 8.0 sec   384 KBytes  3146 Kbits/sec
[SUM]  7.0- 8.0 sec  1792 KBytes  14680 Kbits/sec
[  8]  8.0- 9.0 sec   384 KBytes  3146 Kbits/sec
[  9]  8.0- 9.0 sec   384 KBytes  3146 Kbits/sec
[  5]  8.0- 9.0 sec   384 KBytes  3146 Kbits/sec
[  7]  8.0- 9.0 sec   256 KBytes  2097 Kbits/sec
[  9]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  5]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  7]  9.0-10.0 sec   384 KBytes  3146 Kbits/sec
[  8]  9.0-10.0 sec   512 KBytes  4194 Kbits/sec
[  6]  8.0- 9.0 sec   256 KBytes  2097 Kbits/sec
[SUM]  8.0- 9.0 sec  1664 KBytes  13631 Kbits/sec
[  5] 10.0-11.0 sec   384 KBytes  3146 Kbits/sec
[  8] 10.0-11.0 sec   384 KBytes  3146 Kbits/sec
[  7] 10.0-11.0 sec   384 KBytes  3146 Kbits/sec
[  9] 10.0-11.0 sec   384 KBytes  3146 Kbits/sec
[  6]  9.0-10.0 sec   128 KBytes  1049 Kbits/sec
[SUM]  9.0-10.0 sec  2048 KBytes  16777 Kbits/sec
[  5] 11.0-12.0 sec   512 KBytes  4194 Kbits/sec
[  6] 10.0-11.0 sec   128 KBytes  1049 Kbits/sec
[SUM] 10.0-11.0 sec  1664 KBytes  13631 Kbits/sec
[  9] 11.0-12.0 sec   512 KBytes  4194 Kbits/sec
[  7] 11.0-12.0 sec   512 KBytes  4194 Kbits/sec
[  8] 11.0-12.0 sec   512 KBytes  4194 Kbits/sec
[  5] 12.0-13.0 sec   384 KBytes  3146 Kbits/sec
```

```
[  9] 12.0-13.0 sec   256 KBytes  2097 Kbits/sec
[  6] 11.0-12.0 sec   128 KBytes  1049 Kbits/sec
[SUM] 11.0-12.0 sec  2176 KBytes  17826 Kbits/sec
[  7] 12.0-13.0 sec   256 KBytes  2097 Kbits/sec
[  8] 12.0-13.0 sec   256 KBytes  2097 Kbits/sec
[  9] 13.0-14.0 sec   384 KBytes  3146 Kbits/sec
[  6] 12.0-13.0 sec   128 KBytes  1049 Kbits/sec
[SUM] 12.0-13.0 sec  1280 KBytes  10486 Kbits/sec
[  8] 13.0-14.0 sec   384 KBytes  3146 Kbits/sec
[  5] 13.0-14.0 sec   384 KBytes  3146 Kbits/sec
[  7] 13.0-14.0 sec   384 KBytes  3146 Kbits/sec
[  6] 13.0-14.0 sec   128 KBytes  1049 Kbits/sec
[SUM] 13.0-14.0 sec  1664 KBytes  13631 Kbits/sec
[  5] 14.0-15.0 sec   512 KBytes  4194 Kbits/sec
[  5]  0.0-15.5 sec  6016 KBytes  3181 Kbits/sec
[  7] 14.0-15.0 sec   384 KBytes  3146 Kbits/sec
[  7]  0.0-15.6 sec  5504 KBytes  2886 Kbits/sec
[  8] 14.0-15.0 sec   512 KBytes  4194 Kbits/sec
[  8]  0.0-15.7 sec  5760 KBytes  2999 Kbits/sec
[  9] 14.0-15.0 sec   640 KBytes  5243 Kbits/sec
[  9]  0.0-15.9 sec  6144 KBytes  3165 Kbits/sec
[  6] 14.0-15.0 sec   256 KBytes  2097 Kbits/sec
[SUM] 14.0-15.0 sec  2304 KBytes  18874 Kbits/sec
[  6] 15.0-16.0 sec  0.00 KBytes  0.00 Kbits/sec
[  6]  0.0-16.4 sec  3840 KBytes  1914 Kbits/sec
[SUM]  0.0-16.4 sec  27264 KBytes  13590 Kbits/sec
[  4] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52378
[  5] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52379
[  6] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52377
[  7] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52380
[  8] local 192.168.1.16 port 5001 connected with 192.168.1.3 port
52381
[  4]  0.0- 1.0 sec   257 KBytes  2108 Kbits/sec
[  6]  0.0- 1.0 sec   297 KBytes  2433 Kbits/sec
[  8]  0.0- 1.0 sec   284 KBytes  2328 Kbits/sec
[  5]  0.0- 1.0 sec   321 KBytes  2630 Kbits/sec
[  7]  0.0- 1.0 sec   354 KBytes  2896 Kbits/sec
[SUM]  0.0- 1.0 sec  1513 KBytes  12395 Kbits/sec
[  6]  1.0- 2.0 sec   325 KBytes  2664 Kbits/sec
[  7]  1.0- 2.0 sec   475 KBytes  3892 Kbits/sec
[  5]  1.0- 2.0 sec   427 KBytes  3498 Kbits/sec
[  4]  1.0- 2.0 sec   264 KBytes  2166 Kbits/sec
[  8]  1.0- 2.0 sec   297 KBytes  2433 Kbits/sec
[SUM]  1.0- 2.0 sec  1789 KBytes  14654 Kbits/sec
[  7]  2.0- 3.0 sec   452 KBytes  3707 Kbits/sec
[  4]  2.0- 3.0 sec   242 KBytes  1981 Kbits/sec
```

```
[  5]  2.0- 3.0 sec   433 KBytes  3545 Kbits/sec
[  6]  2.0- 3.0 sec   293 KBytes  2398 Kbits/sec
[  8]  2.0- 3.0 sec   297 KBytes  2433 Kbits/sec
[SUM]  2.0- 3.0 sec  1717 KBytes  14063 Kbits/sec
[  5]  3.0- 4.0 sec   411 KBytes  3371 Kbits/sec
[  6]  3.0- 4.0 sec   327 KBytes  2676 Kbits/sec
[  8]  3.0- 4.0 sec   254 KBytes  2081 Kbits/sec
[  4]  3.0- 4.0 sec   267 KBytes  2185 Kbits/sec
[  7]  3.0- 4.0 sec   489 KBytes  4008 Kbits/sec
[SUM]  3.0- 4.0 sec  1748 KBytes  14321 Kbits/sec
[  8]  4.0- 5.0 sec   298 KBytes  2444 Kbits/sec
[  6]  4.0- 5.0 sec   281 KBytes  2305 Kbits/sec
[  5]  4.0- 5.0 sec   417 KBytes  3417 Kbits/sec
[  7]  4.0- 5.0 sec   445 KBytes  3649 Kbits/sec
[  4]  4.0- 5.0 sec   259 KBytes  2120 Kbits/sec
[SUM]  4.0- 5.0 sec  1701 KBytes  13936 Kbits/sec
[  5]  5.0- 6.0 sec   437 KBytes  3579 Kbits/sec
[  6]  5.0- 6.0 sec   342 KBytes  2803 Kbits/sec
[  7]  5.0- 6.0 sec   486 KBytes  3985 Kbits/sec
[  8]  5.0- 6.0 sec   324 KBytes  2653 Kbits/sec
[  4]  5.0- 6.0 sec   288 KBytes  2363 Kbits/sec
[SUM]  5.0- 6.0 sec  1878 KBytes  15384 Kbits/sec
[  5]  6.0- 7.0 sec   355 KBytes  2908 Kbits/sec
[  4]  6.0- 7.0 sec   303 KBytes  2479 Kbits/sec
[  6]  6.0- 7.0 sec   332 KBytes  2722 Kbits/sec
[  7]  6.0- 7.0 sec   345 KBytes  2826 Kbits/sec
[  8]  6.0- 7.0 sec   311 KBytes  2548 Kbits/sec
[SUM]  6.0- 7.0 sec  1646 KBytes  13484 Kbits/sec
[  4]  7.0- 8.0 sec   344 KBytes  2815 Kbits/sec
[  5]  7.0- 8.0 sec   389 KBytes  3186 Kbits/sec
[  6]  7.0- 8.0 sec   345 KBytes  2826 Kbits/sec
[  8]  7.0- 8.0 sec   349 KBytes  2861 Kbits/sec
[  7]  7.0- 8.0 sec   411 KBytes  3371 Kbits/sec
[SUM]  7.0- 8.0 sec  1838 KBytes  15059 Kbits/sec
[  4]  8.0- 9.0 sec   334 KBytes  2734 Kbits/sec
[  5]  8.0- 9.0 sec   296 KBytes  2421 Kbits/sec
[  7]  8.0- 9.0 sec   286 KBytes  2340 Kbits/sec
[  8]  8.0- 9.0 sec   332 KBytes  2722 Kbits/sec
[  6]  8.0- 9.0 sec   345 KBytes  2826 Kbits/sec
[SUM]  8.0- 9.0 sec  1592 KBytes  13044 Kbits/sec
[  4]  9.0-10.0 sec   328 KBytes  2687 Kbits/sec
[  5]  9.0-10.0 sec   342 KBytes  2803 Kbits/sec
[  6]  9.0-10.0 sec   342 KBytes  2803 Kbits/sec
[  8]  9.0-10.0 sec   320 KBytes  2618 Kbits/sec
[  7]  9.0-10.0 sec   363 KBytes  2977 Kbits/sec
[SUM]  9.0-10.0 sec  1695 KBytes  13889 Kbits/sec
[  6] 10.0-11.0 sec   286 KBytes  2340 Kbits/sec
[  8] 10.0-11.0 sec   297 KBytes  2433 Kbits/sec
[  7] 10.0-11.0 sec   297 KBytes  2433 Kbits/sec
[  5] 10.0-11.0 sec   315 KBytes  2583 Kbits/sec
[  4] 10.0-11.0 sec   141 KBytes  1158 Kbits/sec
```

```
[SUM] 10.0-11.0 sec  1336 KBytes  10947 Kbits/sec
[  5] 11.0-12.0 sec   230 KBytes  1888 Kbits/sec
[  8] 11.0-12.0 sec   365 KBytes  2989 Kbits/sec
[  6] 11.0-12.0 sec   376 KBytes  3081 Kbits/sec
[  7] 11.0-12.0 sec   395 KBytes  3232 Kbits/sec
[  4] 11.0-12.0 sec   375 KBytes  3070 Kbits/sec
[SUM] 11.0-12.0 sec  1741 KBytes  14260 Kbits/sec
[  6] 12.0-13.0 sec   420 KBytes  3440 Kbits/sec
[  8] 12.0-13.0 sec   380 KBytes  3116 Kbits/sec
[  4] 12.0-13.0 sec   230 KBytes  1888 Kbits/sec
[  7] 12.0-13.0 sec   389 KBytes  3186 Kbits/sec
[  5] 12.0-13.0 sec   262 KBytes  2143 Kbits/sec
[SUM] 12.0-13.0 sec  1681 KBytes  13773 Kbits/sec
[  4] 13.0-14.0 sec   263 KBytes  2155 Kbits/sec
[  5] 13.0-14.0 sec   263 KBytes  2155 Kbits/sec
[  6] 13.0-14.0 sec   443 KBytes  3626 Kbits/sec
[  7] 13.0-14.0 sec   431 KBytes  3533 Kbits/sec
[  8] 13.0-14.0 sec   433 KBytes  3545 Kbits/sec
[SUM] 13.0-14.0 sec  1833 KBytes  15013 Kbits/sec
[  4] 14.0-15.0 sec   270 KBytes  2213 Kbits/sec
[  5] 14.0-15.0 sec   297 KBytes  2433 Kbits/sec
[  6] 14.0-15.0 sec   383 KBytes  3139 Kbits/sec
[  8] 14.0-15.0 sec   389 KBytes  3186 Kbits/sec
[  7] 14.0-15.0 sec   396 KBytes  3244 Kbits/sec
[SUM] 14.0-15.0 sec  1735 KBytes  14214 Kbits/sec
[  6] 15.0-16.0 sec   407 KBytes  3336 Kbits/sec
[  8] 15.0-16.0 sec   363 KBytes  2977 Kbits/sec
[  7] 15.0-16.0 sec   409 KBytes  3348 Kbits/sec
[  5] 15.0-16.0 sec   300 KBytes  2456 Kbits/sec
[  4] 15.0-16.0 sec   288 KBytes  2363 Kbits/sec
[SUM] 15.0-16.0 sec  1768 KBytes  14480 Kbits/sec
[  8]  0.0-16.2 sec  5376 KBytes  2724 Kbits/sec
[  7]  0.0-16.2 sec  6528 KBytes  3292 Kbits/sec
[  5]  0.0-16.3 sec  5632 KBytes  2827 Kbits/sec
[  6]  0.0-16.3 sec  5760 KBytes  2887 Kbits/sec
[  4]  0.0-16.4 sec  4736 KBytes  2363 Kbits/sec
[SUM]  0.0-16.4 sec  28032 KBytes  13985 Kbits/sec
Done.
```

BIBLIOGRAFÍA

[1] W. Adarme, M. Arango, and A. Otero, “Coordinación de abastecimiento con información compartida, inventario gestionado por el vendedor, en pymes agroalimentarias Colombianas.,” *Rev. DYNA*, vol. 167, no. 78, 2011.

[2] Grupo de Investigación SEPRO and Wilson Adarme Jaimes, “Propuesta metodológica para coordinar procesos logísticos de producción y distribución de cacao y plátano en las zonas de Caricare y Caño Limón,” Universidad Nacional de Colombia, Bogotá, D.C., Documento de Presentación de Resultados de Investigación, Nov. 2012.

[3] J. Bookbinder and J. Higginson, “Probabilistic modeling of freight consolidation by private carriage,” *Transp. Res. Part E*, vol. 38, pp. 305–318, 2002.

[4] M. Khouja and S. Goyal, “A review of the joint replenishment problem literature: 1989–2005,” *Eur. J. Oper. Res.*, vol. 186, no. 1, pp. 1–16, Apr. 2008.

[5] O. Ahumada and J. R. Villalobos, “Application of planning models in the agri-food supply chain: A review,” *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 1–20, 2009.

[6] R. Akkerman, P. Farahani, and M. Grunow, “Quality, safety and sustainability in food distribution: a review of quantitative operations management approaches and challenges,” *Spectr.*, vol. 32, no. 4, pp. 863–904, Aug. 2010.

[7] A. Rong, R. Akkerman, and M. Grunow, “An optimization approach for managing fresh food quality throughout the supply chain,” *Int. J. Prod. Econ.*, vol. 131, no. 1, pp. 421–429, May 2011.

[8] FAO, “FAO/World Bank workshop on reducing post-harvest losses in grain supply chains in Africa.” The World Bank, 2010.

[9] José Facundo Castillo Cisneros, “Plan de Desarrollo Departamental 2012-2015.” 2012.

[10] DANE - Banco de la República, “Informe de Coyuntura Económica Regional Departamento de Arauca.” Sep-2013.

[11] Portafolio.com.co, “Cacao colombiano gana en Salón del Chocolate de París - Portafolio.co,” *Portafolio.com.co*, 05-Nov-2010. [Online]. Available: http://www.portafolio.co/detalle_archivo/MAM-4237454. [Accessed: 21-Jul-2014].

[12] DNP, “Visión de Desarrollo Territorial Departamental - Visión Arauca 2032: Geoestratégica, innovadora y nuestra.” 2011.

[13] Gobernación de Arauca, “Plan Regional de Competitividad de Arauca.” Mar-2011.

[14] Ministerio de Agricultura y Desarrollo Rural, “Anuario Estadístico del Sector Agropecuario 2012.” Sep-2013.

[15] Oficina de Estudios Económicos - Min. CIT, “Perfil económico: Departamento de Arauca.” 04-Apr-2014.

[16] Comisión Económica para América Latina y el Caribe, CEPAL, “Programa,” presented at the Políticas para la agricultura en américa latina y el caribe: competitividad, sostenibilidad e inclusión social, Santiago de Chile, 2011.

[17] Ministerio de Agricultura y Desarrollo Rural, “Agronet.” [Online]. Available: http://www.agronet.gov.co/agronetweb1/. [Accessed: 30-Jul-2014].

[18] Alcaldía de Medellín, DAGRED, EPM, and ISAGEN, "SIATA - Sistema de Alerta Temprana del valle de Aburrá." [Online]. Available: http://www.siata.gov.co. [Accessed: 12-May-2014].
[19] Maitri Thakur and Charles R. Hurburgh, "Framework for implementing traceability system in the bulk grain supply chain." 2009.
[20] Aiying Rong and Martin Grunow, "A methodology for controlling dispersion in food production and distribution." 2010.
[21] J.-F. Cordeau, F. Pasin, and M. M. Solomon, "An integrated model for logistics network design," *Ann. Oper. Res.*, vol. 144, no. 1, pp. 59–82, 2006.
[22] IETF, "Manet Status Pages." [Online]. Available: http://tools.ietf.org/wg/manet/. [Accessed: 26-May-2015].
[23] Carlos de Morais Cordeiro and Dharma Prakash Agrawal, "Integrating MANETs, WLANs, and Cellular Networks," in *Ad Hoc and Sensor Networks*, 0 vols., WORLD SCIENTIFIC, 2011, pp. 587–619.
[24] Leonardo Rodríguez Mújica and Milton J. Ríos Rivera, "The solution for the construction topology problem for rural wireless networks," in *IEEE COLCOM 2015.*, 2015, p. -.
[25] D. Panigrahi, P. Duttat, S. Jaiswal, K. V. M. Naidu, and R. Rastogi, "Minimum Cost Topology Construction for Rural Wireless Mesh Networks," in *IEEE INFOCOM 2008. The 27th Conference on Computer Communications*, 2008, p. -.
[26] CHRISTOPHER, M., "Logistics and Supply Chain Management. Financial Times," *Kim, C, Tannock, J, Byrne, M et al. (2004). State-of-the-art review Techniques to model the supply Chain in an extended enterprise. (VIVACE WP2.5)*, Nottingham, England: University of Nottingham, Operations Management Division., 1998.
[27] JOHANSSON, M., "The impact of supply integration and information flow on supply chain performance (Tesis)," *Kim B Leung JM Park K T Zhang G Lee 2002 Config. Manuf. Firms Supply Netw. Mult. Suppliers IIE Trans.*, vol. 34, pp. 663–677.
[28] C. D. J. Waters, *Logistics: An Introduction to Supply Chain Management*. Palgrave Macmillan, 2003.
[29] E. Sucky, "The bullwhip effect in supply chains—An overestimated problem?," *Int. J. Prod. Econ.*, vol. 118, no. 1, pp. 311–322, 2009.
[30] Y.-L. Wang, "Logistics supply chain coordination based on multi-agent system," in *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference on*, 2010, pp. 524–528.
[31] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Comput. Surv. CSUR*, vol. 26, no. 1, p. 91, 1994.
[32] ARSHINDER, A., Y DESHMUKH S.G., "Supply chain coordination: Perspectives, empirical studies and research directions," *Journal of production economics*, vol. 115, pp. 316–335, 2008.
[33] J. A. Narus and J. C. Anderson, "Rethinking Distribution: Adaptive Channels," *Harv. Bus. Rev.*, vol. 74, pp. 112–120+, 1996.
[34] D. M. Lambert, M. A. Emmelhainz, and J. T. Gardner, "Building successful logistics partnerships," *J. Bus. Logist.*, vol. 20, no. 1, pp. 165–182, 1999.
[35] BALLOU R.H., GILBERT S.M., MUKHERJEE A., "New managerial challenges from supply chain opportunities," *Industrial Marketing Management*, vol. 29, no. 1, pp. pp. 7–18, 2000.
[36] H. L. Lee, "Creating value through supply chain integration," *Supply Chain Manag. Rev.*, vol. 4, no. 4, pp. 30–36, 2000.
[37] SIMATUPANG, T., SRIDHARAN, R., "The Collaborative Supply Chain," *The International Journal of Logistics Management*, 2002.

[38] P. K. Bagchi and T. Skjoett-Larsen, “Integration of information technology and organizations in a supply chain,” *Int. J. Logist. Manag.*, vol. 14, no. 1, pp. 89–108, 2003.
[39] H. A. Granja Florez, “Comunidad virtual agraria, como un sistema de negocio e intercambio libre de información,” Universidad de Belgrano, Buenos Aires, 2011.
[40] RGX, “Internet y las nuevas Tecnologías como herramientas para las PyMes exportadoras.” 2009.
[41] S. Cassidy, “Internet y la economía real.” 2007.
[42] P. M. Morales, “¿Por qué usar Internet en su empresa?” 2007.
[43] M. Florez Calderón, “La Agrónica Informática, electrónica, telecomunicaciones al servicio de los recursos naturales,” *Ingeniería e Investigación; núm. 23 (1991)*, 2011.
[44] S. E. Pomares Hernández, “Computación Ubicua; un gran desafío,” *Grandes Retos de Investigación Científica y Tecnológica en Tecnologías de Información y Comunicaciones en México*. 21-May-2009.
[45] J. A. Enríquez Hernández, K. L. Silva Martinez, A. Jahuey Muñiz, and I. de J. Robles Cruz, “Sistema de Control Agrícola,” presented at the Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI’2014) ”Excellence in Engineering To Enhance a Country’s Productivity”, Guayaquil, Ecuador, 2014.
[46] M. Strassner and T. Schoch, “Today’s Impact of Ubiquitous Computing on Business Processes,” presented at the Mattern, F. and Naghshineh, M. (Eds.): Pervasice Computing. First Int. Conf. Pervasive Computing, Zurich, Suiza, 2002.
[47] I. Caballero, M. Á. Blanco, and M. Piattini, “Optimización del Proceso de Gestión de Información para la Mejora de la Calidad de la Información,” presented at the Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería del Conocimiento - JIISIC’04, Madrid (España), 2004.
[48] Yan Zhang, Jijun Luo, and Honglin Hu, *WIRELESS MESH NETWORKING Architectures, Protocols and Standards*. Auerbach Publications, 2007.
[49] Venkat Mohan, S. and Kasiviswanath, N., “Routing Protocols for Wireless Mesh Networks,” *International Journal of Scientific & Engineering Research*, vol. 2, no. 8, pp. 42–46.
[50] A. Medina Santos, “Comparativa de los protocolos AODV y OLSR con un emulador de redes Ad-Hoc,” Feb. 2006.
[51] Lucy Coya Rey, Talia Odete Ledesma Quiñones, and Walter Baluja García, “Protocolos de enrutamiento aplicables a redes MANET,” *Revista Telem@tica*, vol. 13, no. 3, pp. 59–74, Sep-2014.
[52] Albert Batiste Troyano, “Protocolos de encaminamiento en redes inalámbricas mesh: un estudio teórico y experimental" (en Español), Tesis de Maestría, Máster Oficial en Software Libre,” Universitat Oberta de Catalunya, 2011.
[53] T. Clausen and P. Jacquet, “RFC 3626 - Optimized Link State Routing Protocol (OLSR) (RFC3626).” [Online]. Available: http://www.faqs.org/rfcs/rfc3626.html. [Accessed: 01-Jun-2015].
[54] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized link state routing protocol for ad hoc networks,” in *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, 2001, pp. 62–68.
[55] “OGM - batman-adv - Open Mesh.” [Online]. Available: http://www.open-mesh.org/projects/batman-adv/wiki/OGM. [Accessed: 04-Jun-2015].

[56] G. Parissidis, M. Karaliopoulos, R. Baumann, T. Spyropoulos, and B. Plattner, “Routing metrics for wireless mesh networks,” in *Guide to Wireless Mesh Networks*, Springer, 2009, pp. 199–230.
[57] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, “Preemptive routing in ad hoc networks,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 43–52.
[58] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi, “Signal stability-based adaptive routing (SSA) for ad hoc mobile networks,” *Pers. Commun. IEEE*, vol. 4, no. 1, pp. 36–45, 1997.
[59] R. J. Punnoose, P. V. Nikitin, J. Broch, and D. D. Stancil, “Optimizing wireless network protocols using real-time predictive propagation modeling,” in *Radio and Wireless Conference, 1999. RAWCON 99. 1999 IEEE*, 1999, pp. 39–44.
[60] R. Draves, J. Padhye, and B. Zill, “Comparison of Routing Metrics for Static Multi-hop Wireless Networks,” *SIGCOMM Comput Commun Rev*, vol. 34, no. 4, pp. 133–144, Aug. 2004.
[61] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” *Wirel. Netw.*, vol. 11, no. 4, pp. 419–434, 2005.
[62] R. Draves, J. Padhye, and B. Zill, “Routing in Multi-radio, Multi-hop Wireless Mesh Networks,” in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2004, pp. 114–128.
[63] B. Awerbuch, D. Holmer, and H. Rubens, “The Medium Time Metric: High Throughput Route Selection in Multi-rate Ad Hoc Wireless Networks,” *Mob Netw Appl*, vol. 11, no. 2, pp. 253–266, Apr. 2006.
[64] P. Kyasanur and N. H. Vaidya, “Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks,” *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 10, no. 1, pp. 31–43, 2006.
[65] C. E. Koksal and H. Balakrishnan, “Quality-Aware Routing Metrics for Time-Varying Wireless Mesh Networks,” *Sel. Areas Commun. IEEE J. On*, vol. 24, no. 11, pp. 1984–1994, Nov. 2006.
[66] Y. Yang, J. Wang, and R. Kravets, “Designing routing metrics for mesh networks,” in *In WiMesh*, 2005.
[67] G. Parissidis, M. Karaliopoulos, M. May, T. Spyropoulos, and B. Plattner, “Interference in wireless multihop networks: A model and its experimental evaluation,” in *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, 2008, pp. 1–12.
[68] C.-K. Toh, “Associativity-based routing for ad hoc mobile networks,” *Wirel. Pers. Commun.*, vol. 4, no. 2, pp. 103–139, 1997.
[69] K. Paul, S. Bandyopadhyay, A. Mukherjee, and D. Saha, “Communication-aware mobile hosts in ad-hoc wireless network,” in *Personal Wireless Communication, 1999 IEEE International Conference on*, 1999, pp. 83–87.
[70] K. Scott and N. Bambos, “Routing and channel assignment for low power transmission in PCS,” in *Universal Personal Communications, 1996. Record., 1996 5th IEEE International Conference on*, 1996, vol. 2, pp. 498–502 vol.2.
[71] J. Sheu, C. Hu, and C. Chao, *The Handbook of Ad Hoc Wireless Networks, Chapter Energy-Conserving Grid Routing Protocol in Mobile Ad Hoc Networks*. RCR Press LLC, 2003.
[72] C.-K. Toh, “Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks,” *Commun. Mag. IEEE*, vol. 39, no. 6, pp. 138–147, 2001.
[73] J.-H. Chang and L. Tassiulas, “Maximum lifetime routing in wireless sensor networks,” *IEEEACM Trans. Netw. TON*, vol. 12, no. 4, pp. 609–619, 2004.

[74] A. Michail and A. Ephremides, “Energy-efficient routing for connection-oriented traffic in wireless ad-hoc networks,” *Mob. Netw. Appl.*, vol. 8, no. 5, pp. 517–533, 2003.
[75] S. Singh, M. Woo, and C. S. Raghavendra, “Power-aware routing in mobile ad hoc networks,” in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998, pp. 181–190.
[76] IEEE, “Draft amendment: ESS mesh networking,” *IEEE P802.11s Draft 1.00*. Nov-2006.
[77] “Wireshark · Go Deep.” [Online]. Available: https://www.wireshark.org/. [Accessed: 24-Jun-2015].
[78] The Avahi Team, “Avahi,” 2014-2005. [Online]. Available: http://www.avahi.org/. [Accessed: 03-Jun-2015].
[79] “Iperf - The TCP/UDP Bandwidth Measurement Tool.” [Online]. Available: https://iperf.fr/. [Accessed: 23-Jun-2015].
[80] Chris Liechti, “pyserial 2.7 : Python Package Index.” [Online]. Available: https://pypi.python.org/pypi/pyserial. [Accessed: 01-Jun-2015].