Un nuevo marco para redes de sensores inalámbricos definidos por software

A Novel Framework for Software Defined Wireless Sensor Networks

Autor 1: Edward Villota Taramuel Autor 2: Andrés Charfuelan Guancha
Universidad Tecnología de Pereira, Pereira (Risaralda), Colombia

Resumen:
Abstract— Se presenta un novedoso marco para redes inalámbricas de sensores definidas por software (SDWSN) que se basa en conceptos y capacidades de redes definidas por software (SDN) para mejorar el control, la gestión y la seguridad, a la vez que reduce la complejidad del dispositivo. Estas complejidades inherentes plantean desafíos significativos para el avance de la detección ubicua y el acceso a los datos sensoriales a través del modelo Sensing-as-a-Service (S2aaS). Por lo tanto, es ventajoso utilizar SDN para desacoplar el control y los planos de reenvío de datos e incorporar un mayor control sobre la virtualización dinámica y los enfoques para mejorar la calidad de la experiencia. Los algoritmos mejorados se pueden aplicar al conocimiento mejorado de las condiciones de la red que se puede lograr cuando se emplea SDN. Ejecutamos simulaciones basadas en el modelo de flujo del sensor y proporcionamos un análisis completo del marco de SDWSN, la arquitectura y las limitaciones de implementación.

Key Word:
Software Defined Networking, Wireless Sensor Networks, Controller, Network Architecture, OpenFlow.

I. INTRODUCCIÓN

El desarrollo de dispositivos sensores como los dispositivos micro electromecánicos (MEM) utilizados para la recopilación y difusión de información ha llevado a la introducción de redes inalámbricas de sensores (WSN) que son un habilitador clave para sistemas inteligentes como sistemas de automatización industrial, redes inteligentes, hogares inteligentes y sistemas de monitoreo de la salud. La flexibilidad ofrecida por WSN impulsa diversas aplicaciones y despliegues específicos de proveedores que brindan una utilización eficiente de los recursos y soluciones innovadoras a problemas del mundo real. Las WSN centradas en aplicaciones generalmente requieren la configuración y el control de los dispositivos de red para implementar y administrar los protocolos de transmisión, la seguridad y los mecanismos de acceso. Además, la gran implementación de WSN enfrenta desafíos debido a la limitación de recursos, la descentralización y los protocolos de enrutamiento ad-hoc para reenviar datos entre nodos [1]. Finalmente, todos estos problemas contribuyen a una gran carga de cómputo, pérdida de paquetes, retrasos en la transmisión y obstrucción a la optimización del rendimiento de la red. Para abordar estos problemas críticos, la opción de utilizar SDN y separar los planos de control y datos proporciona un enfoque que se basa en el concepto de WSN de tener dispositivos inalámbricos de baja potencia, simples pero flexibles desplegados. SDN proporciona un mecanismo de control centralizado que mejora la visibilidad, la seguridad y la flexibilidad cuando se aplica la administración y la prioridad de la aplicación dentro de una WSN.

Fig. 1 muestra una arquitectura conceptual del SDWSN. Un controlador centralizado basado en software es responsable del plano de control de los dispositivos en red de una manera independiente del proveedor. Por lo tanto, no es un requisito para los dispositivos físicos de red mantener el estado y tener las capacidades computacionales y de almacenamiento necesarias para el funcionamiento del plano de control independiente. La actividad clave para los dispositivos WSN es detectar y reenviar datos a la ubicación de almacenamiento o análisis de datos apropiados dentro de la red. Es posible facilitar una variedad de resultados, incluido el protocolo, la seguridad y las actualizaciones de la aplicación sin reemplazar un dispositivo. Además, se pueden incorporar múltiples aplicaciones con diferentes patrones de tráfico, prioridades de procesamiento y tiempos de caducidad de los datos. Por lo tanto, una de las ventajas básicas de la implementación de SDN es el desarrollo de aplicaciones de red de terceros [2]. La adición de un aprovisionamiento inteligente y un plano de orquestación permiten a los operadores de red desplegar aplicaciones de forma abstracta y luego controlar la operación, seguridad y prioridad de la aplicación sin aumentar la complejidad de la red que optimizará la utilización de los recursos de red [3].

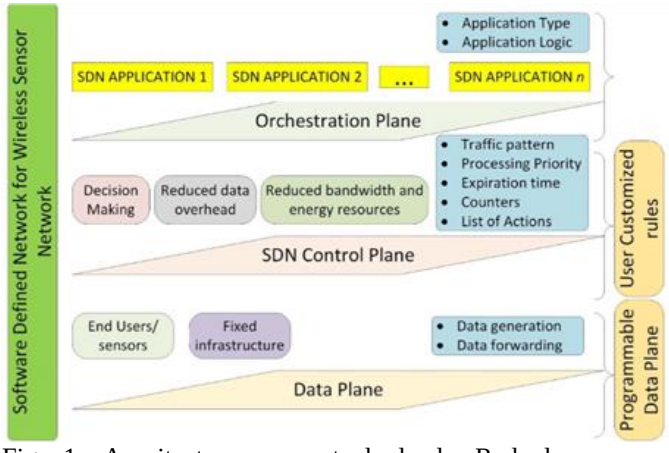


Fig. 1. Arquitectura conceptual de la Red de sensores inalámbricos definidos por software (SDWSN)

Fecha de Recepción: 26 Septiembre de 2016
Fecha de Aceptación:

La transformación de WSN existente en SDWSN es un desafío porque aún no se ha definido un marco adecuado. Además, es necesario reemplazar los dispositivos de red inalámbrica existentes con dispositivos habilitados para SDN. Los autores en [4] resumen los desafíos técnicos y los requisitos asociados con la implementación del plano de control SDN y el protocolo OpenFlow en una WSN. También han resaltado el trabajo futuro que debe llevarse a cabo para que OpenFlow o uno de los otros protocolos del plano de control en desarrollo se incorpore en un WSN definido por software. OpenFlow es un proyecto de código abierto bien recibido que permite la integración de una amplia gama de controladores, dispositivos de red y sistemas corporativos. En [5] se proporciona un modelo de red mallada inalámbrica basada en OpenFlow y resultados iniciales de análisis sobre cómo puede funcionar el modelo; sin embargo, quedan cuestiones fundamentales por resolver, como la investigación sobre escalabilidad de red, rendimiento de servicios de red, configuraciones de red virtualizadas, tunelización esquemas y formatos de encapsulamiento.

En este documento, proponemos un marco SDWSN que ofrece una arquitectura operativa robusta. Desarrollamos el modelo de simulación en Castalia [6] para mostrar la viabilidad del diseño propuesto. Los resultados de la simulación destacan el rendimiento del marco SDWSN. El resto del documento se describe de la siguiente manera: la Sección II proporciona la arquitectura general de la red y analiza brevemente el modelo del sistema. La Sección III describe la implementación de SDWSN. La Sección IV proporciona una descripción del modelo de simulación y analiza los resultados para validar la viabilidad del marco propuesto. Finalmente, la Sección V proporciona la conclusión y plantea temas para futuras investigaciones.

II. SYSTEM MODEL (MODELO DE SISTEMA)

Consideramos un sistema de almacenamiento y agregación de información distribuida, donde definimos cada evento por atributos multidimensionales y los categorizamos en dos aplicaciones. Uno utiliza el esquema de almacenamiento centrado en los datos (DCS) para facilitar el almacenamiento en la red [7], mientras que luego se define en función de los valores y las definiciones de los atributos distribuidos. El modelo de sistema conceptual incluye el desarrollo de una arquitectura de red, un modelo de difusión de paquetes y la consideración de las propiedades de capa Física (PHY) y Control de acceso Medio (MAC).

A. Network architecture (Arquitectura de red)

Fig. 2 muestra la arquitectura de la red basada en clúster superpuesta del marco propuesto. Los dispositivos de red se implementan en tres capas diferentes, que se describen como acceso, plano de datos y capa de control. Los dispositivos en el acceso, el plano de datos y las capas de control se denominan nodos finales (EN), conmutadores SDN (SDSW) y controlador SDN, respectivamente. Como se muestra en la Fig. 2, cada grupo comprende de ENs y un SDSW, que actúa como un cabezal de clúster y puerta de enlace. Cada SDSW corresponde con el controlador SDN más cercano para recuperar información de control. La información de control se guía a través de un conjunto de reglas llamadas comandos de flujo proporcionados por el controlador SDN.

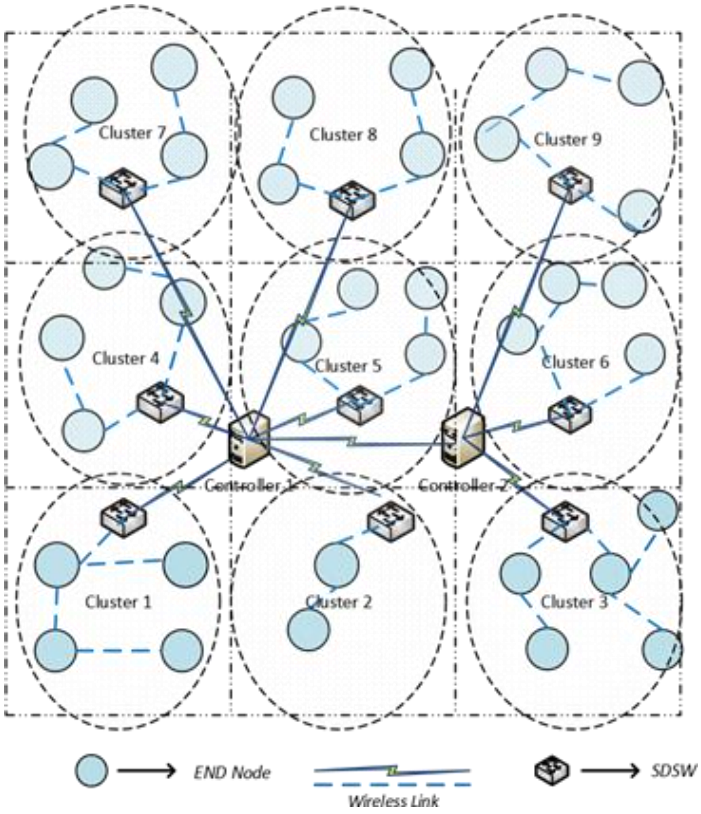


Fig. 2. Network Architecture of SDWSN (Arquitectura de red de SDWSN)

B. Packet Dissemination Model (Modelo de diseminación de paquetes)

Para optimizar la latencia, clasificamos las aplicaciones en sensible a retardos y tolerante al retardo. Los flujos proactivos y reactivos se usan para controlar la diseminación de los paquetes de aplicaciones sensibles a retardos y tolerancias de retardo, respectivamente. En el flujo proactivo, el controlador SDN empuja el flujo a todos los SDSW inmediatamente después de implementar una aplicación sensible a la demora. Esto reduce el tiempo requerido para resolver un paquete de aplicación sensible al retardo al evitar la comunicación del canal de control entre SDSW y el controlador SDN. Por otro lado, los flujos reactivos se unen solo a los SDSW correspondientes según la demanda. Cabe señalar que los flujos

proactivos son permanentes en contraste con los flujos reactivos.

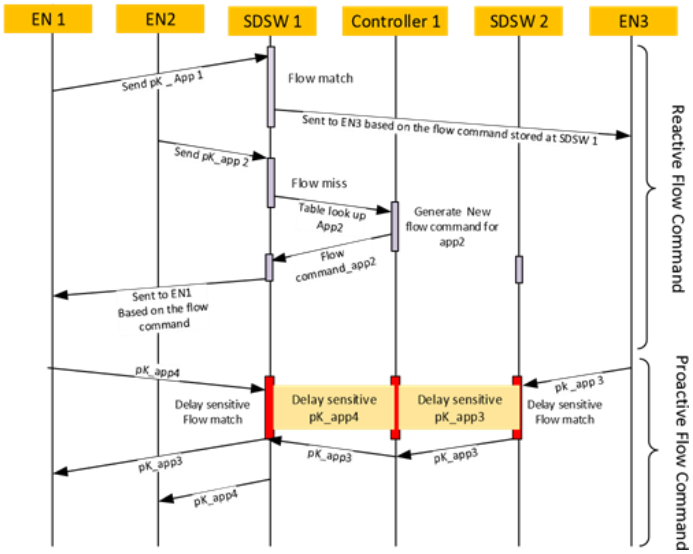


Fig. 3. Packet dissemination model for SDWSN (Modelo de difusión de paquetes para SDWSN)

Fig. 3 muestra el intercambio de mensajes entre EN, SDSW y controlador SDN basado en un comando de flujo reactivo o proactivo. Por ejemplo, pK_App1 y pK_App2 son dos paquetes de aplicaciones genéricas y se envían a los SDSW correspondientes. Al recibir estos paquetes de aplicación, al principio, los SDSW los clasifican como paquetes tolerantes al retardo y realizan una búsqueda en la tabla de flujo. Si hay una "coincidencia de flujo", el SDSW correspondiente inicia la acción en función del comando de flujo y reenvía el paquete en consecuencia. Por el contrario, si hay un "flujo perdido", el SDSW inicia una consulta de flujo (paquete-in) al controlador SDN. Tras recibir con éxito un comando de flujo (paquete-out) desde el controlador correspondiente, el SDSW toma la acción especificada por el flujo y agrega el flujo en la tabla de flujo. Sin embargo, en el caso de un paquete de aplicación sensible a la demora tal como pK_APP3 y pK_APP4, el controlador SDN transmite el comando de flujo proactivo a todos los SDSW asociados inmediatamente después de desplegar la aplicación sensible al retardo. Los comandos de flujo proactivo se almacenan permanentemente en todos los SDSW hasta que las aplicaciones correspondientes estén activas en la red. Tras la recepción exitosa de estos paquetes, los SDSW encuentran una coincidencia de flujo definida y, por lo tanto, toman la acción inmediata.

III. IMPLEMENTATION DETAILS (DETALLES DE IMPLEMENTACION)

Castalia [6], un simulador de fuente abierta para WSN y redes de área del cuerpo (BAN), se amplía para implementar el marco. En el proceso de realización del marco, implementamos un protocolo llamado Aplicación SDWSN (SDWSNAPP), que se agrega a la capa de aplicación. Dado que la arquitectura presentada en la figura 2 es una arquitectura de clúster, por lo tanto, utilizamos una versión modificada del protocolo de enrutamiento basado en la distancia sectorial (SBD) realizado en el controlador. Teniendo en cuenta la restricción de espacio, excluimos la ilustración de SBD en este documento. En las siguientes subsecciones, el marco se ilustra utilizando una conectividad de extremo a extremo en toda la capa de aplicación.

A. Application Layer (Capa de aplicación)

En una conexión de capa de aplicación de extremo a extremo, cada EN recopila los valores detectados de los dispositivos de sensor correspondientes y envía valores agregados al SDSW asociado. Un SDSW clasifica el tráfico recibido por las aplicaciones predefinidas para etiquetar cada paquete con una ID de aplicación particular. Después de la clasificación, el SDSW realiza una búsqueda en la tabla de flujo. Si se encuentra una entrada coincidente, se lleva a cabo la acción asociada con la entrada de flujo. Si no se detecta ninguna coincidencia en la tabla de flujo, el resultado depende de la configuración de SDSW: el paquete se puede reenviar al controlador utilizando OpenFlow, se puede descartar o puede continuar hasta la siguiente tabla de flujo. Las subsecciones siguientes ilustran la comunicación y las funcionalidades de la capa de aplicación de extremo a extremo según tres capas de la arquitectura presentada en la Sección II-A.

1) Access Layer and Data Plane (Capa de acceso y plano de datos)

Las EN ponen los datos agregados recogidos de los sensores en el paquete SDWSN_Traffic_General (Figura 4) y unifican el paquete al SDSW asociado en modo de salto único. A la llegada de un paquete SDWSN_Traffic_General, el SDSW clasifica el paquete en la aplicación correspondiente mediante el uso del 'Módulo de clasificación de aplicaciones' (Algoritmo 1). En esta etapa, la identificación de la aplicación y el NOMBRE de la APLICACIÓN se rellenan con el paquete entrante (Figura 5). Después de clasificar y etiquetar cada paquete con una ID y nombre de aplicación particular, SDSW realiza la búsqueda de tabla en la tabla de flujo. Cuando se encuentra un flujo, la acción asociada con este flujo o aplicación se ejecuta para este paquete. Luego, el paquete se etiqueta como resuelto y se empuja hacia abajo a la capa de red (Fig. 6).

Algoritmo 1 App_Classification_Module () en la capa de aplicación de SDSW. El módulo se activa inmediatamente después de recibir el paquete de tráfico SDN (SDWSN_Traffic_General).

1: **input**: *SDWSN_Traffic_General*
2: **extract** all attributes value from *SDWSN_Traffic_General* and assign to *A1*, *A2*, ……, *An*
3: //Duplicating each packet to classify into DCSApp
4: *DCSAppPacket* ¬*SDWSN_Traffic_General*
5: *appId* ¬1 and *appName*=DCS
6: **set** *appId* and *appName* to *DCSAppPacket*
7: // Classify based on attribute values and given parameters
8: **classify** *SDWSN_Traffic_General*
9: **find** *appId* and *appName*
10: // Duplicate and add application ID and name
11: *AppPacket* ¬*SDWSN_Traffic_General*
12: **set** *appId*→ *AppPacket*
13: *appName*→*AppPacket*
14: //Store in *intermediateBuffer* to be processed by **Flow_Table_Lookup** model
15: **push** *DCSAppPacket* and *SDWSNAppPacket* into *intermediateBuffer*

Packet Type=1	SRC	DST	Aggregated Data				
	EN ID	SDSW ID	*A1*	*A2*	*A3*	...	*An*

Fig. 4. *SDWSN*_Traffic_General Packet (Paquete SDWSN_Traffic_General)

Packet Type=2	SRC	DST	APP ID	APP NAME	Aggregated Data				
	EN ID	SDSW ID			A_1	A_2	A_3	...	A_n

Fig. 5. SDWSN_App/DCS_App Packet

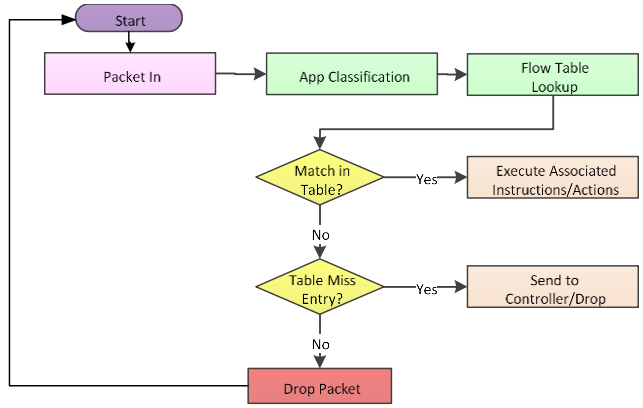


Fig. 6. Packet Flow through SDSW (Paquete de flujo a través de SDSW)

En el caso de una falla de flujo, el SDSW almacena el paquete en Buffer no resuelto y crea un paquete de control (SDWSN_Cntrl_Pkt, ver la Fig. 7) al replicar solo el preámbulo del paquete SDWSN_App. Luego reenvía el SDWSN_Cntrl_Pkt al controlador. El controlador responde con SDWSN_Cntrl_Pkt_Actions (Fig. 8) al SDSW solicitante. En SDWSN_Cntrl_Pkt_Actions, el controlador incluye la acción requerida para el tráfico respectivo. Después de recibir el paquete SDWSN_Cntrl_Pkt_Actions, el SDSW crea un nuevo flujo en la tabla de flujo y saca el paquete de aplicación correspondiente de *unresolvedBuffer* no resuelto y lo empuja al *intermediateBuffer*.

Packet Type=3	Preamble from SDWSN_Traffic_General				SRC	DST
	EN ID	SDSW ID	APP ID	APP NAME	SDSW ID	CNTRL ID

Fig. 7. Control Packet (SDWSN_Cntrl_Pkt)

Packet Type=4	Preamble from SDWSN_Traffic_General				SRC	DST
	EN ID	SDSW ID	APP ID	APP NAME	CNTRL I)	SDSW ID
Actions						
Forward/ Drop	Unicast/ Broadcast	Destination ID				

Fig. 8. Control Packet with Action Instruction (SDWSN_Cntrl_Pkt_Actions)

2) *Control Plane (Plano de control)*

Un administrador de red configura, modifica e instala una nueva aplicación de la red en un controlador. El controlador difunde inmediatamente la actualización a otros controladores y SDSW asociados. Los controladores de destinatarios insertan esta actualización en sus SDSW asociados. La segunda función central de un controlador es proporcionar los mensajes de control de flujo a los SDSW. En el caso de un flujo proactivo, el controlador empuja el flujo (SDWSN_Proactive_Cntrl_Pkt_Actions) (Fig. 9) a los SDSW inmediatamente después de desplegar una nueva aplicación, mientras que en el caso de un flujo reactivo, el controlador responde con SDWSN_Cntrl_Pkt_Actions solo después de recibir un SDWSN_Cntrl_Pkt .

Packet Type=5	SRC	DST	APP ID	APP NAME
	CNTRL ID	Broadcast = -1		
actions				
Forward/ Drop	Unicast/ Broadcast	Destination ID		

Fig. 9. Pro-active Flow Control Packet (SDWSN_Proactive_Cntrl_Pkt_Actions)

IV. Simulation Results

Para validar el marco presentado, construimos un modelo de simulación en Castalia. La Tabla 1 muestra los parámetros y configuraciones utilizados en las simulaciones. Cada experimento se ejecuta con 100 repeticiones para encontrar el promedio, la mediana y el intervalo de confianza del 95%. Modelamos la red en un campo rectangular con distintas configuraciones de parámetros y evaluamos el rendimiento del marco SDWSN en términos de tasa de éxito y latencia frente a un número variable de aplicaciones. La tasa de éxito y la latencia se definen como:

Tasa de éxito. La relación entre el número total de paquetes de aplicaciones resueltos y la cantidad total de paquetes de aplicaciones generados.

SuccessRate(α) = *Total # of packets resolved* / *Total # of packets generated* *(11)*

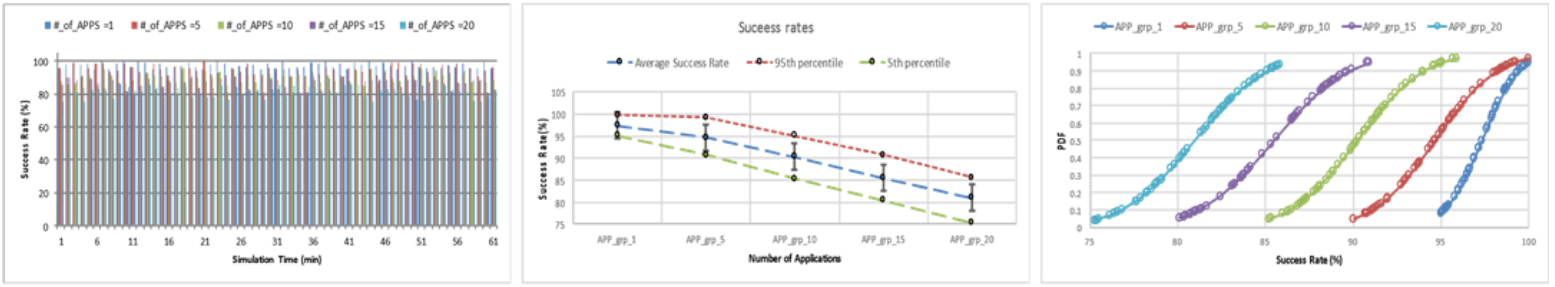
Estado latente. El tiempo entre la generación y resolución de un paquete de aplicación.

TABLE I. SIMULATION PARAMETERS

Parameter	Setting
Field Size	150x150 m^2
Number of Nodes (N)	500 (22500 m^2)
EN Density (f_m), SDSW Density (f_{SH})	1 node / 56.25 m^2, 1 node / 225 m^2
Radio Range (EN, SDSW, Controller)	~8 m, ~20 m, ~20 m
Transmission Power	0 dBm (*SDSW*), -10 dBm (EN)
Data Rate, Modulation Type, Bits Per Symbol, Bandwidth, Noise Bandwidth, Noise Floor, Sensitivity	250 Kbps, PSK, 4, 20 MHz, 194 MHz, -100 dBm, -95 dBm
pathLossExponent	2.4
Initial Average Path Loss ($PL(d_0)$)	55
Reference Distance (d_0)	1.0 m
Gaussian Zero-Mean Random Variable (X_α)	4.0

A. *Success Rate (Tasa de éxito)*

En este experimento, ejecutamos la simulación durante 60 minutos en un campo rectangular de 150 x 150 m2. La red estaba compuesta por 400 ENs, 100 SDSWs y 20 ontrollers SDN. Medimos la tasa de éxito frente a diferentes grupos de aplicaciones. El grupo más pequeño contiene una aplicación, mientras que el grupo más grande contiene 20. Los resultados se muestran en la Figura 10 (a) - (c). La Figura 10 (a) muestra la tasa de éxito de diferentes grupos de aplicaciones contra el tiempo de simulación. Para proporcionar una representación clara de los resultados, la tasa de éxito promedio de cada uno de los grupos de aplicaciones se muestra en la Figura 14 (b) con el percentil 95 y 5.



(a) (b) (c)

Fig. 10. a) Success rate against simulation time for different groups of applications b) Average, 95th & 5th percentile of success rate against the application group c) CDF

Aunque el número de aplicaciones aumenta gradualmente al aumentar el tamaño del grupo, mantenemos la tasa de generación de paquetes como constante para cada grupo. Por lo tanto, cuando el número de aplicaciones es alto, la tasa de tráfico para una aplicación particular de las EN a SDSW se reduce mientras que la variación del tráfico aumenta. Con el tiempo, la probabilidad de recibir paquetes de aplicaciones del mismo tipo disminuye, mientras que aumenta la posibilidad de recibir paquetes de aplicaciones de una gama más amplia de tipos de aplicaciones. En última instancia, aumentan las solicitudes de flujo y las respuestas de flujo entre los controladores SDSW y SDN. Esto aumenta la carga de tráfico general en el canal de control y produce un punto de acceso alrededor del controlador, lo que resulta en la caída de más paquetes. Esto se visualiza adicionalmente en la figura 10 (c), donde podemos ver que la tasa de éxito varía entre 95% y 100% para el grupo uno (una aplicación), mientras que para el grupo cinco (20 aplicaciones) varía entre 75% y 85%.

B. Latency (Estado latente)

En esta sección, llevamos a cabo los experimentos con configuraciones de simulación similares utilizadas para medir la tasa de éxito en la Sección IV-A. Evaluamos la latencia frente a distintos grupos de aplicaciones y presentamos los resultados en la figura 11 (a) - (c). La Fig. 11 (a) representa la latencia promedio de los grupos de aplicaciones a lo largo del tiempo. Para una representación explícita, la latencia promedio de cada grupo se muestra en la figura 11 (b) con el percentil 95 y 5. De la Fig. 11 (b), se puede ver que la latencia promedio aumenta gradualmente a medida que aumenta el número de aplicaciones. Sucede debido a un mayor número de paquetes de control entre los controladores SDSW y SDN para la creciente variación de aplicaciones. De la Fig. 11 (c), podemos ver que el sistema resuelve paquetes al 100% dentro de los 24 ms cuando una aplicación (grupo uno) se está ejecutando en la red.

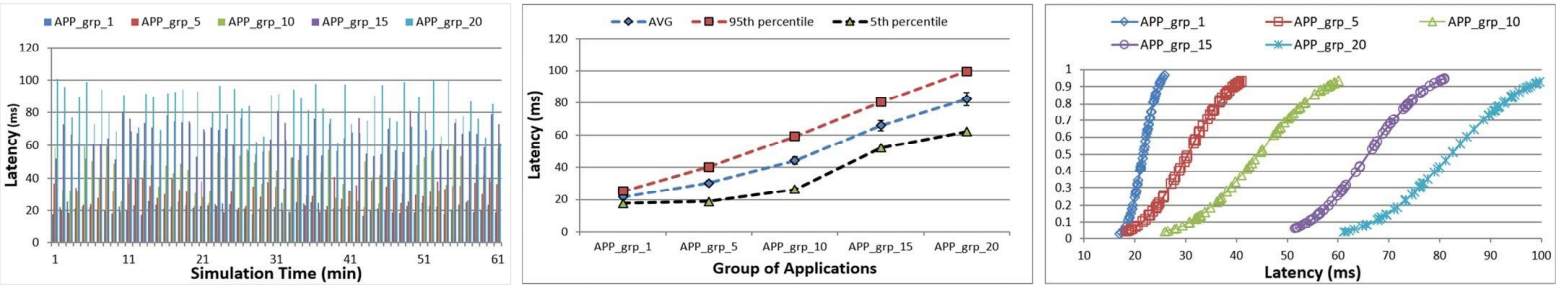


Fig. 11. a) Latency against simulation time for different groups of applications b) Average, 95th & 5th percentile of latency against group of applications c) CDF

Mientras que para el grupo cinco (20 aplicaciones), la latencia varía de 60 ms a 100 ms para resolver todos los paquetes

V. CONCLUSION & FUTURE WORK

En este documento, presentamos un marco para SDWSN e ilustramos los detalles y las especificaciones de la implementación de la capa de aplicación. Un modelo de simulación desarrollado en Castalia se utiliza para evaluar el rendimiento del modelo de red y el marco desarrollado que proporciona una plataforma excelente para que los investigadores interesados investiguen SDWSN con un número variable de aplicaciones presentadas a los controladores sin intervención física a los dispositivos de red de capa física. Para mantener la demostración simple, limitamos nuestra evaluación dentro de las dos aplicaciones mencionadas en la Sección I. Este trabajo se enfoca en la comunicación hacia el sur e ignora el hecho de la comunicación hacia el norte. Como se ilustra en las Figuras 10 y 11, la tasa de éxito y la latencia del flujo de datos se ven principalmente afectados por los flujos de control debido a la naturaleza de gestión en banda de los WSN. Por lo tanto, o bien debemos esperar utilizar dos frecuencias de canales de radio por separado [15] o se debe desarrollar un modelo de optimización para reducir el impacto de los flujos de control sobre los flujos de datos. Finalmente, hemos simulado el marco propuesto utilizando una topología de red basada en clúster rectangular donde el número de EN, SDSW y controladores implementados se basan en una suposición heurística relacionada con WSN. La elección del número de controlador y la ubicación es estática en esta etapa que necesita ser investigada para construir una relación entre las EN, los SDSW y los controladores.

En futuras investigaciones, también sería apropiado desarrollar un modelo validado para optimizar el número de aplicaciones y controladores que se ejecutan en la red para lograr el mayor rendimiento de la red. El marco también debe probarse variando la topología y el tamaño de la red. Tampoco hemos considerado ninguna comunicación este-oeste entre el controlador, lo que plantearía la cuestión de la sincronización entre los controladores. Los trabajos futuros también incluyen la implementación del banco de pruebas y el análisis del rendimiento en un entorno del mundo real.

VI. ACKNOWLEDGEMENT

Este trabajo de investigación fue apoyado por Zayed University Research Cluster Award R16086.

VII. REFERENCES

[1] A. De Gante, M. Aslan and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," *2014 27th Biennial Symposium on Communications (QBSC)*, Kingston, ON, 2014, pp. 71-75

[2] S. Scott-Hayward, C. Kane and S. Sezer, "OperationCheckpoint: SDN Application Control," *2014 IEEE 22nd International Conference on Network Protocols*, Raleigh, NC, 2014, pp. 618-623.

[3] Y. Wang and I. Matta, "SDN Management Layer: Design Requirements and Future Direction," *2014 IEEE 22nd International Conference on Network Protocols*, Raleigh, NC, 2014, pp. 555-562.

[4] T. Luo, H. P. Tan and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," in *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896-1899, November 2012.

[5] P. Dely, A. Kassler and N. Bayer, "OpenFlow for Wireless Mesh Networks," *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, Maui, HI, 2011, pp. 1-6

[6] A. Boulis. Castalia: A simulator for wireless sensor networks and body area networks: Version 3.2: User's manual [Online]. Available: http://castalia.npc.nicta.com.au/

[7] K. Ahmed and M. A. Gregory, "Decentralized coding algorithm in Data Centric Storage for wireless sensor networks," *2013 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, Christchurch, 2013, pp. 106-111

[8] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), Santa Clara, CA, 2004, pp. 517-526.

[9] T. V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," *1st international conference on Embedded networked sensor systems*, Los Angeles, California, USA, 200, pp. 171-180.

[10] Wei Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, 2002, pp. 1567-1576

[11] K. Ahmed and M. Gregory, "Distributed Efficient Similarity Search Mechanism in Wireless Sensor Networks," Sensors, vol. 15, p. 5474, Mar 2015.

[12] Y. C. Chung, I. F. Su and C. Lee, "Supporting Multi-Dimensional Range Query for Sensor Networks," *27th International Conference on Distributed Computing Systems (ICDCS '07)*, Toronto, ON, 2007, pp. 35-35.

[13] D. Ganesan, D. Estrin, and J. Heidemann, "DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks?," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 143-148, Jan 2003.

[14] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao and D. Estrin, "Networking issues in wireless sensor networks", *Journal of Parallel and Distributed Computing*, vol. 64, no. 7, pp. 799-814, 2004.

[15] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan and R. Wichman, "In-Band Full-Duplex Wireless: Challenges and Opportunities," in IEEE Journal on Selected Areas in Communications, vol. 32, no. 9, pp. 1637-1652, Sept. 2014.