Treball de recerca

Màster oficial en Enginyeria Informàtica i de la Seguretat

Overlay Routing Protocols for Mobile Ad Hoc Networks

Gerard París Aixalà

Director: Dr. Pedro García López

Grup de recerca del director: AST

5 de setembre de 2008

Taula de continguts

Abstract

Mobile ad hoc networks (MANET) are wireless network which do not rely on any fixed infrastructure. The nodes in a MANET may not have all other nodes in radio range, so each node must act both as final node and as router. The interest in MANETs has grown with the increasing popularity of wireless enabled devices (laptops, PDA, mobile phones).

Due to mobility and multi-hop nature of MANETs, each node needs to run a routing protocol in order to be able to forward messages from the node to the destination. These protocols should take into account the node mobility and react to topology changes as well as do not add high overhead to the network. Moreover, depending on application needs, nodes must be able to route both unicast and multicast packets.

Mobile ad hoc networks represent an interesting substrate for many types of applications that do not require a fixed network infrastructure. For instance, spontaneous collaborative applications are of particular interest for settings where many users can interact with close-by participants (conferences, campuses, stadiums, popular events, cities, games). In these settings, it is very important for MANET middleware and applications to be aware of the underlying routing protocols and topology. If the middleware just uses the underlying transport protocol as a black box, it can incur in communication inefficiencies due to the multi-hop nature of the medium.

Existing MANET middleware either ignore the underlying routing protocol or create specific cross-layer solutions that break the strict layering of the network stack. Our proposal, based on peer-to-peer protocols for the Internet, is to move the routing logic to the application layer. Thus, MANET middleware could benefit from routing protocols information. Furthermore, applications based on such a middleware, could be deployed as standalone applications without relying on another routing protocol.

Keywords:
MANET, overlay routing protocols, overlay multicast, middleware

1. Introducció

1.1 El Projecte POPEYE

El Grup de recerca en Arquitectura i Serveis Telemàtics (AST) de la Universitat Rovira i Virgili va participar en el projecte europeu POPEYE, que es va desenvolupar entre els mesos de maig de 2006 i abril de 2008. El projecte POPEYE ("Professional Peer Environment Beyond Edge Computing") és un Specific Targeted Research or Innovation Project (STREP) de 2 anys de duració, format per 7 socis de 4 països diferents:

- Universitat Rovira i Virgili, Tarragona, Espanya
- Universidad de Murcia, Murcia, Espanya
- École Nationale Supérieure des Télécommunications, París, França
- Thales Communications S.A., Colombes, França
- Softeco Sismat SpA, Gènova, Itàlia
- Università degli Studi dell'Aquila, L'Aquila, Itàlia
- OFFIS, Oldenburg, Alemanya

En el context general dels entorns de treball virtualitzats, POPEYE s'adreça als grups peer-to-peer (P2P) mòbils i ad hoc. Aquests grups es caracteritzen perquè no requereixen una infrastructura fixa, i les comunitats virtuals poden emergir-hi espontàniament i compartir dades en un entorn que proporcioni els mecanismes adequats de persistència, sincronització i seguretat.

Fig. 1: Logo del projecte POPEYE

Els objectius del projecte POPEYE són:

- Dissenyar una arquitectura de xarxa overlay integrada que combini l'estabilitat i el rendiment de les xarxes d'infrastructura (quan estiguin disponibles) amb la flexibilitat i el caràcter espontani de les comunicacions mòbils ad hoc.
- Desenvolupar una plataforma de comunicació que proporcioni primitives de comunicació i gestió P2P eficients.

- Desenvolupar serveis d'alt nivell contextuals, segurs i personalitzats, per facilitar el desenvolupament d'aplicacions, permetent la combinació de les preferències d'usuari amb la informació ambiental, com ara el temps, la localització, l'activitat d'usuari i la presencia d'altres peers.

El projecte està dividit en 8 Work Packages (WP), dels quals el WP4 ("P2P over MANET") i el WP5 ("Core Services") són els que la Universitat Rovira i Virgili hi té un paper més rellevant. Com a contribució a aquests Work Packages, la URV ha desenvolupat les tasques següents:

- Desenvolupar una arquitectura per suportar la integració d'una xarxa peer-to-peer mòbil sobre MANETs.
- Proporcionar una estructura topològica d'overlay i una lògica d'encaminament que tingui en compte la proximitat física dels peers a la MANET per tal de reduir la latència en els enllaços de l'overlay i l'ample de banda utilitzat.
- Dissenyar un servei de noms i de gestió de grups sobre l'overlay.
- Proporcionar a les capes de sobre un suport bàsic de serveis col·laboratius.
- Integrar els serveis d'alt nivell en un únic framework.

1.2 Les xarxes ad-hoc mòbils (MANET)

Les xarxes ad-hoc mòbils (MANET, de l'anglès Mobile Ad Hoc Network), tot i que van començar a desenvolupar-se a principis dels anys 70 i 80, han tingut la seva màxima evolució durant els deu últims anys, convertint-se en un dels temes que ha acaparat més recerca en el camp dels sistemes distribuïts. Aquest auge es deu, en gran part, a l'àmplia introducció dels ordinadors portàtils i la implantació de l'estàndard Wi-Fi 802.11 a nivell global des de mitjans dels anys 90.

El nom de MANET defineix clarament dues de les principals característiques d'aquestes xarxes. En primer lloc, el fet de ser xarxes ad-hoc implica que no és necessari disposar d'una infrastructura prèvia, com ara nodes fixos, passarel·les o altres, per tal que els membres de la xarxa es puguin comunicar.

Aquesta manca d'infrastructura converteix les xarxes MANET en candidats ideals a l'hora de crear xarxes en escenaris improvisats o temporals. Per exemple, la coordinació d'equips d'emergència en extinció d'incendis o en operacions de rescat després de desastres naturals, o fins i tot en aplicacions militars en les quals no es disposa d'una infrastructura fixa de xarxa. Aquests camps són els focus actuals d'investigació en xarxes mòbils. En tots aquests escenaris es valora positivament el fet que, en no haver-hi cap entitat central, la permanent entrada i sortida de dispositius de l'abast de la xarxa

no suposi un possible punt de fallada. Per altra banda, el fet de no disposar d'una infrastructura suposa, no obstant, diversos inconvenients.

El fet que tots els nodes de la xarxa puguin observar el tràfic generat per altres nodes, degut a que aquests també realitzen funcions d'encaminament; els recursos limitats de que disposen els dispositius mòbils; i la dinamicitat de la topologia física incrementen la complexitat del desenvolupament de sistemes en aquestes xarxes.

L'altra característica principal d'aquestes xarxes és la mobilitat. Degut a la naturalesa mòbil dels diferents dispositius com les PDAs, els portàtils i els telèfons mòbils, la topologia de xarxa pot canviar contínuament. Això implica possibles particions de la xarxa i fusions de subxarxes. Ambdues situacions són complexes tant per a l'encaminament de paquets com per al desenvolupament de les aplicacions. També cal considerar que a priori, cada dispositiu només es conscient de l'existència dels dispositius que es troben en el seu rang de transmissió. Això implica que si dos nodes que no estan situats en rang volen comunicar-se, els missatges s'han d'encaminar a través de nodes intermedis, que els proporcionen així accés indirecte. Aquesta característica es coneix com escenari *multihop* (multi-salt).

També és important considerar que la majoria d'aquests dispositius presenten certes limitacions en termes de recursos com ara bateria, ample de banda, memòria o capacitat de procés. Això implica que si els nodes han de realitzar accions d'encaminament per enviar missatges a altres nodes, és possible que els dispositius menys potents tinguin problemes per tractar situacions de tràfic dens. D'aquí podem extreure un dels requisits importants de les xarxes MANET: el tràfic generat ha de ser baix, o almenys s'hauria d'evitar la comunicació excessiva entre nodes distants en la xarxa. Per altra banda, en disposar de recursos limitats és bastant probable que certs dispositius suspenguin el seu funcionament temporalment per poder estalviar recursos. Aquest fet afecta la topologia de la xarxa, que es pot veure alterada constantment degut a la connexió i desconnexió d'aquests nodes. D'aquesta manera, la xarxa tindrà enllaços estables, dispositius potents i relativament estàtics, i altres de poc estables com els comentats anteriorment.

La comunicació entre els nodes d'una MANET s'efectua a través d'un medi compartit com és l'aire. Donada la naturalesa broadcast d'aquest medi, qualsevol dispositiu que estigui dins del rang de transmissió de l'altre podrà llegir tots els missatges que el primer enviï. En primer lloc, això representa un problema de seguretat important tant pel que es refereix a l'autenticació, a la confidencialitat i a la integritat de les dades, a més a més d'altres problemes com la denegació de servei. En segon lloc, es poden produir múltiples col·lisions en voler accedir diversos dispositius al medi físic.

Totes aquestes característiques poden no presentar-se simultàniament en tots els escenaris on s'utilitzi una MANET. Depenent de l'escenari concret, seran unes

característiques o unes altres les que posaran més o menys limitacions als protocols d'encaminament utilitzats.

1.3 Protocols d'encaminament overlay per a MANET

Dins dels objectius i la descripció del treball del projecte POPEYE, aquest treball de recerca se centra en la recerca, disseny i desenvolupament de protocols d'encaminament overlay per a xarxes MANET. Com s'ha explicat en la secció anterior, les xarxes MANET no utilitzen cap infrastructura fixa i els nodes que la formen poden estar dispersos en l'espai de manera que qualsevol node de la xarxa no té tots els altres nodes en el rang de comunicació. En conseqüència els nodes d'una MANET actuen tant de nodes finals com d'encaminadors de tràfic. Per tant, tots els nodes han de tenir un protocol d'encaminament per dirigir el tràfic cap als nodes que garanteixin la màxima eficiència en la comunicació. Els protocols d'encaminament per a MANETs han de tenir en compte que els nodes poden estar en moviment i per tant els enllaços siguin poc estables.

En el context del projecte POPEYE, un dels requisits més importants és tenir una arquitectura de xarxa sòlida per tal de suportar els requeriments de les aplicacions col·laboratives d'alt nivell. Per tant, era necessari investigar sobre les diferents alternatives per als protocols d'encaminament MANET. Fruit d'aquesta recerca, van sorgir els protocols que recull aquest treball: OMCAST i jOLSR-OMOLSR. Ambdós són protocols overlay multicast. El primer requereix un protocol d'encaminament unicast a nivell de xarxa, mentre que el segon incorpora el protocol unicast a nivell d'aplicació.

El protocol OMCAST és un protocol d'encaminament multicast a nivell d'aplicació, basat en l'intercanvi d'informació d'estat de l'enllaç i en la computació local dels arbres d'expansió mínima. OMCAST treu profit de la naturalesa broadcast del medi per minimitzar el tràfic de la xarxa MANET.

jOLSR-OMOLSR està format per 2 protocols: jOLSR, una implementació JAVA a nivell d'aplicació del protocol d'encaminament unicast OLSR; i OMOLSR, un protocol d'encaminament multicast basat en OMCAST que es beneficia de la informació de la topologia que li proporciona jOLSR per tal de reduir el tràfic de manteniment de les taules d'encaminament.

2. Treballs relacionats

En aquesta secció volem introduir el tema de l'encaminament a les xarxes MANET i enumerar i descriure les principals característiques dels protocols existents. No es tracta d'una llista exhaustiva, sinó d'un repàs dels protocols més significatius de cada tipus, per tal de justificar les nostres aportacions als protocols que hem dissenyat. Tot i que es poden fer moltes classificacions dels protocols per a MANETs, hem optat per diferenciar-los segons el tipus de missatges que encaminen (unicast/multicast), i per als multicast el nivell de l'arquitectura de xarxa per al qual han estat dissenyats (nivell de xarxa/nivell d'aplicació o overlay). D'aquesta manera podem situar clarament les nostres propostes en l'esmentada classificació.

2.1 Encaminament a les xarxes MANET

L'encaminament dels paquets és un pilar fonamental en el correcte funcionament de qualsevol xarxa. Per això cal estudiar amb cautela quines aproximacions són les més adequades en funció de les característiques de cada xarxa.

A les xarxes fixes, és a dir a les xarxes cablejades, s'han proposat diversos protocols, dels quals les més remarcables són, a grans trets, els de vector distància i estat d'enllaç [1].

Els protocols d'encaminament que segueixen la filosofia de vector distància estan basats en l'algoritme d'encaminament de Bellman-Ford. Aquests protocols basats en vector distància tenen en compte la distància entre nodes, ja sigui el número de salts o una altra mesura, per tal d'obtenir el millor camí a través del qual retransmetre el missatge. Cada node envia periòdicament la informació que posseeix a tots els seus veïns per tal que actualitzin les seves taules de distància. Degut a aquesta generació excessiva de tràfic i a la lenta convergència de la informació sobre la xarxa, aquest algoritme no és adequat per a xarxes molt dinàmiques com podrien ser algunes xarxes MANET.

Per una altra banda, els protocols basats en l'algoritme d'estat de l'enllaç envien de manera periòdica únicament informació sobre els seus "veïns" més propers, evitant així generar massa tràfic extra. A diferència de les aproximacions de vector distància, en aquest cas es pot prendre qualsevol valor com a possible mètrica de distància entre els nodes, aportant una major flexibilitat als protocols que es basen en aquest algoritme. Una altra diferència és que les rutes de mínim cost són calculades utilitzant l'algoritme de Dijkstra. Tot i que els protocols d'estat de l'enllaç no generen tant tràfic, no

solucionen els possibles problemes d'informació inconsistent i de generació de llaços entre nodes.

Com es pot veure, les aproximacions tradicionals de les xarxes "fixes" no són compatibles directament amb els requisits de les xarxes MANET. En conseqüència, s'han dissenyat diverses adaptacions i noves propostes per encaminar els paquets en aquestes xarxes.

Com ja hem dit, els protocols per a MANET es poden classificar de diferents maneres. En aquesta secció seguirem la classificació esmentada segons el tipus de missatge i el nivell de l'arquitectura de xarxa, però volem enumerar les classificacions més usuals que es troben a la literatura:

- Protocols reactius: el càlcul de la ruta s'efectua només quan és necessari. No s'intenta tenir informació d'encaminament actualitzada en tot moment. Aquest protocols no tenen coneixement de la topologia de la xarxa ni de la latència entre nodes en el moment d'enviar el primer paquet a un destinatari.
- Protocols proactius: mantenen la informació sobre les rutes abans que s'utilitzin. Són tècniques més semblants a les utilitzades en les xarxes físiques, però adaptades per al seu ús en xarxes MANET.
- Protocols híbrids o de zona: combinen les dues aproximacions anteriors. Normalment estructuren la xarxa en una jerarquia de diferents nivells.

Fora d'aquestes classificacions es poden trobar protocols que consideren altres paràmetres. Els protocols geogràfics tenen en compte la localització dels nodes mitjançant sistemes de posicionament global (GPS) o els protocols *power-aware* que intenten minimitzar l'energia requerida en les transmissions.

2.2 Protocols unicast

Dels protocols d'encaminament unicast per a MANET, en destacarem els dos que el grup de treball MANET de l'IETF està considerant per a la seva estandardització: DYMO i OLSR. Ambdós són protocols a nivell de xarxa.

2.2.1 DYMO

DYMO (Dynamic MANET On-Demand) [2] és un protocol unicast reactiu, successor del popular AODV (Ad-Hoc On-Demand Distance Vector) i comparteix moltes de les seves característiques. Les rutes es creen sota demanda enviant paquets de control de

petició i resposta. En conseqüència, la informació global de la topologia de xarxa no està mai disponible. Per altra banda, quan un node para d'enviar paquets, no hi ha overhead de tràfic a la xarxa. Per això DYMO s'adapta ràpidament als canvis de topologia de la xarxa i és adequat per comunicacions que es produeixen a intervals de temps.

Com la resta de protocols reactius, DYMO ofereix dues operacions bàsiques: l'operació de crear una ruta i l'operació de mantenir-la.

Per descobrir una nova ruta, el node inicial envia per tota la xarxa un missatge Route Request (RREQ), fins que troba el node destí. En tot aquest procés, els nodes intermedis que reben el paquet RREQ emmagatzemen una ruta des d'ells mateixos fins a l'origen.

Quan el node destí rep un missatge RREQ, respon amb un missatge Route Reply (RREP) que recorre la ruta en sentit contrari fins que arriba al node font. De la mateixa manera que amb els missatges RREQ, tots les nodes intermedis per on passa el missatge RREP emmagatzemen una ruta des d'ells mateixos fins al node destí (el que ha originat el missatge RREP). D'aquesta manera, quan el missatge RREP arriba a l'origen, la ruta queda establerta.

Per mantenir les rutes, si un node intermedi rep un paquet d'una ruta que no coneix (i per tant, hi ha hagut un error ja que aquest node no hauria d'haver rebut aquest missatge) o d'una ruta que sap que ha caigut, llavors s'envia un missatge d'error (RERR). Quan el node origen rep un missatge RERR, sap que ha de començar un altre procés de descobriment de ruta.

Conceptualment, DYMO és un protocol dissenyat per a xarxes ad-hoc mòbils de mida petita, mitjana i gran. També està dissenyat per suportar diferents tipus de tràfic (concentrat, dispers, abundant), però s'ha demostrat que en xarxes grans, DYMO ofereix un bon rendiment si els nodes només es comuniquen amb una certa part de la xarxa, és a dir, en situacions de tràfic dispers.

DYMO es pot aplicar en entorns de memòria reduïda, ja que requereix poca memòria per al manteniment de les rutes. Només es desa informació sobre les rutes que relacionen a orígens i destins actius, a diferència d'altres protocols que han de mantenir la informació d'encaminament de tots els nodes de la xarxa.

2.2.2 OLSR

OLSR (Optimized Link State Routing Protocol) [3] és un protocol unicast proactiu, és a dir que manté actualitzada contínuament la informació de les taules d'encaminament. La

informació de la topologia de la xarxa s'intercanvia mitjançant una inundació controlada de missatges de topologia.

Els missatges HELLO proporcionen informació sobre els veïns a dos salts, de tal manera que cada node pot seleccionar un veí com a MPR (Multi-Point Relay). Els MPRs són els nodes encarregats d'enviar els missatges de topologia (missatges TC) a tots els nodes de la xarxa de manera controlada. Amb la informació de la topologia de la xarxa, cada node pot construir la seva taula d'encaminament per tal de poder enviar missatges als altres nodes.

OLSR té un bon rendiment en xarxes de mida petita i mitjana, on la densitat dels nodes és relativament alta. El coneixement de la topologia, juntament amb el bon rendiment sota patrons de comunicació densos, fa que OLSR sigui un bon protocol per la comunicació de grup en xarxes MANET.

2.3 Protocols multicast a nivell de xarxa

Una de les aplicacions més usuals de les MANETs és la comunicació de grups, és a dir que diversos nodes de la xarxa es constitueixen com a grup i estableixen una comunicació on tots els nodes del grup reben els missatges que envia cada node. Una manera bastant comuna de proporcionar aquesta comunicació és mitjançant el flooding (inundació). Amb aquesta aproximació, les dades s'envien a tota la xarxa MANET, perquè quan un node rep un paquet, el reenvia per broadcast a tots els seus veïns. En un xarxa ad hoc d'alta mobilitat, inundar tota la xarxa podria ser una alternativa per aconseguir una comunicació de grup fiable. No obstant, la inundació té un overhead considerable.

Com a alternativa a la inundació, s'han proposat els protocols multicast per a MANETs [4]. Basats en els protocols multicast dels encaminadors de les xarxes fixes, aquests protocols es poden classificar en diferents categories segons com es creen les rutes cap als membres del grup. En concret, per als protocols multicast a nivell de xarxa es diferencien les propostes basades en arbre i les basades en malla. També hi ha propostes híbrides que combinen diferents aspectes de les basades en arbre i les basades en malla. En la següent secció veurem també els protocols multicast a nivell d'aplicació.

En aquesta secció veurem alguns dels protocols multicast a nivell de xarxa més significatius, com ara MAODV o ODMRP.

2.3.1 MAODV

MAODV (Multicast Ad Hoc On-Demand Distance Vector) [5, 6], és un protocol basat en arbre que parteix dels conceptes utilitzats per dissenyar el protocol unicast AODV, predecessor de DYMO. Descobreix rutes sota demanda utilitzant un mecanisme de descobriment de rutes que empra els mateixos missatges de Route Request (RREQ) i Route Reply (RREP) que ja existien a AODV.

Quan un node vol unir-se a un grup multicast envia un missatge RREQ. Només un membre del grup multicast desitjat pot respondre a un missatge RREQ d'unir-se a un grup. Si el missatge RREQ no és una petició d'unió, pot respondre qualsevol node amb una ruta prou fresca (recent) cap al grup multicast. Si un node intermedi rep un RREQ per a un grup multicast o rep un RREQ però no té cap ruta cap a aquell grup, es torna a retransmetre per broadcast el missatge RREQ als seus veïns. A AODV, el missatge RREQ s'utilitza per generar rutes inverses a la taula de rutes, i això mateix passa a MAODV. No obstant, aquesta entrada a la taula de rutes multicast no s'activa fins que la ruta no es selecciona per formar part de l'arbre multicast. Un node ha de respondre a un missatge RREQ d'unió si és membre del grup multicast i el seu número de seqüència per al grup multicast és almenys tan gran com el que conté el RREQ. El node que respon actualitza la seva ruta i les taules de rutes multicast, afegint a les taules la informació del següent salt del node que fa la petició, i llavors envia un missatge RREP unicast cap a la font.

Quan un node font envia per broadcast un RREQ, sovint rep més d'un RREP. No obstant, només manté, per un determinat període de temps, la resposta amb el número de seqüència més alt i el nombre de salts més baix fins al membre més proper de l'arbre multicast. Quan acaba aquest període, s'activa el següent salt seleccionat a la taula d'encaminament multicast i s'envia per unicast un missatge d'activació (MACT) al següent salt seleccionat. Aquest node activarà l'entrada per al node font a la seva taula d'encaminament multicast, propagant el missatge MACT fins que arribi al membre escollit del grup multicast. Un node intermedi pot rebre més d'un missatge d'activació, però només mantindrà el millor salt següent per a la seva ruta cap al grup multicast.

El primer membre del grup multicast esdevé el líder del grup i té la responsabilitat de mantenir el número de seqüència del grup multicast i de retransmetre per broadcast aquest número a tot el grup multicast. L'actualització es fa a tracés d'un missatge Group Hello. Aquests missatges contenen extensions que indiquen l'adreça IP del grup multicast i els números de seqüència. Com que AODV manté l'estat a la taula d'encaminament, el protocol ha de seguir i reaccionar als canvis en el seu arbre. Quan un membre de l'arbre finalitza la pertinença al grup, cal podar algunes branques de

l'arbre. Els enllaços de l'arbre es monitoritzen per detectar trencaments de l'enllaç: el node més antic del grup multicast és el responsable de reparar l'enllaç trencat. Si l'arbre no es pot reconnectar, s'escull un nou líder per a la partició desconnectada.

2.3.2 ODMRP

ODMRP (On-Demand Multicast Routing Protocol) [6, 7], és un protocol multicast per a MANETs basat en malles, que utilitza el concepte de "forwarding group" (només un subconjunt del nodes retransmet els paquets multicast).

Al contrari de MAODV, aquest protocol fa servir una aproximació d'estat lleuger, de tal manera que no es necessita cap missatge de control explícit per abandonar el grup. A ODMRP, la pertinença al grup i les rutes multicast són establertes i actualitzades per la font sota demanda.

Quan una font multicast té paquets per enviar, envia per broadcast un paquet de control Join-Query a tota la xarxa. Aquest missatge es reenvia periòdicament per refrescar la informació de membership i actualitzar les rutes. Quan un node intermedi rep el paquet Join-Query, emmagatzema l'ID font i el número de seqüència per detectar qualsevol duplicat potencial. Si el missatge no és un duplicat i el TTL (time-to-live) del paquet és més gran que zero, el missatge es torna a enviar per broadcast.

Quan un paquet Join-Query arriba a un receptor multicast, aquest crea un missatge Join-Reply que envia a tots els seus veïns per broadcast. Quan un node rep un Join-Reply, comprova l'ID per saber si ell està en la ruta cap a la font. Si és així, s'activa el seu flag de forwarding group (FG_FLAG) i envia per broadcast el seu propi missatge Join-Reply.

Aquest procés construeix o actualitza les rutes des dels nodes font als nodes receptors, creant una malla de nodes. Després d'establir el forwarding group i haver creat les rutes, els nodes font poden enviar paquets per multicast cap als receptors a través de les rutes establertes. Mentre el node tingui dades per enviar, periòdicament s'enviaran paquets Join-Query per refrescar el forwarding group i les rutes.

2.4 Protocols multicast a nivell d'aplicació

Com hem vist en la secció anterior, els protocols com MAODV o ODMRP segueixen aproximacions a nivell de xarxa. Aquest tipus de protocols tenen com a objectiu minimitzar l'ús de l'ample de banda, proporcionant un encaminament fiable i eficient. No obstant, cap de les propostes de protocols multicast per a MANET té

implementacions madures i testejades. Sovint són prototipus per fer simulacions en simuladors com ns-2, o bé implementacions per a unes plataformes concretes. Com veurem més endavant, això representa un problema a l'hora de dissenyar un framework de treball col·laboratiu per a MANETs, ja que dificulta la creació d'una arquitectura de xarxa a baix nivell que suporti les primitives de comunicació de grups que utilitzen els serveis de nivell superior.

L'alternativa als protocols multicast de nivell de xarxa són els protocols de nivell d'aplicació (o protocols multicast overlay), on els paquets multicast s'encapsulen en datagrames unicast que s'entreguen a tots els membres del grup multicast. En els protocols multicast a nivell d'aplicació, només els membres del grup han de mantenir informació d'estat.

Tots aquests protocols suposen que hi ha un protocol d'encaminament unicast per sota, que s'encarrega d'encaminar aquests paquets unicast.

En aquesta secció, enumerarem més protocols que en les anteriors, ja que les nostres propostes s'engloben en aquest grup.

2.4.1 AMRoute

El primer protocol multicast overlay per a MANETs que es va proposar va ser AMRoute [8]. Aquest protocol crea un arbre compartit per distribuir dades fent servir únicament els membres del grup com a nodes. L'arbre compartit es construeix a partir d'una malla virtual, amb túnels unicast que permeten la connexió entre membres del grup. Un dels principals inconvenients de AMRoute és el comportament estàtic de la malla virtual, ja que no es fan canvis en l'estructura un cop s'ha construït. AMRoute no té en compte la dinamicitat de la xarxa i deixa tota la responsabilitat de les transmissions en mans del protocol unicast de sota.

2.4.2 ALMA

El protocol ALMA (Application Layer Multicast Algorithm) [9] crea un arbre d'enllaços lògics entre els membres del grup. L'objectiu d'aquest protocol és reduir el cost de cada enllaç de l'arbre mitjançant la reconfiguració de l'arbre en situacions de congestió i alta mobilitat. Quan un node s'uneix al grup, ha de seleccionar un node com a pare, per tal de formar part de l'arbre. Si el rendiment de l'arbre cau per sota d'un llindar predefinit, el node ha de reconfigurar l'arbre o bé canviant el node pare o bé alliberant els nodes fills. Aquest mecanisme incorpora un sistema complex per detectar i evitar llaços. ALMA també considera l'existència d'un node rendezvous per obtenir

l'estructura de l'arbre lògic així com informació dels veïns durant el procés de bootstrapping.

2.4.3 AOMP

AOMP (Ad-hoc overlay Multicast Protocol) [10] és un protocol multicast de nivell d'aplicació que depèn d'un protocol reactiu unicast per construir un arbre de lliurament de manera dinàmica i descentralitzada. Aquest protocol té dues etapes: una primera que connecta nous nodes a l'overlay i una segona que realitza la construcció i el manteniment de l'arbre. AOMP es beneficia del protocol d'encaminament unicast, evitant l'overhead d'encaminament i millorant l'escalabilitat. No obstant, aquest protocol està limitat a utilitzar protocols reactius com AODV o DSR i només considera un sol node font a la sessió multicast.

2.4.4 NICE-MAN

NICE-MAN [11] incorpora diverses millores al protocol d'Internet NICE, explotant la capacitat broadcast del medi sense fils per tal de reduir el tràfic de la xarxa. La principal millora respecte al seu homònim de les xarxes d'infrastructura és el manteniment d'un overlay reduït format pels líders de cada clúster en que es divideix la xarxa. Els membres que no són líders de clúster estan situats a una distància d'un salt d'algun líder de clúster. Com a conseqüència, un node pot enviar un missatge a diversos nodes simultàniament, beneficiant-se de la naturalesa broadcast del medi. El principal avantatge d'aquest protocol és l'escalabilitat. Tot i això, NICE-MAN té una sèrie d'inconvenients com ara una adaptació qüestionable als canvis de topologia i la selecció contínua de líders de clúster. A més a més, els nodes que no formen part de l'overlay estan dèbilment connectats ja que no envien cap missatge de control. Això podria implicar una alta pèrdua de missatges quan els nodes s'estan recuperant després d'una pèrdua de connexió. En conseqüència, NICE-MAN tampoc pot proporcionar informació de membership.

2.4.5 PAST-DM

PAST-DM (Progressively Adaptive Subtree in Dynamic Mesh) [12] és un protocol multicast overlay basat en la construcció d'una malla virtual dinàmica. La malla es manté dinàmicament a través de l'intercanvi de paquets d'estat de l'enllaç, aconseguint adaptar-se als canvis en la topologia de la xarxa. Aquests paquets proporcionen

informació de l'estat de l'enllaç, és a dir, una vista parcial de la xarxa. Tots els nodes han d'iniciar la sessió multicast simultàniament i tot seguit comencen el procés de bootstrapping, enviant missatges broadcast limitats per un TTL.

Amb la informació sobre la topologia obtinguda de la malla, els nodes calculen un arbre Steiner basat en la font, per tal de poder lliurar els missatges a tots els membres del grup multicast. Les distàncies en salts físics i lògics s'utilitzen com a heurística per calcular l'arbre Steiner. El node font pren els seus veïns virtuals (lògics) com a fills en l'arbre. La resta de nodes s'agrupen en diversos subgrups, els quals formen un sub-arbre l'arrel dels quals és un dels veïns lògics del node font. Així, cada fill de l'arbre font és responsable de lliurar el missatge multicast procedent del node font a tots els nodes del seu sub-arbre. Aquest procés es va repetint a través de cada node fins que els sub-arbres queden buits. La decisió de quina ruta de lliurament de paquets es segueix, correspon a cada receptor, de manera que la selecció de ruta sempre es beneficia de la informació més actualitzada (ja que a cada salt el paquet s'aproxima més al destí i per tant les taules d'encaminament han pogut ser actualitzades abans). Tot i que es tracta d'una forma eficient de lliurar dades, alguns paquets es poden perdre si hi ha canvis de localització, una vegada el node font ha calcular el sub-arbre corresponent.

Els principals avantatges d'aquest protocol són l'adaptació dinàmica als canvis de topologia i el reduït cost de computació dels arbres multicast. En canvi, l'escalabilitat és reduïda ja que cal indicar als paquets de dades les adreces dels receptors. Això mateix és la causa de l'overhead que afegeix aquest protocol als paquets de dades.

Altres propostes de protocols multicast a nivell d'aplicació intenten minimitzar la latència de l'overlay i millorar l'eficiència global tenint en compte algunes altres característiques de la xarxa. Per exemple, el protocol POM [13] està enfocat als overlays prioritzats, i el protocol LGT [14] considera la localització geoespacial dels nodes com una heurística per millorar el procediment de construcció de l'arbre multicast.

3. OMCAST

El protocol OMCAST (Overlay Multicast) és fruit de la recerca duta a terme en el context del projecte POPEYE. Com ja hem dit a la introducció, un dels objectius del projecte era proporcionar unes primitives de comunicació a baix nivell per tal de poder-hi construir aplicacions col·laboratives a sobre. Per poder proporcionar aquestes primitives calia muntar una arquitectura de xarxa amb un protocol unicast per a les comunicacions bàsiques punt a punt i un protocol multicast per a les comunicacions de grup.

Com a protocol unicast es va escollir DYMO, una implementació del qual (DYMOUM) havia estat desenvolupada per un dels socis del consorci (la Universidad de Murcia). Per a les comunicacions multicast vam dissenyar i desenvolupar el protocol OMCAST, sempre tenint en compte les característiques de l'escenari MANET considerat pel projecte. Aquestes característiques són les següents:

- Escenari de treball col·laboratiu: conferència, reunió, ...
- Nombre moderat de nodes localitzats en una àrea reduïda.
- Probable disposició física en grups de treball.
- Comportament poc dinàmic dels nodes.
- Gran dependència de la comunicació de grup.

Es va optar per un protocol a nivell d'aplicació perquè proporcionen una millor facilitat de desenvolupament i de desplegament, i capacitat d'ocultar els errors de la capa d'enllaç. A més a més, els protocols multicast a nivell d'aplicació només requereixen que els membres del grup mantinguin la informació d'estat.

El protocol OMCAST vol treure profit de la naturalesa broadcast del medi, cosa que no tenen en compte els protocols ALMA i PAST-DM, considerats els més eficients a nivell d'aplicació. En aplicar aquests protocols a entorns col·laboratius presenten una sèrie d'inconvenients. Per exemple, PAST-DM té un procés de bootstrap molt rígid on tots els membres de la xarxa s'hi han d'unir simultàniament, incompatible amb una inicialització escalonada i independent de cada node.

Per això, com que creiem que explotar les capacitats broadcast del medi és la clau per dissenyar un protocol eficient per a escenaris MANET col·laboratius (conferències científiques, reunions, treball en grup), presentem el protocol OMCAST, un protocol d'encaminament multicast per a MANET a nivell d'aplicació dissenyat per escenaris col·laboratius. Així, les principals característiques del protocol OMCAST són:

- Un procés de bootstrapping flexible, que permet que els nodes s'uneixin al grup en qualsevol moment.
- Utilització dels enviaments broadcast de paquets de dades als veïns situats a un salt per reduir l'overhead de la comunicació.
- Disponibilitat de la informació de membership de forma descentralitzada per a les capes superiors.

3.1 Descripció del protocol

El protocol OMCAST es basa en l'intercanvi d'informació d'estat de l'enllaç i en la computació local dels arbres d'expansió mínima. Mitjançant l'intercanvi d'aquesta informació es construeix una malla virtual que connecta tots els membres del grup multicast. A més a més, OMCAST proporciona un procés de bootstrapping que millora el d'altres protocols com PAST-DM. Però la principal diferència amb els protocols descrits és que té en compte la naturalesa broadcast del medi per reduir el tràfic en situacions de densitat de nodes.

4.1.1 Procediment de bootstrapping

El protocol OMCAST incorpora un procediment d'inicialització que permet que els nodes s'uneixin a un grup multicast en qualsevol moment. Qualsevol node font o receptor que vol participar a la sessió multicast ha d'enviar un missatge JOIN. Aquest missatge consisteix en un enviament broadcast local (és a dir a tots els veïns a un salt), que es retransmès successivament pels nodes que no són membres del grup. Quan el missatge arriba a un membre del grup, s'envia un missatge unicast cap a la font com a resposta. Aquest missatge de resposta conté informació de l'estat de l'enllaç d'aquell node membre, així com el nombre de salts que hi ha des del node font al node membre del grup. D'aquesta manera, el node que vol entrar al grup rep informació sobre els seus veïns virtuals (veïns de grup), que són els nodes del grup més propers a ell. A partir d'aquest moment, el node està preparat per enviar missatges multicast a tots els nodes del grup.

4.1.2 Manteniment de la malla

El procediment de manteniment de la malla a OMCAST és semblant al procediment que utilitza PAST-DM. L'estructura de dades bàsica que utilitza el protocol és la taula

d'estat de l'enllaç (LST, Link State Table). Es necessita una taula per cada grup multicast que es vulgui gestionar. Les entrades més recents d'aquesta taula s'intercanvien periòdicament amb els nodes veïns virtuals, és a dir, amb els membres del grup multicast que estan situats a un salt lògic del node local.

Llavors, cada taula d'estat de l'enllaç crea una entrada d'estat de l'enllaç (LSE, Link State Entry) per cada membre del grup multicast en qüestió. Cada LSE manté informació sobre l'estat de l'enllaç del node corresponent: grau del node i veïns virtuals, així com un identificador del node i un temps d'expiració de l'entrada. Valors baixos en el temps d'expiració indiquen que la informació d'aquella entrada és menys acurada que les entrades que tenen un temps més alt. Quan aquest valor arriba a un cert límit, l'entrada es marca com a invàlida. Quan es reben versions actualitzades de les entrades d'un node, l'entrada és torna a marcar com a vàlida i s'assigna el valor més alt al temps d'expiració.

Gràcies al manteniment de la informació de l'estat de l'enllaç, els nodes poden calcular l'arbre d'expansió mínima, i així poder enviar i lliurar paquets de dades a tots els membres del grup. A més a més, els nodes poden proporcionar informació de membership a les capes altes. Aquesta informació és molt útil en aplicacions col·laboratives, ja que es pot tenir la noció de quins nodes estan actius i controlar l'entrada i sortida de nodes al grup. Per proporcionar la informació de membership, els nodes recuperen una llista d'identificadors de node de cada entrada de la taula.

4.1.3 Enviament i recepció de missatges

Quan un node vol enviar un missatge a la resta de nodes del grup, ha de computar l'arbre d'expansió mínima. El càlcul de l'arbre d'expansió mínima s'ha efectuat seguint l'algoritme de Prim. Aquest algoritme troba un subconjunt d'arestes que formen un arbre que inclou tots els vèrtexs de tal manera que el pes total de totes les arestes sigui el mínim. Donat que els pesos de les arestes del graf format pels enllaços virtuals és el nombre d'enllaços físics, l'arbre d'expansió mínima assegura que el nombre total de salts que haurà de fer el missatge multicast sigui mínim. A la Fig. 2 es pot veure un procediment de càlcul de l'arbre d'expansió mínima. Els nodes negres són els que formen part del grup, mentre que els nodes blancs no hi estan units. El node A és el node font.

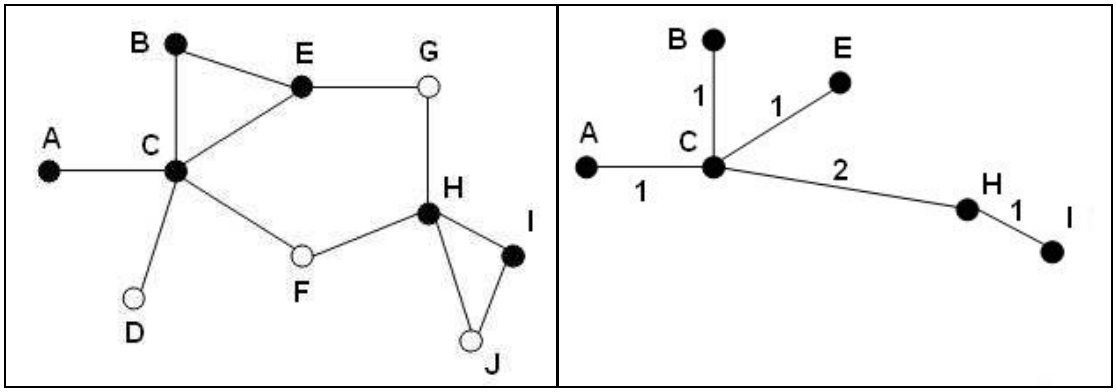

Fig. 2: Càlcul de l'arbre d'expansió mínima. El número de cada enllaç virtual és el nombre de salts físics per aquell enllaç.

Una vegada calculat l'arbre (de fet pot estar pre-calculat), el node font envia una còpia del paquet de dades a cada veí virtual. Cadascun d'aquests veïns és responsable de lliurar el missatge a un cert subconjunt de nodes, que vindrà definit per l'arbre creat al node font. Llavors, juntament amb el contingut del paquet, s'afegeixen unes capçaleres amb informació sobre el subconjunt de nodes que encara han de rebre el missatge. El veí virtual rep el missatge i calcula l'arbre només per al subconjunt de nodes que s'inclou a la capçalera. Després es repeteix el procés: el node envia una còpia del missatge als seus veïns virtuals amb el nou contingut de la capçalera. I així es va repetint el procés a cada salt fins que el subconjunt de nodes de la capçalera dels missatges és buit.

En altres protocols com PAST-DM, quan un node vol enviar o reenviar un missatge a través de l'arbre generat, primer comprova la seva informació d'estat. Llavors, el node genera tantes còpies com nodes han de rebre el missatge. Per tant es generarà un paquet unicast diferent amb la còpia del missatge i s'enviarà a cada node receptor. Amb OMCAST proposem fer servir una tècnica d'estalvi d'ample de banda consistent en enviar el missatge a través d'un enviament broadcast local. Si hi ha prou veïns virtuals situats a un salt físic, llavors es pot enviar una única còpia del missatge fent servir broadcast local. D'aquesta manera, es redueix l'ample de banda utilitzat quan els membres del grup volen enviar missatges a diversos membres que estan en el rang de transmissió. A la Fig. 3 es pot veure una comparativa gràfica del mecanisme habitual d'enviar una còpia a cada veí amb el mecanisme del broadcast local. Com es pot veure en la primera gràfica, el paquet m es copia i s'envia 4 vegades a través de la xarxa. En canvi, en la segona gràfica, el missatge s'envia en un sol paquet de manera que els nodes veïns poden rebre'l i s'evita la repetició en el lliurament dels missatges. A la Fig. 4 es pot veure un exemple complet de transmissió d'un missatge multicast des d'un node cap a tots els membres del grup.

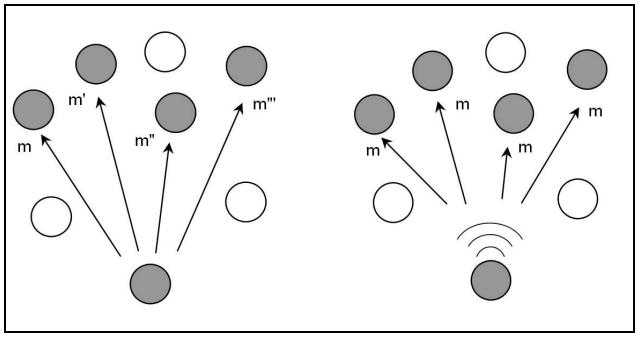

Fig. 3: Comparativa entre l'enviament de múltiples unicast (esquerra) i de broadcast local (dreta).

Cal tenir en compte que, encara que els escenaris col·laboratius que prenem com a base són bastant estàtics, els nodes poden canviar de posició de forma inesperada. Per això és possible que els paquets que s'envien a través del broadcast local no arribin a la seva destinació a causa de problemes de mobilitat, com per exemple una informació de veïns virtuals poc acurada. Per tal de minimitzar aquestes pèrdues, el lliurament broadcast només es realitza quan es compleixen certes condicions:

- El nombre de veïns virtuals que estan preparats per rebre el missatge ha de ser major que un llindar MIN_BC_NEIGH.
- Un veí virtual es considera preparat per rebre missatges broadcast si el node font ha rebut un missatge d'estat de l'enllaç d'aquell veí en els últims BC_PERIOD segons. D'aquesta manera el protocol s'assegura que un node enviarà missatges broadcast només als nodes que estaven situats a un salt físic en un període de temps recent. Així s'evita l'enviament de missatges a nodes que potencialment poden haver canviat de localització.

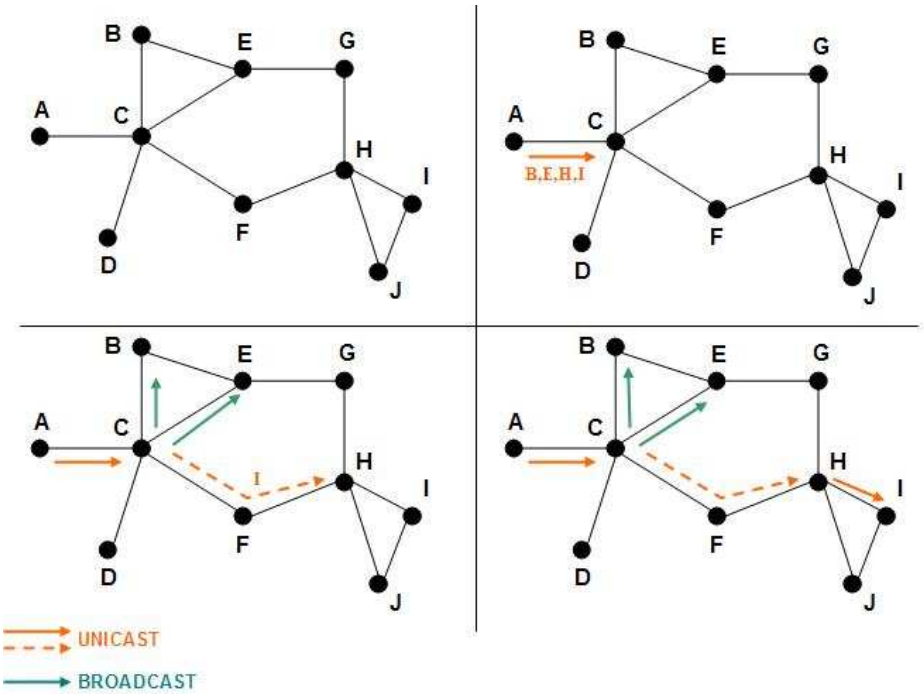

Fig. 4: Procés d'enviament d'un missatge multicast en OMCAST. El node A és el node font. Només pertanyen al grup els nodes A, B, C, E, H, I. Els enllaços són els físics.

Un altre problema causat per la mobilitat es produeix quan un node ha de retransmetre un paquet cap a un receptor que no apareix al seu arbre. Això pot passar perquè el node font és el responsable d'assignar els primers subconjunts de nodes. OMCAST resol aquest problema forçant els nodes retransmissors a enviar aquests paquets per unicast als receptors sempre que aquests no apareguin a l'arbre.

3.2 Disseny de la implementació

La implementació funcional del protocol OMCAST permet usar el protocol per a l'enviament de missatges multicast en una xarxa MANET que tingui un protocol d'encaminament unicast a nivell de xarxa. Per facilitar el desenvolupament, s'ha fet servir el toolkit JGroups.

JGroups és un conjunt de protocols per a comunicacions multicast fiables. La característica principal d'aquest framework és la seva pila de protocols flexible i adaptable. En aquesta pila es poden posar diferents protocols per adreçar diferents requeriments de xarxa: fiabilitat, ordenació de missatges, detecció de fallades, membership de grups o fragmentació de missatges.

OMCAST s'ha implementat com un protocol JGroups, seguint la interfície de disseny de protocols. D'aquesta manera, OMCAST es pot afegir a una pila de protocols JGroups i les aplicacions poden accedir fàcilment a primitives bàsiques i notificacions d'events.

JGroups proporciona un canal de comunicació de grups que accedeix de forma transparent a l'enviament de missatges multicast d'OMCAST. A més a més, OMCAST s'utilitza per proporcionar la informació de membership a JGroups, que a la vegada la passa a les aplicacions de nivell superior.

3.3 Avaluació

L'avaluació del protocol OMCAST s'ha fet mitjançant un entorn d'emulació sobre una xarxa Ethernet. Per aproximar-se a les característiques multi-hop d'una xarxa MANET, s'ha fet servir un entorn d'emulació sobre imatges VMWare que utilitzen una versió modificada del programa MobiEmu [16]. Aquest programa permet emular la mobilitat dels nodes enviant comandes a cada node per tal que permetin o deneguin la connexió amb altres nodes. La versió original del programa MobiEmu utilitzava IPtables, però la nostra versió utilitza APE MACKill [15]. L'entorn d'emulació fa servir DYMOUM [17] (una implementació de DYMO) com a protocol d'encaminament unicast.

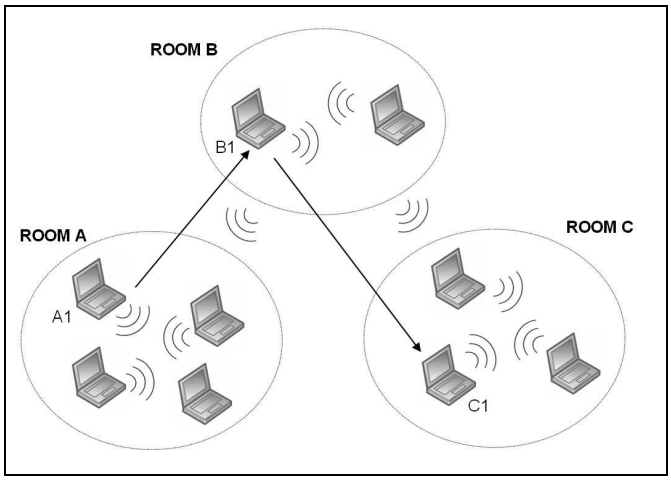

Fig. 5: Escenari col·laboratiu en tres àrees diferenciades

L'escenari més usual en les simulacions de protocols MANET és el Random Waypoint Model, però l'hem descartat perquè no s'adequa als patrons de mobilitat dels escenaris col·laboratius. Així doncs el tipus d'escenaris escollits per a l'emulació són diversos escenaris col·laboratius, com el representat a la Fig. 5, on els nodes estan localitzats de manera densa en tres àrees diferenciades. En cada escenari hi ha alguns nodes que asseguren la connexió entre les diferents àrees. Les emulacions duren 140 segons, i els nodes es van movent entre les àrees. Els nodes que són membres del grup multicast envien paquets de dades a una ràtio constant de 4 Kb/s si no s'especifica el contrari en cada emulació.

Els tres conceptes que s'han estudiat en l'emulació són l'impacte del lliurament broadcast, la taxa de lliurament de paquets de dades i el retard *round trip time*.

En primer lloc s'ha avaluat l'impacte del lliurament broadcast en els paquets de dades. S'ha calculat el tràfic que es genera quan s'utilitza l'enviament unicast entre salts físics amb el que es genera utilitzant l'enviament broadcast. Es vol provar que la utilització de l'enviament broadcast en àrees denses afavoreix la disminució del tràfic que ha de suportar la xarxa.

L'escenari seleccionat està format per 20 nodes membres del grup, que envien paquets de dades a diferents ràtios de bits. El paràmetre MIN_BC_NEIGH es canvia per restringir l'ús del broadcast a diferents valors. Això implica que només s'enviarà broadcast quan hi hagi almenys MIN_BC_NEIGH veïns virtuals preparats per rebre el missatge. OMCAST-2, OMCAST-3, OMCAST-5 indiquen que el valor de MIN_BC_NEIGH és 2, 3 o 5 respectivament. Per altra banda, OMCAST-NOBC indica que el lliurament broadcast local està desactivat, i per tant tots els missatges entre salts s'envien per unicast.

Com es pot veure a la Fig. 6, el tràfic de la xarxa s'ha reduït de forma significativa. Podem observar que si el protocol es beneficia de la capacitat broadcast del medi, el tràfic es redueix gairebé un 20%. En les situacions on s'usa el lliurament broadcast local s'envien menys paquets a la xarxa. El tràfic extra que es crea quan no s'usa el broadcast pot produir retard d'extrem a extrem i incrementar la congestió i la conseqüent pèrdua de paquets. La variació del llindar d'ús del broadcast sembla no influir en el tràfic generat, donada la densitat dels grups utilitzats en l'emulació.

La segona avaluació s'adreça a la influència de les transmissions broadcast en la taxa de lliurament de paquets. Cal destacar que els paquets unicast estan destinats a un node receptor específic, i per tant el protocol d'encaminament unicast acostuma a encaminar-lo correctament cap a la seva destinació. En canvi, els missatges broadcast estan adreçats als nodes situats a un salt físic del node font, i per tant només arriben als nodes situats en el rang de transmissió del node font. Com a conseqüència, un node pot no rebre un missatge broadcast que segons el protocol OMCAST se suposa que havia de rebre, degut a inconsistències temporals en la informació dels veïns virtuals. No obstant, com s'ha vist en la figura anterior, el tràfic global es redueix quan s'utilitza la retransmissió broadcast, i per tant la congestió pot veure's disminuïda assegurant així una taxa de lliurament de paquets alta. Prenent els mateixos paràmetres que en l'emulació prèvia, la Fig. 7 mostra que la taxa de lliurament del protocol OMCAST amb capacitats broadcast supera en gairebé un 15% al mateix protocol amb els enviaments broadcast desactivats.

L'última figura mostra l'efecte de la retransmissió broadcast en el retard extrem a extrem. Tenint en compte les figures prèvies, el retard també hauria de reduir-se en un percentatge significatiu. En aquest cas el retard es mesura per diferents mides de grup.

En aquesta emulació es computa el retard mitjançant l'enviament de diversos missatges multicast amb una marca de temps. Els nodes que reben aquest missatge responen a la font amb un missatge ACK unicast que conté la marca de temps del missatge original. D'aquesta manera el node font recopila tots els missatges i pot establir el round-trip-time cap a tots els nodes. A la Fig. 8 es mostra la mitjana global per les diferents mides de grup.

Com es pot veure, el round-trip-time és més alt quan s'utilitza lliurament no-broadcast. La reducció del retard quan s'utilitza lliurament broadcast es pot explicar per la reducció del tràfic global de la xarxa.

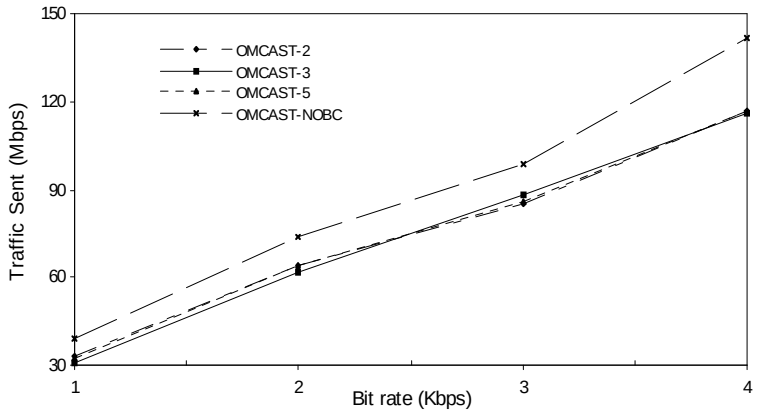

Fig. 6: Tràfic global a la xarxa

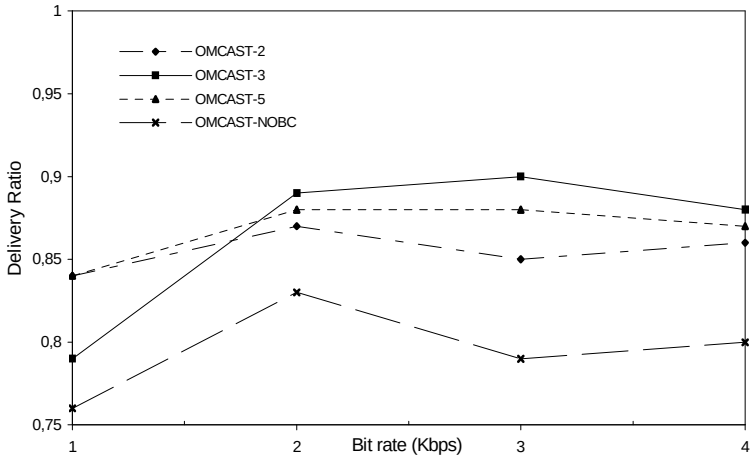

Fig. 7: Taxa de lliurament de paquets

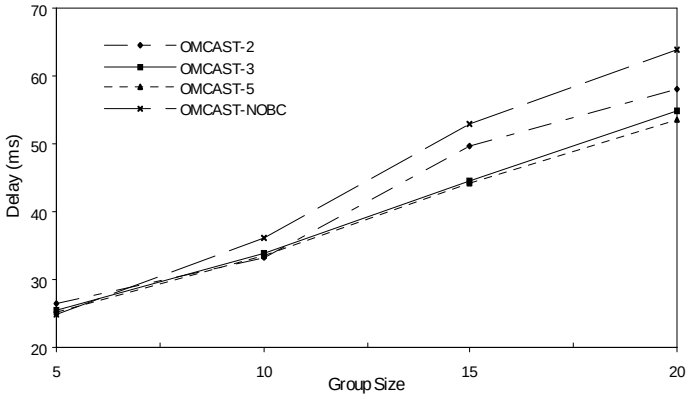

Fig. 8: Retard extrem a extrem

4. jOLSR i OMOLSR

El protocol OMCAST necessita un protocol unicast a nivell de xarxa que encamini els paquets que encapsulen els missatges multicast per enviar-los a través de l'arbre de lliurament. És a dir que s'utilitzen dos protocols d'encaminament alhora sobre la mateixa xarxa, un per encaminar tràfic unicast i l'altre per al tràfic multicast. Ara bé, aquests dos protocols no interactuen ni comparteixen informació sobre la topologia de la xarxa. Per tant, cadascun dels protocols enviarà missatges de descobriment de rutes a través de la xarxa amb independència del que enviï l'altre. És evident que això pot ser la causa d'un important overhead de missatges de control.

Per fer front a aquest problema i intentar treure profit del protocol de nivell de xarxa, s'han proposat diverses solucions cross-layer que possibiliten l'ús de la informació de topologia al nivell d'aplicació. No obstant, aquestes solucions cross-layer no respecten l'estructura del model de capes de xarxa i són difícils d'adaptar en diferents escenaris.

Per això vam considerar completament justificat el fet de moure el protocol d'encaminament unicast del nivell de xarxa al nivell d'aplicació. Tenir els dos protocols d'encaminament (unicast i multicast) a nivell d'aplicació proporciona una sèrie d'avantatges que cal considerar:

- La interacció entre el middleware i els protocols d'encaminament és més neta i senzilla.
- Els protocols es poden adaptar per requeriments específics.
- Es simplifica el desenvolupament, les proves i el desplegament d'aplicacions.
- Les aplicacions es poden beneficiar de la informació de la topologia de xarxa subjacent.

Fruit d'aquesta idea van sorgir els protocols jOLSR i OMOLSR. jOLSR és una implementació Java a nivell d'aplicació del protocol d'encaminament unicast OLSR, i OMOLSR és el protocol multicast overlay que s'executa per sobre de jOLSR i que treu profit de la informació obtinguda per aquest. Els dos protocols s'integren en un middleware preparat per a ser utilitzat com a base per al desenvolupament d'aplicacions col·laboratives sobre xarxes MANET.

4.1 Descripció del protocol

El middleware format pels dos protocols ofereix un MChannel (multi-hop channel) a les aplicacions per poder comunicar-se en xarxes MANET multi-hop. Els protocols

d'encaminament recullen els missatges generats al M-channel i s'encarreguen de fer-los arribar als seus destinataris de forma transparent a les aplicacions. L'estructura bàsica del middleware, representada a la Fig. 9, és la següent:

- OMOLSR calcula l'arbre de lliurament multicast a partir de la informació rebuda via events des de jOLSR. Aquests events permeten que OMOLSR actualitzi periòdicament la seva malla local de la xarxa (ONG, OMOLSR Network Graph).
- Quan el MChannel vol enviar un missatge multicast, OMOLSR crea diferents còpies del missatge, les encapsula en paquets OMOLSR i els envia per unicast a diferents nodes situats a un salt lògic. Una capa intermitja proporciona la fiabilitat unicast, és a dir que s'encarrega de retransmetre els paquets que s'han perdut.
- jOLSR s'encarrega de fer arribar aquests paquets unicast als seus receptors. jOLSR manté informació de la xarxa en diferents taules: taula de veïns (NIB, Neighbor Information Base), taula d'estat del enllaços amb els veïns (LLIB, Local Link Information Base), i taula de topologia (TIB, Topology Information Base). A més a més d'aquestes taules, com a requeriment especial d'OMOLSR, també es manté la taula de grups multicast (MGT, Multicast Groups Table) que recopila informació sobre quins nodes estan presents en cada grup multicast.
- A les capes inferiors, jOLSR utilitza el protocol UDP per enviar paquets de control (fent servir enviaments broadcast) i de dades (amb enviaments unicast).

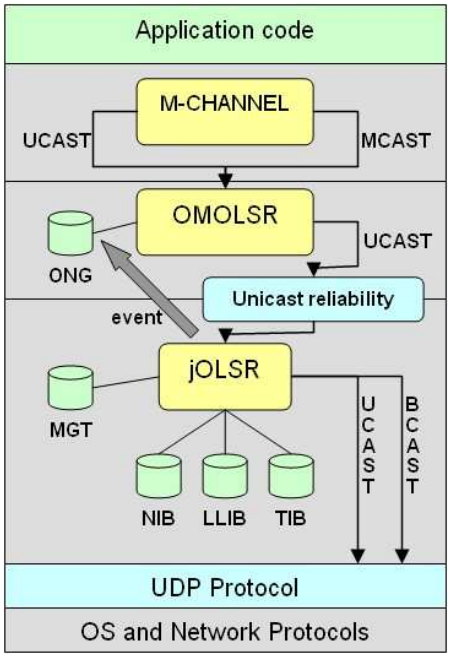

Fig. 9: Arquitectura de jOLSR-OMOLSR

A l'Annex 2 d'aquesta memòria hi ha la guia de desenvolupador d'OMOLSR, on s'explica en detall la funcionalitat de cadascuna de les taules i dels missatges de control, així com detalls de la implementació.

4.1.1 jOLSR

jOLSR és una implementació Java a nivell d'aplicació del protocol d'encaminament OLSR. El protocol OLSR està definit al RFC 3626. Per simplificar la implementació, algunes funcionalitats d'OLSR com el suport a múltiples interfícies no s'han considerat a jOLSR, que tot i això segueix les funcionalitats bàsiques descrites al RFC. Així mateix hi ha algunes diferències en el format dels paquets, ja que en ser una implementació a nivell d'aplicació alguns camps eren innecessaris.

A més a més, ha estat necessari realitzar algunes modificacions a l'especificació bàsica d'OLSR per tal de proporcionar informació de topologia i de membership de grup al protocol multicast de nivell superior. OMOLSR, el protocol multicast, traurà profit d'aquesta informació i així evitarà enviar els seus propis missatges de control a través de la xarxa. Aquests canvis en la implementació de jOLSR són els següents:

- Els missatges TC inclouen una llista dels grups multicast als quals s'ha unit el node local.
- jOLSR incorpora una nova taula a les seves estructures. Aquesta taula manté la informació sobre els grups multicast que es rep amb els missatges TC. Amb aquesta informació, i la de la resta de taules, es pot generar una representació parcial de la xarxa que forma cada grup multicast mitjançant un graf. Aquest graf es calcula cada vegada que es detecten canvis en les diferents taules. El graf està disponible per al protocol multicast de nivell superior.

Tant jOLSR com OLSR envien dos tipus de missatges de control bàsics: missatges HELLO i missatges TC (Topology Control). Els missatges HELLO permeten que un node conegui els seus veïns a un i dos salts, ja que s'envien mútuament informació dels seus veïns. D'aquesta manera quan un node rep un missatge HELLO d'un veí, el missatge contindrà informació sobre els veïns d'aquell veí, que a la vegada poden estar a un salt del node que rep el missatge o bé a dos salts. Prenent com a base aquesta informació, el node pot seleccionar el seu MPR (multipoint relay) que s'encarrega de controlar el broadcast per inundació. Aquest mecanisme de broadcast controlat és el que permet que certs missatges arribin a tots els nodes de la xarxa.

Els missatges TC són un exemple de missatges que genera un node i arriben a tota la xarxa mitjançant el broadcast controlat. Cada node envia periòdicament un missatge TC que conté els nodes que han escollit al node que envia el missatge TC com a Multi-

Point Relay. És a dir, que tots els nodes rebran informació sobre els enllaços a MPR que tenen els altres nodes i així poden crear una visió parcial, però molt útil, de la xarxa.

Per aprofitar aquests enviaments de missatges TC creiem justificat afegir-hi la informació de grups multicast, de manera que aquesta informació s'escamparia eficientment per la xarxa. La informació afegida a cada missatge és reduïda (es podria limitar a un byte identificador per grup multicast) i suposa que el protocol d'encaminament multicast obtindrà la informació que necessita sobre els membres que s'han unit a cada grup. La modificació del format d'aquest missatge es pot veure a la Fig. 10.

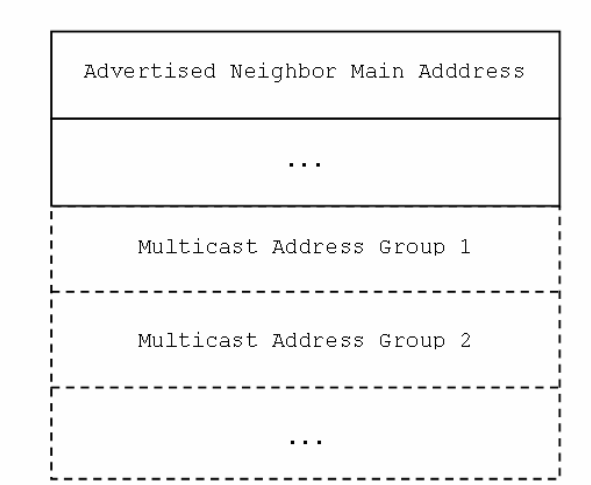

Fig. 10: Modificació missatge TC

La segona modificació important a jOLSR respecte l'estàndard OLSR és la incorporació d'una taula de grups multicast (MGT). Aquesta taula manté la informació dels grups on s'ha unit cada node, i s'actualitza cada vegada que el node local s'uneix a un nou grup, abandona un grup o es rep un missatge TC.

Per cada node de la xarxa es manté una llista dels grups als que pertany. El protocol multicast de nivell superior (OMOLSR) utilitza la informació d'aquesta taula. A més a més, cada canvi que s'hi produeix s'informa a les aplicacions mitjançant events. Quan es detecta un canvi a la taula de veïns, a la taula de topologia o a la taula de grups multicast, es genera un graf representant els membres del grup. Aquest graf es passa al protocol multicast mitjançant un nou event.

El procediment per crear el graf dels membres del grup és el següent:

- S'obté una representació aproximada de la xarxa, creant un graf de xarxa a partir de la informació de la taula de topologia (enllaços significatius entre els nodes).
- Es consulta a la taula de grups multicast quins nodes pertanyen a cada grup.

- Es crea un event per cada grup diferent de la taula.
- El protocol multicast rep l'event amb el graf i la informació dels nodes del grup.

4.1.2 OMOLSR

OMOLSR (Overlay Multicast over OLSR) és el protocol d'encaminament multicast a nivell d'aplicació que s'ha dissenyat per treballar sobre jOLSR. Aspectes com el càlcul de l'arbre multicast o l'encaminament dels paquets estan basats en el treball previ d'OMCAST. Però la principal diferència és que OMOLSR no necessita enviar missatges de control a la xarxa: el protocol d'encaminament unicast ja proporciona tota la informació necessària.

Funcionalitats bàsiques

Les funcionalitats bàsiques d'OMOLSR són la computació dinàmica de l'arbre d'expansió mínima i l'encaminament del paquets multicast. En primer lloc, OMOLSR genera una malla virtual que connecta tots els membres del grup. Aquesta malla és una contracció del graf de la xarxa que genera jOLSR, amb la particularitat que només conté els membres del grup. A partir d'aquesta gràfica, OMOLSR sap quins nodes del grup estan a un salt lògic: els veïns virtuals del node local. El procediment de computació del graf contret es realitza una vegada per cada grup multicast del qual es membre el node local. A partir d'aquest graf, OMOLSR computa l'arbre d'expansió mínima prenent el node local com a font, de la mateixa manera que es fa a OMCAST. L'arbre resultant és el que utilitza per encaminar els paquets multicast cap als altres membres del grup.

L'encaminament multicast és l'altra funcionalitat bàsica d'OMOLSR. S'utilitza el que s'anomena un esquema de multi-unicast explícit. Quan l'aplicació genera un paquet multicast nou, OMOLSR encamina el missatge segons l'arbre computat per aquell grup multicast. S'envia una còpia del paquet a cada veí virtual, que és responsable de lliurar-lo, directa o indirectament, a un subconjunt de nodes. D'aquesta manera, el node font realitza el primer repartiment de nodes que han de rebre el missatge, de manera que cada veí virtual s'encarregui de la part de la xarxa que té més accessible. Aquests subconjunts de nodes són doncs els nodes que estan al sub-arbre de cada veí virtual. Cada missatge enviat a un veí virtual inclou una capçalera amb el subconjunt de nodes associat. Quan el veí virtual rep el missatge, calcula un arbre a partir del subconjunt de nodes continguts a la capçalera. Una vegada calcular l'arbre es repeteix el procés: envia una còpia del missatge als seus veïns virtuals amb un nou contingut a la capçalera. El procés es repeteix fins que, en algun node receptor, el subconjunt que ha de rebre el

missatge és buit. L'algoritme de l'encaminament de paquets multicast és doncs gairebé igual que en el cas d'OMCAST. Les principals diferències són la provinença de la informació de la topologia (que en el cas d'OMCAST s'aconseguia mitjançant missatges propis de control) i l'ús de l'enviament broadcast local (que en OMOLSR no s'ha implementat per simplificar la integració amb jOLSR).

Fiabilitat unicast

Com hem vist en l'estructura del middleware, s'ha incorporat una capa de fiabilitat unicast que proporciona retransmissió de paquets perduts i garanteix l'ordre de les comunicacions. D'aquesta manera, el canal ofert a les aplicacions ja no és una simple façana sinó que es garanteix una certa fiabilitat en les comunicacions. Aquesta capa de fiabilitat utilitza un esquema d'*acknowlegment* per proporcionar transmissió sense pèrdues de missatges unicast. Com que també cal proporcionar aquesta fiabilitat en les transmissions multicast, aquesta capa està situada entre els dos protocols. Així doncs, quan OMOLSR separi cada paquet multicast en diversos paquets unicast, aquests últims passaran a través de l'esquema d'ACK i se'ls proporcionarà una fiabilitat punt a punt entre els veïns lògics.

4.1.3 MChannel

Seguint l'arquitectura descrita, hem desenvolupat un canal que doni les primitives bàsiques per a la comunicació de grups sobre la xarxa MANET. Des de l'òptica de l'aplicació, es crea un MChannel per a un grup determinat, de manera que proporcionarà la possibilitat d'enviar missatges multicast a tots els peers que hagin creat un MChannel per al mateix grup. El MChannel també permet enviar un missatge unicast a qualsevol membre del grup. Si l'aplicació necessita crear 2 grups diferents, haurà de crear dos MChannels, un associat a cada grup. L'API del MChannel proporciona els següent serveis bàsics:

- Obtenció de la llista de membres del grup associat. També es dóna la possibilitat de registrar listeners dels events que es generen cada cop que un peer entra o surt del grup.
- Obtenir els nodes situats a 1 salt (veïns físics del mateix grup) o a N-salts.
- Obtenir un veí ben connectat: a partir de la informació sobre els Multi-Point Relays que recull jOLSR, el MChannel pot proporcionar una llista de veïns ben connectats, és a dir que tenen molts veïns i donen accés a molts altres nodes a altres parts de la xarxa.

Aquestes són les funcionalitats bàsiques que proporciona el MChannel. Ara bé, seria molt senzill modificar-lo per a que oferís serveis més complexos per a aplicacions sobre MANETs. Per exemple, es podria oferir Qualitat de Servei (QoS) en l'encaminament tenint en compte diversos paràmetres alhora (ample de banda, bateria, potència de càlcul,...) i proporcionant així un middleware adaptatiu. També es podria oferir un enviament multicast limitat usant els paràmetres de TTL relacionats amb els salts en la MANET. Aquest sistema permetria restringir els missatges multicast als nodes propers. Fins i tot es podria implementar de forma senzilla un servei anycast (encaminament al millor o més proper).

Fent el paral·lelisme amb les xarxes P2P sobre Internet, creiem que es podria definir una mena de Common API estàndard per als components MANET, que podria definir-se sobre la base esmentada. D'aquesta manera les aplicacions podrien ser independents dels protocols o del middleware utilitzat, sempre que fessin servir les crides i events definits a la Common API.

4.1.4 Avantatges i inconvenients

Una proposta com la que planteja jOLSR-OMOLSR amb el fet de moure tota la lògica d'encaminament al nivell d'aplicació té avantatges clars però també alguns inconvenients.

Entre els principals avantatges trobem la facilitat d'interacció entre els diferents nivells de l'aplicació. L'intercanvi d'informació entre els dos protocols és molt més senzill que si el protocol unicast treballés a nivell de xarxa. Les complexes solucions cross-layer no són necessàries, ja que l'encaminament MANET és un dels components del middleware. També s'afavoreix la flexibilitat i l'adaptació de les solucions adoptades per a usos més concrets. Amb la lògica d'encaminament a nivell d'aplicació adaptar-se a uns requeriments específics pot ser tan senzill com canviar un component del middleware.

Però sens dubte, el principal avantatge és la facilitat de desenvolupament, prova i desplegament. El middleware es pot crear de forma que sigui independent del sistema operatiu utilitzat. Es guanya independència respecte protocols a nivell de nucli de sistema operatiu, amb dependències de llibreries i configuracions del hardware. S'afavoreix la portabilitat de les aplicacions a entorns com els telèfons mòbils o PDAs.

L'inconvenient principal és el fet que aquest servei d'encaminament és específic per a les aplicacions que corrin sobre el framework, mentre que els protocols a nivell de xarxa serveixen a tots els processos del sistema operatiu. Aplicacions preexistents i que utilitzin protocols estàndards TCP/IP no es poden fer servir automàticament sobre els

nostres protocols. Aquest problema, però, es podria solucionar mitjançant xarxes privades virtuals peer-to-peer (P2P VPN), un sistema que es fa servir a Internet amb les xarxes P2P. Aquest component podria proporcionar serveis de nivell de transport estàndard per a aquestes aplicacions pre-existents.

4.2 Disseny de la implementació

Per a la implementació d'aquest middleware format pels dos protocols (jOLSR i OMOLSR) s'ha seguit el mateix model que en OMCAST: el toolkit JGroups.

Com s'ha explicat en la definició de l'arquitectura, s'ha utilitzat una capa de fiabilitat unicast. Aquesta capa és un dels protocols que proporciona JGroups (UNICAST) per tal d'oferir connexions unicast sense pèrdues de paquets i ordenades.

També s'ha aprofitat la flexibilitat de la pila de protocols JGroups per crear un entorn de validació del protocol. El protocol de més baix nivell de JGroups és UDP, que s'encarrega de mantenir els sockets de comunicació unicast. Per crear l'entorn de validació hem substituït aquest protocol per un de nou EMULATION (veure Fig. 11). Aquest protocol virtualitza la comunicació de manera que els missatges no surten de la xarxa sinó que són lliurats a un altre node que està corrent sobre la mateixa màquina. D'aquesta manera s'ha validat el protocol sense haver d'utilitzar forçosament entorns reals. També s'han pogut recollir estadístiques de comunicació entre nodes i de validesa de la informació de membership. Com es pot veure al manual del desenvolupador, inclòs com a annex en aquesta memòria, es poden afegir tasques personalitzades de recol·lecció d'estadístiques a l'entorn d'emulació.

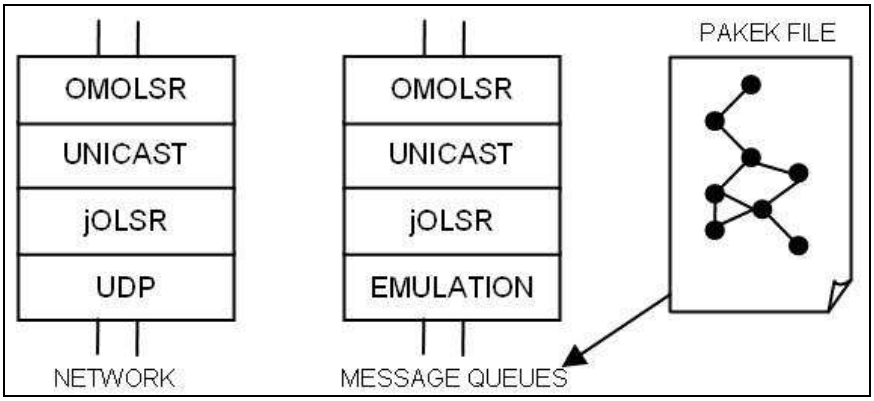

Fig. 11: Implementació de l'entorn d'emulació.

El codi font de la implementació completa del middleware està disponible al wiki del grup de recerca en Arquitectura i Serveis Telemàtics [18].

4.3 Avaluació

Com hem comentat en la secció anterior, abans de fer proves reals amb el middleware, es va provar la implementació sobre un entorn virtualitzat. Una vegada provat satisfactòriament, es van fer diverses proves amb màquines reals en mode ad-hoc, per finalment fer una prova-demostració que va comptar amb la participació d'una quarantena d'alumnes de la Universitat.

4.3.1 Emulació

La capa d'emulació afegida a la implementació del middleware permet passar fàcilment (canviant un fitxer de propietats) d'un entorn real a un de virtualitzat: es substitueix el lliurament de missatges per socket per unes cues locals de missatges (veure Fig. 11). El funcionament bàsic de l'emulació és el següent:

- Es carrega un graf de xarxa dinàmic a partir d'un fitxer de text. Aquest fitxer segueix el format Pajek [19], que descriu els nodes de la xarxa i les arestes que enllacen els nodes.
- L'emulador aplica les restriccions de connectivitat en cada pas de simulació. És a dir que si en un moment donat el fitxer indica que entre el node 1 i el node 3 no hi ha connexió directa, si s'envien missatges s'hauran d'encaminar a través de diversos salts.
- La capa d'emulació afegeix tots els paquets unicast de sortida (enviats pel protocol jOLSR) a la cua d'entrada del node destinatari del paquet, sempre que aquest últim tingui connexió directa amb el node font, i sinó es descarten. De la mateixa manera, els paquets broadcast només s'afegeixen a les cues d'entrada d'aquells nodes situats a un salt físic del node font en aquell moment.
- La capa d'emulació també pot simular una taxa de pèrdua de paquets broadcast, una funcionalitat útil per comprovar la convergència dels protocols d'encaminament quan no arriben tots els paquets de control.

Les aplicacions utilitzades per validar els dos protocols generaven tràfic unicast cap a tots els nodes de la xarxa i tràfic multicast per diferents grups multicast. Es van utilitzar diferents topologies variant el nombre de nodes i la densitat de la xarxa, comprovant que el funcionament del protocol era acceptable.

4.3.2 Test real

La motivació principal d'aquest treball ha estat poder fer proves reals de connectivitat sobre MANET en escenaris col·laboratius que podem trobar-nos en el dia a dia: conferències, congressos, reunions, situacions d'emergència, ... Al maig de 2008 es van fer proves reals amb aquest middleware al Campus Sescelades comptant amb la col·laboració d'una quarantena d'estudiants d'Enginyeria Informàtica. Els estudiants portaven un ordinador portàtil amb targeta wireless i Windows XP o Linux. Cada estudiant rebé un número de dorsal que va enganxar al portàtil. El número de dorsal corresponia amb l'últim número de la IP amb que participaven a la xarxa MANET.

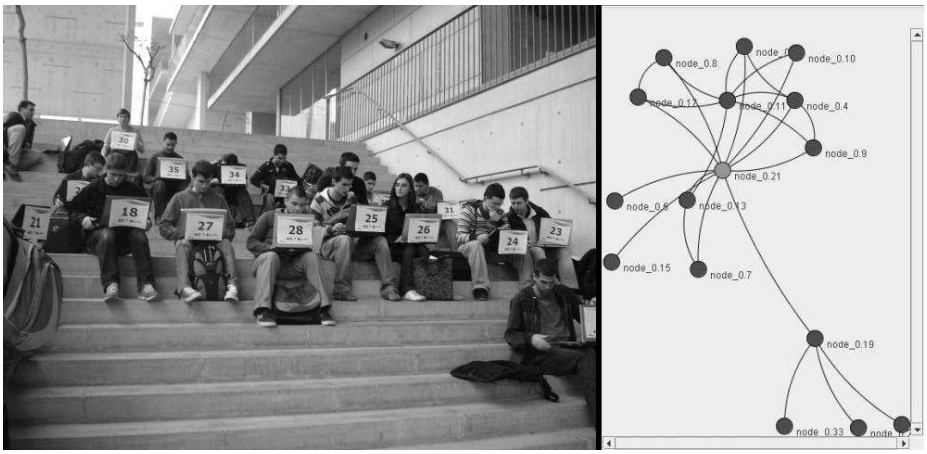

Fig. 12: Fotografia d'un moment de la prova real. A la dreta, es mostra la captura de pantalla d'una de les aplicacions on s'observern els diferents nodes segons la visió local.

Se'ls va distribuir en diversos grups que arribaven a tenir un diàmetre de xarxa Ø = 3. Cada node tenia una aplicació de xat que obria diversos canals i permetia enviar missatges unicast i multicast. A més a més, aquesta aplicació recollia estadístiques de tràfic i mostrava un graf en temps real de la situació de la xarxa. Els detalls de l'aplicació es poden trobar a l'Annex 1, la guia de l'usuari d'OMOLSR.

Com es veu a la Fig. 12, una captura de pantalla d'un moment de la demostració, es van formar grups de fins a tres salts de diàmetre. Vam poder comprovar que quan els grups creixien en nombre d'usuaris per sobre d'un llindar, el rendiment de la xarxa MANET decreixia ràpidament degut a les col·lisions. Tot i així, amb pocs nodes, les comunicacions unicast i multicast funcionaven perfectament. Es pot veure un vídeo [20] d'aquesta demostració a Youtube.

5. Conclusions

En aquest treball de recerca hem presentat dos resultats concrets de la recerca realitzada sobre protocols d'encaminament per MANETs en el marc del projecte POPEYE. Després d'introduir el problema de l'encaminament multicast i les solucions proposades fins a dia d'avui, hem presentat dos protocols especialment dissenyats per a aplicacions col·laboratives en xarxes ad-hoc.

En primer lloc, el protocol OMCAST és una primera aproximació als protocols multicast a nivell d'aplicació. Prenent com a model les xarxes P2P d'Internet, hem construït un overlay sobre un protocol d'encaminament unicast a nivell de xarxa. Tenint en compte la característica broadcast de l'aire, hem proposat un mecanisme d'enviaments broadcast locals, per tal de minimitzar el tràfic de la xarxa i el retard d'extrem a extrem. OMCAST també incorpora un sistema per proporcionar informació de membership a les aplicacions.

Hem validat el mecanisme de broadcast local mitjançant emulacions del protocol real sobre un entorn d'emulació. A més a més, OMCAST s'ha utilitzat posteriorment com a protocol d'encaminament multicast en un middleware per aplicacions col·laboratives [21] dissenyat i implementat al mateix grup de recerca.

En segon lloc, i prenent com a base de treball el protocol OMCAST, s'han dissenyat i implementat els protocols jOLSR i OMOLSR. Anant un pas més enllà, tota la lògica d'encaminament es puja al nivell d'aplicació, permetent que la interacció entre les diverses capes de l'aplicació sigui molt més neta i senzilla. Aquests dos protocols juntament amb el canal multihop de comunicació (MChannel) formen un middleware portable per aplicacions col·laboratives que no necessita de cap més protocol a nivell de sistema operatiu. Aquest middleware es caracteritza per ser molt flexible i adaptable, podent-se adaptar fàcilment a requeriments més específics.

El fet de moure l'encaminament a nivell d'aplicació queda justificat pels avantatges que comporta en simplicitat de desenvolupament, interacció i adaptació. Però a més a més és un model més elegant que les solucions cross-layer, des del punt de vista de l'enginyeria del software.

Els tests reals d'aquest middleware, poc pràctics de realitzar sovint, ens han proporcionat un feedback molt important. El fet d'haver pogut fer proves reals dels protocols desenvolupats en una MANET és una gran oportunitat per comprovar els problemes que comporta establir una xarxa d'aquestes característiques. La majoria de protocols per a MANET no arriben a tenir implementacions reals sinó que es queden en implementacions sobre simuladors. Això passa perquè els investigadors acostumen a

centrar-se en resoldre problemes que no apareixen en les experiències del món real. El fet d'haver fet proves amb quaranta nodes ens ha permès veure les dificultats de les MANETs per convertir-se en xarxes populars i útils per a la gent en diverses situacions de la vida quotidiana.

Amb l'experiència d'aquesta recerca hem identificat diverses línies de treball per crear middleware d'encaminament depenent dels requeriments de diferents escenaris o aplicacions. Com a recerca futura es podria experimentar amb alternatives més simples que jOLSR com ara protocols de gossiping, d'inundació o algoritmes formiga. Dispositius com els telèfons mòbils amb WiFi, que cada vegada són més populars, també es poden beneficiar d'aquests protocols a nivell d'aplicació, ja que no depenen del sistema operatiu ni són intrusius. Desenvolupar middleware per aquests dispositius podria possibilitar la definitiva generalització de les MANETs com a alternativa eficient, útil i viable a les xarxes d'infrastructura.

6. Bibliografia

[1] W.Stallings. *Comunicaciones y Redes de Computadores*, pp. 644-645, Prentice Hall, 2004.

[2] Chakeres, I., Perkins, C.: Dynamic MANET On-demand (DYMO) Routing. Internet draft: draft-ietf-manet-dymo-12.

[3] Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). Request for Comments 3626, October 2003.

[4] C.M. Cordeiro, H. Gossain, and D.P. Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions", IEEE Network, vol. 17, no. 1, 2003, pp. 52-59.

[5] Royer, E., Perkins, C.: Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing. Internet Draft: draft-ietf-manet-maodv-00.

[6] Kunz, T., Cheng, E.: Multicasting in Ad-Hoc Networks: Comparing MAODV and ODMRP. In Proceedings of the Workshop on Ad Hoc Communications, Bonn, Germany, September 2001.

[7] Yi, Y., Lee, S., Su, W., Gerla, M.: On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks. Internet draft: draft-ietf-manet-odmrp-04.

[8] J. Xie, et al., "AMRoute: ad hoc multicast routing protocol", ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks 7 (6) (2002).

[9] M. Ge, S. V. Krishnamurthy and M. Faloutsos. "Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol", Ad Hoc Networks Volume 4, Issue 2, March 2006, 283-300.

[10] M. A. Kaafar, C. Mrabet, T. Turletti, "A Topology-Aware Overlay Multicast Approach for Mobile Ad-Hoc Networks", AINTEC, Bangkok, Thailand, November, 2006.

[11] S. Blödt, "Efficient End System Multicast for Mobile Ad Hoc Networks", in Proc. of PERCOMW 2004, Orlando, Florida, USA, March 2004.

[12] C. Gui and P. Mohapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks," in Proc. IEEE Wireless Comm. and Networking Conf., vol. 2, IEEE Press, 2003, pp. 1118-1123.

[13] L. Xiao, et al., "Prioritized Overlay Multicast in Mobile Ad Hoc Environments", IEEE Computer, vol. 37, no. 2, pp. 67-74, Feb. 2004.

[14] K. Chen and K. Nahrstedt, "Effective location–guided tree construction algorithm for small group multicast in MANET", in Proceedings of IEEE Infocom'02 (2002).

[15] http://apetestbed.sourceforge.net/#APE_Tools The APE testbed

[16] Y. Zhang and W. Li, "An Integrated Environment for Testing Mobile Ad-Hoc Networks", Proceedings of ACM MobiHoc 2002, Lausanne, Switzerland, June 2002.

[17] http://masimum.dif.um.es/?Software:DYMOUM

[18] http://ast-deim.urv.cat/wiki/OMOLSR

[19] Batagelj, V., Mrvar, A.: PAJEK: Program for large network analysis. Connections, 21:47-57, 1998.

[20] http://www.youtube.com, keyword: OMOLSR

[21] Arrufat, M., Paris, G., Garcia Lopez, P., "AGORA: an integrated approach for collaboration in MANETs", Mobilware 2008, Innsbruck, Austria, February 2008.

Annex 1: Guia d'usuari d'OMOLSR

OMOLSR User's Guide 1.0

Grup de recerca en Arquitectura i Serveis Telemàtics (AST)
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
May 2008

Index

1 OMOLSR Basic functioning

OMOLSR is presented as a jar file named test_application.jar. This jar file is auto-executable and starts the class urv.emulator.AppLauncher. This class is the entry point both for the emulator and to run applications in a real environment. The behaviour of this class is specified in the ./conf/omolsr.properties file.
test_application.jar is available at http://ast-deim.urv.cat/popeye/test_application.jar

1.1 Emulator

When the jar file is launched as emulator, the AppLauncher creates as many instances of the application as nodes the emulation graph contains. Messages are not sent through the network, but through a special Emulation layer.

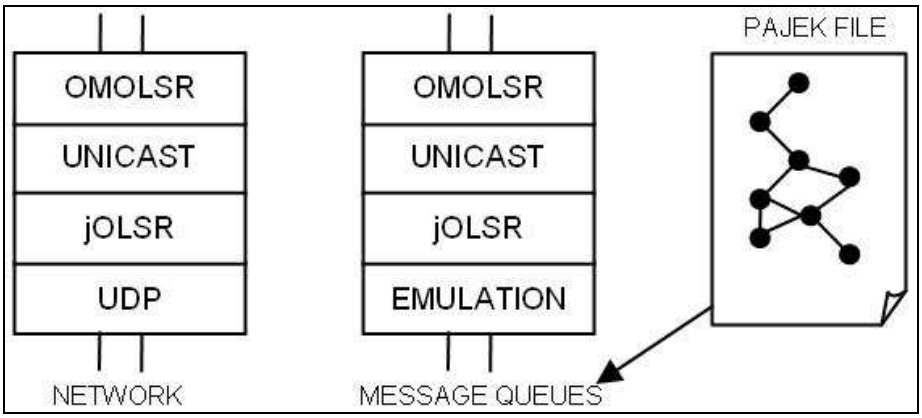

Fig. 1. JGRoups protocol stacks for real applications (left) and emulated applications (right). Messages from real applications are routed through the network, whereas messages from emulated applications are delivered considering the constraints in the graph file.

The emulator accepts task classes that support the emulation and collect statistics. Existing task classes are the following:

- CommunicationStatsTask: This task gathers information about all messages sent and received in the network by all applications. It verifies that all nodes that were in the view of the source node received the multicast message.
- MembershipStatsTask: This task gathers information about the groups created in the applications and the nodes that joined these groups. This information is checked with the view of each MChannel, in order to verify the correct behaviour of getView() method in the channel.
- TopologyChangesTask: This task is in charge of changing the underlying topology as it is defined in the network graph. This task also checks that the Neighbor table of the jOLSR protocol of each node is consistent with the real neighborhood of each node.

1.2 Real applications

When the jar file is launched in real environment, the AppLauncher only creates one instance of the application. Messages are sent using standard UDP sockets. Although tasks cannot be launched in real environment applications, a special flag (COMM_LOG) can be enabled to gather logs about sent and received messages.
There are several applications in the distributed jar file, mainly used for testing purposes. The most complete application is AppTest, that provides unicast and multicast

communication through an easy-to-use GUI. Another useful testing application is the SimpleSenderApplication. Under the package urv.app there are also other applications like MembeshipTest or RealTest.

1.2.1 AppTest

AppTest is a GUI application that was designed to test the protocols in real environments. This application allows the user to join different groups and start chats with all the group members. Chats can use multicast communication so every member in the group can read the messages, or unicast communication, that can only be read by the destination of the message.

Fig. 2 is a screenshot of this application. In the upper side the user can specify the multicast group to join. Once the user joins a group, a new big tab appears. In the left side of this tab there is the group members list and in the right side there is a graph representation of the network as seen by the local node. The local node is colored in green and the other nodes in red. There exists the possibility of changing the layout of the nodes by selecting the desired layout from the list below the graph. It is also possible to zoom in and out the graph representation.

The center panel holds the chat tabs. By double-clicking on a member of the list, a new chat tab for this member appears in the center panel. You can chat with different people of the same group and also in the default multicast chat. Messages sent to the multicast chat are received by all group members. When a chat message is received, the tab corresponding to the sender is highlighted in blue.

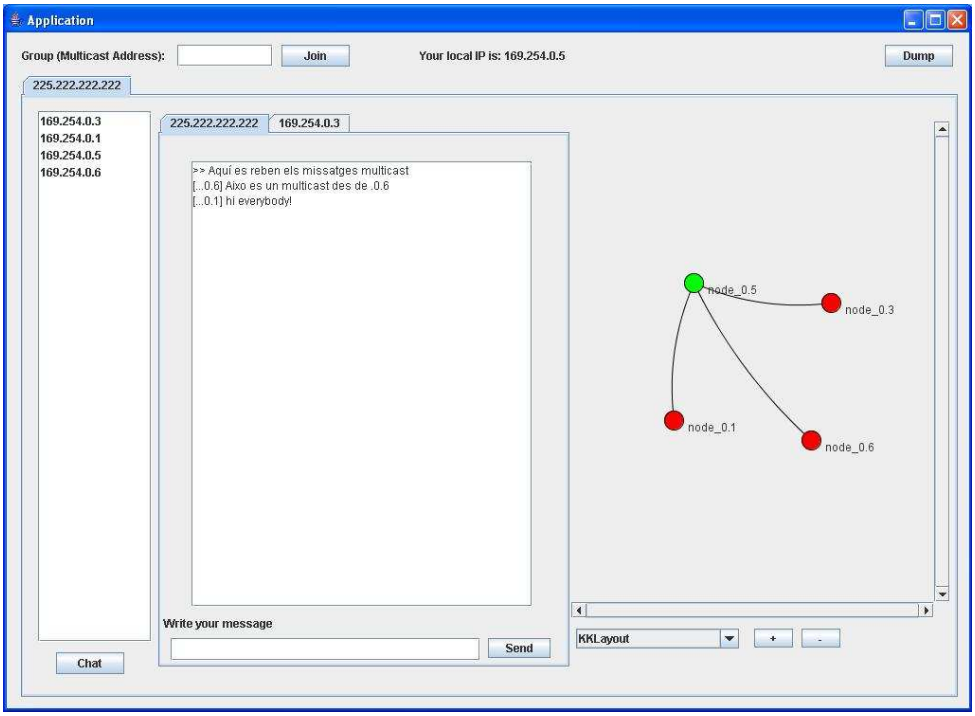

Fig. 2. Screenshot of AppTest application.

A button (Dump) in the top-left corner can be used to show the internal tables of jOLSR in run-time: Routing table, NeighborTable, TopologyTable and MprSet.

1.2.2 SimpleSenderApplication

SimpleSenderApplication was designed to test the protocols. This application basically creates a MChannel and sends multicast messages through it. If launched with the CommunicationStatsTask, you can see how many messages were correctly delivered to the other members of the group.

1.3 Properties file

The Java properties file that defines the behavior of OMOLSR is located at ./conf/omolsr.properties. The function of each property is described here:

- EMULATED: This flag indicates whether to perform network emulation or run a real application.
- COMM_LOG: This flag indicates if communication logging is enabled. This feature is only used for non-emulated environments.
- UNICAST_PORT: The unicast UDP port used in real applications.
- RELIABILITY: This flag indicates whether the protocol stack includes reliability protocols or not.
- MULTICAST_PROTOCOL: The protocol that will be used to send multicast messages. Currently, there are two protocols available: OMOLSR and SMCAST.
- GRAPH_FILE: The pajek file used for the network simulation. One application will be launched for each node in the network graph. The file can be static (with extension .net) or dynamic (with extension .tim). Dynamic graphs involve adding and hiding edges at different time instants.
- APPLICATION: The application class that will be launched, either in emulation or in real environment. Most applications can be used in both environments.
- EMULATION TASK: A list of tasks classes. These tasks are launched to support the emulation.
- SENDING_PROB: jOLSR control messages are sent with this probability. If the probability is 1.0, all messages will be delivered to the destination. If the probability is 0.0, all sent messages are discarded.

The possible values of the properties are listed in the following table:

Property name	**Possible values**
EMULATED	TRUE, FALSE
COMM_LOG	TRUE, FALSE
UNICAST_PORT	A port number > 1024
RELIABILITY	TRUE, FALSE
MULTICAST_PROTOCOL	OMOLSR, SMCAST
GRAPH_FILE	A graph file (extension .net or .tim)
APPLICATION	The fully qualified class name of an application class
EMULATION_TASKS	A comma separated values list of tasks classes
SENDING_PROB	A float value between 0.0 and 1.0

2 Creating new applications

In order to create new applications you must follow these guidelines:

- The application class must extend urv.emulator.core.Application. E.g.:

```
import urv.emulator.core.Application;

public class AppTest extends Application{
     ...
}
```

- The application class must implement the abstract method Application.start(). After the invocation of this method, the new application can start creating MChannels.
- The creation of the MChannel must be done with the method Application.createMChannel(MulticastAddress). Before sending and receiving messages, the application must invoke the method start on the created MChannel. E.g.:

```
MulticastAddress mcastAddr = new MulticastAddress();
mcastAddr.setValue("224.0.0.10");

MChannel mChannel = super.createMChannel(mcastAddr);
mChannel.registerMessageListener("MyApp",...);
mChannel.addMembershipListener("MyApp");
//We must call start before sending and receiving messages
mChannel.start();
```

Annex 2: Guia de desenvolupador d'OMOLSR

OMOLSR Developer's Guide 1.0

Grup de recerca en Arquitectura i Serveis Telemàtics (AST)
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
May 2008

Index

1 General Architecture

This middleware is based in offering a channel to enable multi-hop communication in Mobile Ad Hoc Networks. As depicted in Fig. 1, the multi-hop channel (M-Channel) uses both unicast and multicast functionalities provided by the two underlying protocols: jOLSR and OMOLSR.

In first place, OMOLSR computes multicast routing thanks to the information received via events coming from jOLSR. These events allow OMOLSR to update the OMOLSR Network Graph (ONG), which will be used for routing computation as well as for providing membership information. Since OMOLSR is an application level multicast protocol, it will split a multicast message into multiple unicast messages. In order to do this, messages are sent to jOLSR so they can be routed to the other members of the group. Furthermore, message delivery is provided with unicast reliability, so nearly full delivery ratio is achieved.

Besides, jOLSR stores network information in different tables similarly to OLSR specification: Neighborhood information base (NIB) stores neighbor information; Local Link Information Base (LLIB) keeps updated information about the state of links to the neighbors; Topology Information Base (TIB) maintains information of the network topology to perform routing calculation. Apart from these tables, OMOLSR needs a new table called Multicast Group Table (MGT) which stores information about which nodes are present in each multicast group.

Finally, on the bottom layers, UDP Protocol allows jOLSR to send control (broadcast) and data (unicast) packets.

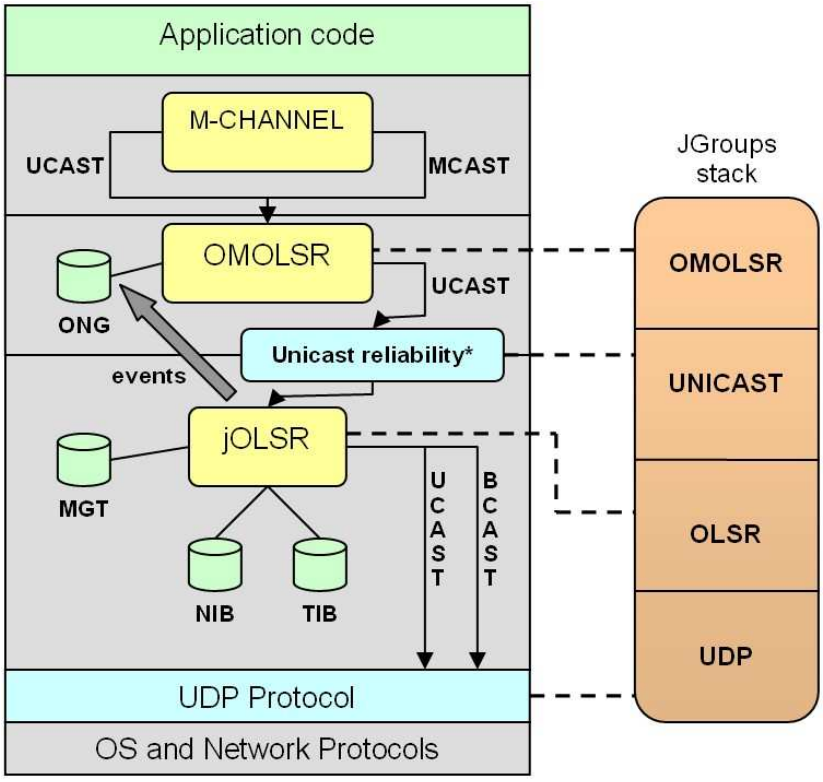

Fig. 1 General Architecture. Each component of the architecture is linked with the corresponding protocol in the JGroups stack

1.1 JGroups

In order to ease and clarify the development of both jOLSR and OMOLSR, we have extended an existing toolkit for reliable communication: JGroups. The key feature of JGroups is its flexible protocol stack, which can be configured and extended depending on the communication needs. Each protocol in the stack provides different functionalities: ordering, reliability, membership, state transfer, etc. In our case, we have implemented both routing protocols (jOLSR and OMOLSR) as JGroups protocols so we can benefit from unicast reliability and ordering by adding the UNICAST protocol to our stack.

Each JGroups channel (JChannel) is initialized with a protocol stack specified in a String or an XML file. The stack is a sequence of protocol names and their optional initialization parameters. For instance, the following stack is used to initialize JChannels in OMOLSR (see urv.conf.ApplicationConfig):

```
"UDP(mcast_send_buf_size=640000;discard_incompatible_packets=true;ucas
t_recv_buf_size=20000000;loopback=false;mcast_recv_buf_size=25000000;m
ax_bundle_size=64000;max_bundle_timeout=30;use_incoming_packet_handler
=true;ucast_send_buf_size=640000;tos=16;port_range=1000;enable_bundlin
g=false;ip_ttl=32;bind_port=5034):OLSR(mcast_addr=224.0.0.10):UNICAST(
timeout=1200,1800,2400,5000,8000;use_gms=false):OMOLSR(mcast_addr=224.
0.0.10)"
```

In OMOLSR, each MChannel is mapped to a JGroups channel.

1.1.1 JGroups dependencies

We have modified some classes of JGroups version 2.5.2 to fit our needs. The source code of JGroups is located under the srcJG252 folder. Modified blocks are noted with a comment in the code (//OMOLSR CHANGE). Anyway, the classes with modified blocks are the following:

- org.jgroups.protocols.UDP: The local address is not obtained from the socket as JGroups does, but is obtained from the protocol stack properties bind_addr and bind_port.
- org.jgroups.protocols.UNICAST: A restriction that does not allow sending messages to non group members is removed.
- org.jgroups.JChannel: User defined events are allowed to be passed up to the MChannel.

2 jOLSR

jOLSR is an application level implementation of the OLSR routing protocol written in JAVA. Although jOLSR includes the basic functionality of OLSR, some modifications have been added to provide topology and group membership information to the upper multicast protocol. For the complete technical specification of OLSR, see RFC3626.
The implementation of jOLSR can be found under the package urv.olsr.

2.1 Data Structures

Data structures used by jOLSR can be found under the package urv.olsr.data. The main data structures are the following:

- NeighborTable: This table contains information about neighbors of the current node and status with all neighbors.
- MprSet: A set that holds information about the multi-point relay nodes (MPRs) of the current node.
- MprSelectorSet: A set that holds information about the nodes that have the current node as MPR.
- NeighborsOfNeighborsSet: This table contains a list of neighbors for each neighbor of the current node.
- TopologyInformationBaseTable: this table stores the information received in TC messages, i.e.: a list of neighbors of each TC message originator.
- RoutingTable: This table defines the routing policy of the node. It defines which is the next intermediate node A that should receive the message when we want to deliver a message to node B.
- MulticastNetworkGraph: This data structure maintains a graph of the network using the information stored in NeighborTable and TopologyTable.
- MulticastGroupsTable: This table contains the list of multicast groups each node has joined.

There are also other auxiliary data structures such as the DuplicateTable, which is used to detect duplicated control messages, or the packet format structures that encapsulate the messages, that will be deeper explained in following sections.

2.1.1 NeighborTable

The NeighborTable is defined in the class urv.olsr.data.neighbour.NeighborTable. Since it is a subclass of ExpiringEntryTable, its entries are removed when the timeout expires. For each node (neighbor of the current node) in the table we store the link status and the neighbors of that node (neighbors of neighbors of the current node).
Whenever the NeighborTable changes (by adding or removing a neighbor or a neighbor of neighbor), a status flag is set to true. This flag is used by other tables that depend on the information stored in this table. After a change takes place, the MPR set must be recomputed as well, and a flag for this action is also set to true.
HELLO messages are created using the information stored in the NeighborTable, and a convenience method is provided to create these control messages.

2.1.2 MprSet

The MprSet is defined in the class urv.olsr.data.mpr.MprSet. This class simply contains a list of the nodes that are chosen as Multi-point Relay nodes.

2.1.3 MprSelectorSet

The MprSelectorSet is defined in the class urv.olsr.data.mpr.MprSelectorSet. This class contains the list of nodes that have chosen the current node as MPR. It also provides a convenience method to create TC messages. Each of these messages include a sequence number that is auto-incremented every time the MPR selector set is changed.

2.1.4 NeighborsOfNeighborsSet

The NeighborsOfNeighborsSet is defined in the class urv.olsr.data.neighbour. NeighborsOfNeighborsSet. The set contains a list of neighbors for each neighbor of the current node. The NeighborTable includes a reference to this set.

2.1.5 TopologyInformationBaseTable

The TopologyInformationBaseTable (or TopologyTable) is defined in the class urv.olsr.data.topology.TopologyInformationBaseTable. Since it is a subclass of ExpiringEntryTable, its entries are removed when the timeout expires.
Entries of this table are indexed by a pair of nodes. A sequence number is used to replace old entries by newer ones.
Whenever the TopologyTable changes (by adding or removing a pair of linked nodes), a status flag is set to true. This flag is used by other tables that depend on the information stored in this table.

2.1.6 RoutingTable

The RoutingTable is defined in the class urv.olsr.data.routing.RoutingTable. The routing table contains the routing information for every node of the network. Each entry of the routing table contains the address of the next hop to reach a specific destination, and the number of hops to the final destination.

2.1.7 MulticastNetworkGraph

The MulticastNetworkGraph is defined in the class urv.olsr.mcast. MulticastNetworkGraph. This class contains a reference to the MulticastGroupsTable as well. The graph is built using the information available in the local Neighbor Table and Topology Table: so the graph is recomputed when changes are detected in Neighbor Table and Topology Table.
This class also provides a method to obtain a contracted graph that only includes the nodes that belong to a specific group. The edges of the contracted graph are tagged with a weight that is the number of physical hops between the vertices of the edge.

2.1.8 MulticastGroupsTable

The MulticastGroupsTable is defined in the class urv.olsr.mcast.MulticastGroupsTable. This class provides methods to register and unregister a node to a specific multicast group, as well as methods to obtain the list of groups a node is registered to.
Whenever the MulticastGroupsTable changes (by adding or removing a multicast group in a node's list), a status flag is set to true. This flag is used by other tables that depend on the information stored in this table.

2.2 jOLSR messages

In order to maintain the routing information up-to-date, each jOLSR node must send control messages to the other nodes of the network. Two types of control message are used in jOLSR: HELLO messages and TC messages.

2.2.1 HELLO messages

A Hello message is periodically broadcasted to the neighbor nodes. It includes a list of neighbors of the local node with the link status of each neighbor. HELLO messages permit a node to know its one-hop and two-hop neighbors, since each node sends information about its local neighborhood.
Hello message is defined in the class urv.olsr.message.HelloMessage. In order to decrease the size of the packets, the message encoding is the following:

- A byte that indicates the number of different linkcodes.
- For each different linkcode, the message includes a byte for the linkcode.
- After the linkcode, the message indicates the size of the list of neighbors which share the same linkcode.
- And finally, the list of neighbors. Each neighbor is represented by its IP address encoded in four bytes.

2.2.2 TC messages

A Topology Control message is periodically broadcasted to all nodes in the network using the multi-point relays (MPRs). This message includes the list of nodes contained in the local MPR selector set, thus topology information of the local node is disseminated to all nodes in the network.
TC message is defined in the class urv.olsr.message.TcMessage. TC Messages perform the same function than in the OLSR specification, but a modification has been added to disseminate information about the multicast groups to all nodes in the network. Therefore, the multicast addresses of the groups joined by the local node are attached in TC messages. This information about the groups each node has joined, is used later to fill the Multicast groups table. Thus, the encoding of TC messages is the following:

- An Advertised Neighbor Sequence Number (ANSN) is associated with the advertised neighbor set.
- The size of the advertised neighbor set.
- The advertised neighbor set: Each neighbor is represented by its IP address encoded in four bytes.
- The size of the list of joined multicast groups.
- The list of joined multicast groups. Each group is represented by its multicast IP address encoded in four bytes.

2.2.3 Message generation

jOLSR needs to send HELLO and TC messages periodically in order to maintain the routing information up-to-date. The class that is in charge of controlling the intervals between message sending is urv.olsr.core.OLSRThread. This class invokes a method on Generator classes that generate a new HELLO or TC message and send it to the neighbors or to all nodes in the network, respectively.
As stated before, HELLO messages are generated in the NeighborTable, since these messages only include information about the local neighborhood.

On the other hand, TC messages are generated in the MprSelectorSet, since these messages include a list of MPR selectors. The information about joined multicast groups is automatically added to the TC messages.

2.2.4 Message handling

When a new message is received, it is processed by different classes and depending on its contents is discarded, delivered to the upper protocols of the stack, forwarded or handled. The overall process is described here:
Messages are received by the OLSR protocol (org.jgroups.protocols.OLSR), which based on the OLSR headers determines whether it is a user data message or a control message. If it is a control message, the OLSRController (urv.olsr.core.OLSRController) is in charge of checking that the message is not a duplicated and its time-to-live is valid. Then, depending on the type of message, the controller invokes either the HelloMessageHandler or the TCMessageHandler.
The HelloMessageHandler (urv.olsr.handlers.HelloMessageHandler) is in charge of updating the data structures taking into account the information contained in the received HELLO message. Data structures that can be modified by the reception of a HELLO message are the NeighborTable, the NeighborsOfNeighborsTable, the MprSet and the MprSelectorSet.
If the received message is a TC message and the current node is a MPR, this message must be forwarded. After checking if the TC message must be forwarded, the message is handled by the class urv.olsr.handlers.TcMessageHandler. This class updates the topology table by removing entries with a sequence number older than the one included in the received message. If the message includes new entries that do not exist in the topology table, these new entries are added to the table.
TcMessageHandler also updates the Multicast Groups Table with the information about the multicast groups joined by the originator of the message.

2.3 jOLSR functioning

jOLSR must maintain tables up-to-date continuously. So, control messages must be sent periodically as well as table information must be removed when timeouts are over. Therefore, jOLSR needs a timing thread that handles these periodic tasks. The implementation of this thread is defined in the class urv.olsr.core.OLSRThread. Tasks assigned to this thread are the following:

- Handle the timeouts of all data structures that extend ExpiringEntryTable.
- Check the flag that indicates whether MprSet should be recomputed.
- Look for changes in NeighborTable, TopologyTable and MulticastGroupsTable (as indicated by their respective change flag). When there are changes in any of these tables, the multicast network graph must be recomputed. If there are changes in the NeighborTable or in the TopologyTable, the RoutingTable must be also recomputed.
- Control intervals between HELLO and TC messages sending.

The algorithm that computes the MprSet is implemented in the class urv.olsr.core.MprComputationController.
Instances and sequence diagram
As aforementioned, JGroups stack is composed by different protocols, one of them being jOLSR. Application messages are passed down the stack and are intercepted by each protocol, which may add a header in each message. Likewise, messages from the network are passed up the stack and protocols may retrieve information from headers of

its own type. One application may start more than one JChannel, the basic structure that gives access to the JGroups protocol stack. Therefore, more than one instance of org.jgroups.protocols.OLSR may be started. However, in order to have a single instance of the protocol, the OLSRController is implemented as a singleton. Since this class creates the data structures and starts the OLSRThread, we ensure that there will be only one instance of the core classes of the protocol.
Each JChannel is mapped to a group, and hence to a multicast address. The org.jgroups.protocols.OLSR object registers itself in the OLSRController using this multicast address. The first OLSR object that registers in the OLSRController will be used for message sending. So, the first created channel is the only one used to send messages to the network. The messages sent through the other channels will be redirected to the first channel at jOLSR level. The first channel of each node will always share the same UDP port. This port is used for all message sent, hence the only channel that receives messages from the network is the first created channel.
Message headers associated to each protocol are used to send and retrieve information about these protocols. OLSR headers provide information about the type of message. There are two possible types: DATA (application data) and CONTROL (Hello messages and TC messages). The multicast address of the group (or channel) is also included in the headers. Control messages are delivered two the core classes of the protocol. On the other hand, data messages addressed to the local node are passed up to the corresponding protocol stack depending on the multicast group indicated in the header.

2.4 Unicast routing

The routing of unicast messages in jOLSR is very simple because routing table is continuously updated. jOLSR routes two kind of messages depending on their origin: unicast messages sent down by upper protocols and unicast messages coming from the lower protocols. The former messages are directly routed but the latter are only routed if their final destination is not the local node.
The method OLSRController.handleOutgoingDataMessage() is in charge of sending the message to the next address obtained from the Routing table. The final destination address is kept in a special OLSR header added to all messages, because receiving nodes must know which the actual destination of a message is.

3 OMOLSR

OMOLSR (Overlay Multicast over OLSR) is a new application level multicast routing protocol, designed to work on top of jOLSR. OMOLSR computes locally minimum spanning trees by benefiting from the topology information gathered by jOLSR. The main characteristic of OMOLSR is that it does not need to send additional control packets to perform multicast delivery. The unicast routing protocol already provides all necessary information.

3.1 Data Structures

Data structures of OMOLSR are minimal because all the topology information is provided by the underlying unicast routing protocol, jOLSR. The class urv.omolsr.data.OMOLSRData gives access to the main OMOLSR data structures:

- omolsrNetworkGraph: This object is a representation of the graph that nodes of the multicast group form.
- mstNetworkGraph: This object is a representation of the minimum spanning tree computed from the omolsrNetworkGraph.
- temporalNodes: An auxiliary table that lists nodes that are temporally out of the group membership.

The class OMOLSRData provides the necessary methods to access these data structures and computing minimum spanning tree.

3.2 OMOLSR functioning

The "entry point" of this protocol is the class org.jgroups.protocols.OMOLSR. This class is a JGroups protocol that is included in the protocol stack of MChannel willing to use multicast communication. Like in jOLSR, one application may start more than one JChannel and therefore, more than one instance of org.jgroups.protocols.OMOLSR may be started. But although jOLSR only has an instance of the core classes of the protocol, in OMOLSR there is an instance of each core class for each multicast group.
OMOLSR receives updates of the network graph via events. Once a new update is received, the minimum spanning tree must be computed again.

In the following sections we will explain the basic operations of OMOLSR: dynamic computation of the Minimum Spanning Tree and routing of multicast packets.

3.2.1 Tree computation

When an update of the network graph is received, OMOLSR computes a minimum spanning tree with the local node as the source of the tree, as depicted in Fig. 2. The tree will then be used for routing multicast packets to all the members of the group.

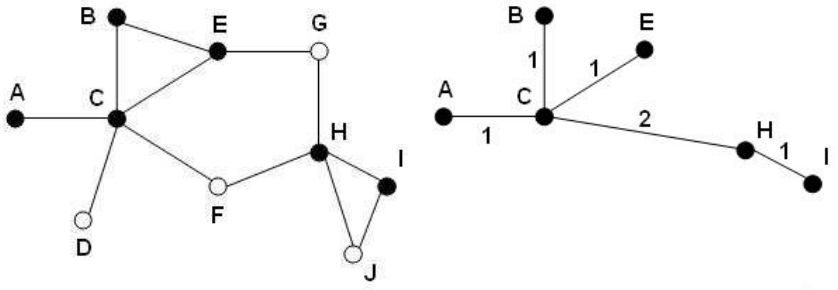

Fig. 2. The graph on the left is the real topology of the network, with the members of the group in black. The graph on the right is the minimum spanning tree for the node A.

3.2.2 Multicast routing

Multicast routing is implemented in the class urv.omolsr.core.StandardHandler. In order to route multicast packets, OMOLSR uses an explicit multi-unicast scheme. When the application generates a new multicast packet, OMOLSR routes the message based on the tree computed for that multicast group. A copy of the packet is sent to each virtual neighbor (a neighbor in the minimum spanning tree), which is responsible for delivering the message to a certain subset of nodes. This subset is defined in each node by using the source-created tree and consists of all the nodes that are in the subtree of each virtual neighbor. This information is then attached to the header of the data packet. When the virtual neighbor receives the message, it computes a tree with the subset of nodes contained in the header. Again, it sends a copy of the data message to its virtual neighbors with new header content. The process is repeated until the subset which must receive the message is empty.

4 Messages and events

The different protocols of a JGroups stack exchange events up and down the stack in order to share information or modify user messages. A message sent by the user's application is itself an event that is passed down through the stack until is sent to the network and is passed up again through the receiver stack.

4.1 Messages

As aforementioned, a message is passed through the stack as a Message Event. At MChannel level we distinguish 2 kinds of outgoing message: multicast message and unicast message. Multicast messages are intercepted by OMOLSR and are split in several unicast packets and passed down. On the other hand, unicast messages are ignored by OMOLSR and are also passed down. So, at jOLSR level, all the packets intercepted are unicast (i.e.: they have a single destination address). Thus, jOLSR simply sends each packet to the next hop to achieve the destination address. It is worth noting that a special protocol header is added to multicast packets intercepted by OMOLSR and unicast packets intercepted by jOLSR.

Received unicast packets at jOLSR level can be unicast packets to forward or, unicast messages arriving at its final destination. However, if these unicast messages have an OMOLSR header, they can also be multicast packets to forward or, multicast messages arriving at their final destination. So, headers are used to mark packets to be intercepted by the protocols.

4.2 Events

OMOLSR uses an event to pass up the network graph from the jOLSR protocol to the OMOLSR protocol and the application layer. This event is encapsulated in the class urv.olsr.mcast.UpdateEvent and is passed up using the USER_DEFINED type of JGroups.

When jOLSR detects changes in the underlying topology or local neighborhood, a new network graph is computed, and a contraction of this graph is passed up to all started channels. Each contracted graph only includes the nodes that are joining the group of the specific channel.

5 Emulator

5.1 Tasks

The emulator accepts task classes that support the emulation and collect statistics.

5.1.1 Topology task

This task, defined in the class urv.emulator.tasks.topology.TopologyChangesTask, is in charge of changing the underlying topology as it is defined in the network graph. This task also checks that the Neighbor table of the jOLSR protocol of each node is consistent with the real neighborhood of each node.

5.1.2 Statistics task

There are two statistics tasks:

- urv.emulator.tasks.stats.CommunicationStatsTask: This task gathers information about all messages sent and received in the network by all applications. It verifies that all nodes that were in the view of the source node received the multicast message.
- urv.emulator.tasks.stats.MembershipStatsTask: This task gathers information about the groups created in the applications and the nodes that joined these groups. This information is checked with the view of each MChannel, in order to verify the correct behavior of getView() method in the channel.

6 Package structure

The implementation classes of jOLSR and OMOLSR are structured under two different top level packages: org.jgroups.protocols and urv.

Under org.jgroups.protocols, we can find the classes that are inserted in the JGroups protocol stack (both protocols: OLSR and OMOLSR). Since these protocols are used in the JGroups protocol stack, they must maintain the package hierarchy specified in the JGroups toolkit. Other classes under this package are the protocol's headers and the SMCAST protocol (a simple multi-unicast protocol that can replace OMOLSR).

Under urv top level package we can find different sub-packages that contain most of the implementation of both protocols. The sub-packages are organized as follows:

- urv.app: This package contains several applications that work on top of the protocols. These applications can be used in emulation and in a real environment.
- urv.conf: This package contains classes that load and store configuration parameters.
- urv.emulator: This package contains the classes that are used in an emulated environment to test applications and protocols.
    - urv.emulator.core: Core classes of the emulator
    - urv.emulator.tasks: This packa
    - urv.emulator.topology:
- urv.olsr: This package contains the main classes of the jOLSR implementation.
    - urv.olsr.core: Core classes of jOLSR: controller, timing thread, table computation.
    - urv.olsr.data: This package holds all data structures used in the protocol.
    - urv.olsr.handlers: this package contains the classes that handle jOLSR control messages.
    - urv.olsr.mcast: This package includes the functionalities added to jOLSR to support upper layer multicast protocols.
    - urv.olsr.message: This package contains the implementation of messages and packet formats used by jOLSR.
    - urv.olsr.util: Utility functions to handle jOLSR data structures.
- urv.omolsr: This package contains the main classes of the OMOLSR implementation.
    - urv.omolsr.core: Core classes of OMOLSR: controller and message handlers.
    - urv.omolsr.data: This package holds all data structures used in the OMOLSR protocol.
    - urv.omolsr.util: Utility functions to handle OMOLSR data structures.
- urv.resources: This package is used to access resources such as GUI images.
- urv.util: This package contains utility classes to operate with dates and graphs.

Annex 3: Article presentat a l'International Conference on Networking, ICN 2008

An Application Layer Multicast for Collaborative Scenarios: the OMCAST protocol

Gerard París, Marcel Arrufat, Pedro García López, Marc Sánchez-Artigas
Dep. d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili
Avinguda dels Països Catalans, 26 - 43007 Tarragona, Spain
{gerard.paris, marcel.arrufat, pedro.garcia, marc.sanchez}@urv.cat

Abstract

New collaborative applications relying on a MANET need multicast to perform efficient group communication. Application Layer Multicast is seen as a good alternative to the traditional network layer routing protocols because of its flexibility and ease of deployment. However, existing application layer multicast protocols for MANETs are not designed for collaborative scenarios. In this paper, we present OMCAST, a new overlay multicast protocol for mobile ad hoc networks that is specifically targeted for collaborative scenarios. OMCAST benefits from the broadcast nature of the wireless medium in order to minimize network delay and congestion. Evaluation shows that OMCAST achieves better network performance than current message delivery approaches.

1. Introduction

Mobile Ad Hoc networks are formed by heterogeneous devices that communicate without any existing infrastructure. Ease of deployment and decentralized administration have turned MANETs in an attractive scenario for building different kind of applications like management in emergency/disaster situations, battlefield coordination and many others. Among these, collaborative working environments (CWE) face a promising future due to proliferation of handheld and mobile devices together with the need of overcoming the limitations of current collaboration schemes. Decentralization, self-adjusting necessities and dynamic behavior must be satisfied in order to achieve efficient collaboration mechanisms in the MANET environment.

Collaboration scenarios are characterized by having a moderate number of nodes located in a small area. Normally, they are located in a room or in a large hall, where some of them may remain static for long periods of time. Some nodes may change its location from time to time in order to interact with other existing groups. However, most communication is performed in well-defined areas, where nodes are located at most two or three hops from the most distant node. Besides, most scenarios and especially collaborative working environments depend highly on group communication. One-to-one and one-to-many communication primitives are essential for the development of these applications. In the past years, several alternatives for group communication in MANETs have been studied, such as broadcast [10], multicast [6], and even geocast [9]. From these candidates, multicast seems the most suitable solution since it makes no assumptions about devices' location and, at the same time, is more efficient than message broadcasting for most scenarios.

There are two existing approaches for multicast message delivery. In first place, most of the proposed multicast protocols like MAODV or ODMRP follow network layer approaches. These protocols tackle bandwidth, quality of service and other issues for multicast communication. However, none of these protocols have mature and widely tested implementations. The second alternative for enabling multicast communication is application layer multicast (overlay multicast) in which multicast packets are encapsulated in unicast datagrams and delivered to all group members. In application multicast only group members need to keep state information. Moreover, application layer multicast provides ease of deployment as well as capability of hiding underlying link errors. Hence, an overlay multicast protocol could be easily adapted to collaborative applications requirements.

Whereas numerous network layer protocols can be found in the literature [6], not many overlay multicast protocols have been considered. Current proposals, ALMA and PAST-DM, are more flexible than network protocols and easier to bring into real collaborative environments. Although they are considered as the best performing protocols, they present several constraints when applied to collaboration scenarios. For instance, PAST-DM presents a rigid bootstrap process which forces all members of the multicast session to start simultaneously. Besides, ALMA proposes a complex loop avoidance mechanism as well as the existence of a rendezvous host for obtaining member and tree information. Moreover, and what it is more important for collaboration scenarios, none of both approaches take benefit from broadcast nature of the medium. We believe that in MANET collaboration scenarios, like a scientific conference or a meeting, exploiting broadcast capabilities is the key to obtain higher communication performance.

Therefore, we present OMCAST, an application layer multicast specially designed for collaborative scenarios. OMCAST was designed to fit the need of a flexible multicast routing protocol in the scope of the EU-funded POPEYE project. This two-year project addresses spontaneous peer-to-peer collaboration architectures over mobile ad hoc networks. The lack of mature and widely tested network layer multicast protocols arose the need of an application layer multicast protocol that could adapt to different unicast protocols. Thus, main features provided by OMCAST are the following:

- Flexible bootstrapping process, where nodes may join at any time.
- Broadcasting of data packets to 1-hop neighbors to reduce communication overhead.
- Decentralized membership information available for higher level layers

In the following section, we outline the different application level multicast approaches proposed in the literature. In section 3, we describe the concepts about OMCAST behavior. After this, we present an evaluation of the multicast protocol under different scenarios. To conclude, future and current work on real implementations is explained.

2. Related Work

In this section we will review most important application layer multicast and analyze its main features. Since OMCAST is inspired by PAST-DM, we will explain this protocol in greater detail.

AMRoute [5] was the first overlay multicast protocol proposed for MANETs. This protocol creates a shared tree for data distribution using only group members as nodes. The shared tree is built, from a virtual mesh, with unicast tunnels that allow connection between group members. One of the main disadvantages of AMRoute is the static behavior of the virtual mesh, since no changes are made in the structure once it has been built. AMRoute does not handle network dynamics and leaves all responsibility for the underlying unicast routing protocol.

ALMA [1] (Application Layer Multicast Algorithm) creates a tree of logical links between the group members. The aim of this protocol is to reduce the cost of each link in the tree by reconfiguring the tree under mobility and congestion situations. When a node joins the network it must select a node as a parent, so as to become part of the tree. If tree performance drops below a defined threshold, the node must reconfigure the tree by switching the parent or freeing children. This mechanism leads to a complex loop avoiding and detection system, since synchronous switching can occur. ALMA also considers the existence of a rendezvous host for obtaining the structure of the logical tree as well as neighbor information in the bootstrapping process.

AOMP [3] (Ad-hoc Overlay Multicast Protocol) is an application-layer multicast that relies on reactive routing protocols to construct a delivery tree in a dynamic and decentralized way. This protocol proposes two stages: a first one that connects new nodes to the overlay and a second one that performs the tree construction and maintenance. AOMP takes advantage of the unicast routing protocol, and thus avoids routing overhead and improves scalability. However, this protocol is limited to use reactive protocols like AODV or DSR and only considers a single source node for the multicast session.

NICE-MAN [4] presents several improvements from the existing internet NICE protocol by exploiting the broadcast capability of the medium to reduce network traffic. The basic improvement is the maintenance of a reduced overlay formed just by cluster leaders whereas the rest of members are located one-hop away from at least one cluster leader. Thus, a node may send a message to various nodes simultaneously by benefiting from the broadcast nature of the medium. However, there are several drawbacks like the continuous selection of cluster leaders. Furthermore, non-overlay nodes are loosely connected since they do not send any control messages. This may imply high message loss,

as nodes need to recover from the loss of connectivity. In consequence, membership information is not available either.

PAST-DM [2] (Progressively Adaptive Subtree in Dynamic Mesh) is an overlay multicast protocol based on the construction of a dynamic virtual mesh. The mesh is maintained dynamically through the exchange of link state packets, thus adapting to network topology changes. These packets provide link state table information, that is, a partial view of the network. All nodes need to start the multicast session simultaneously, and afterwards initiate the bootstrapping process by sending TTL-bounded broadcast messages. With the topology information extracted from the mesh, nodes compute a source-based Steiner tree to deliver information to all members in the multicast group. Logical and physical hop distances are used as heuristics to compute the Steiner tree. The source node takes its logical (virtual) neighbors as children in the tree. The rest of the nodes are packed into subgroups, which form a subtree where the root of this tree is one of the logical neighbors. Thus, each child of the source tree is responsible for delivering the multicast message to all nodes in the subtree. This process is repeated through every node until the subtree becomes empty. The decision of packet delivery path is computed at each receiver, so path selection is performed always with the most up-to-date information. Although this is an efficient way of delivering data, some packets may be lost if nodes change location, once the source node has computed its corresponding subtree.

Other research works intend to minimize the latency of the overlay and improve the overall efficiency taking into account other characteristics. POM [7] is focused on prioritized overlays, and LGT [8] considers the geometrical location of nodes as a heuristic to improve the tree formation procedure.

3. OMCAST protocol description

In this section, we will explain in greater detail OMCAST functionalities. OMCAST follows the basic algorithm of PAST-DM, so it relies on link state info exchange and locally computed Steiner trees. A virtual mesh connects all members in the multicast group, which is built dynamically by periodically exchanging link state information. Furthermore, OMCAST provides modifications that improve overall performance for collaborative communication. So the main differences with PAST-DM are that OMCAST uses a more flexible bootstrap mechanism and takes advantage of local broadcast transmission for data delivery in dense areas.

3.1. Basic operation

OMCAST presents a flexible bootstrap procedure, which allows multicast members to join at any time. Any source or receiver that wants to enter the multicast session must send a join message. This join message consists in a local broadcast message, which is successively forwarded by non-member nodes. When the message reaches a group member, a unicast message is sent to the source as a reply. This reply message contains local link state information as well as the number of hops the node is located from the replier. Thus, the bootstrapping node automatically receives information about the current neighbors in the group, and it is ready to send multicast messages to any node.

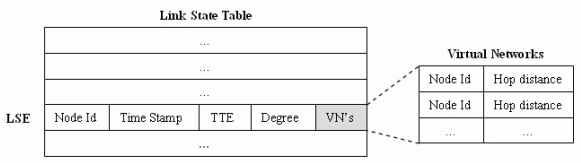

Figure 1. Link State Table structure

The mesh maintenance in OMCAST is similar to the one used by PAST-DM, but some changes were necessary in order to improve the whole process. One link state table (LST), depicted in Figure 1, is needed to store information for a single multicast group in each node. The most recent entries of this structure are periodically exchanged with the node virtual neighbors, that is, the members of the multicast group located at one logical hop from the local node.

Then, each link state table creates a link state entry (LSE) for each member of the current multicast group. Each LSE stores information about the link state of the corresponding member: node degree, virtual neighbors; together with the node identifier and the entry time to expire. Lower values on time to expire indicate that the information kept is less accurate than information with higher values. When this value reaches a certain threshold, the entry is marked as invalid. When updated versions of the entries arrive from a node, time to expire is set to the highest value and the entry is marked again as valid.

By maintaining this link state information, nodes can compute source-based Steiner trees and deliver data to all members. Furthermore, membership information can be provided to upper layers, which is in fact very useful in collaborative scenarios. Providing

this information just implies offering a list of node identifiers retrieved from each link state entry.

When a node wants to send a message to all members, it computes the source-based Steiner tree, and sends a copy of the data packet to each virtual neighbor (one-level children). Each virtual neighbor is responsible for delivering the message to a certain subgroup of nodes, which will be defined by the source-created tree. Then, together with the packet content, headers are sent with information about the subgroup of nodes that must still receive the message. The virtual neighbor receives the message, and computes the tree with the received subgroup of nodes. Again, sends a copy of the message to its virtual neighbors with new header content. The process is repeated until the subgroup which must receive the message is empty.

3.2. Local broadcast delivery

In other protocols like PAST-DM, when a node wants to send or forward a message through the generated tree, it first checks its state information. Then, the node generates as many copies as members must receive the message. One different unicast packet with the copied message is sent to each member. In OMCAST we propose a bandwidth-saving technique that consists in sending the message via local broadcast capability. If there are enough virtual neighbors located at one physical hop willing to receive the message, the message will be sent just once as a broadcast message. Thus, bandwidth is saved whenever one of these nodes wants to send a message to members located just one hop away. Figure 2 shows the standard mechanism to send one multicast message to each receiver, compared to the local broadcast approach. As we can see in the first approach, the m packet is copied and sent 4 times through the network. On the other hand, in the second approach the message is just sent once to the medium, so nodes can hear the packet and avoid repeated delivery of messages.

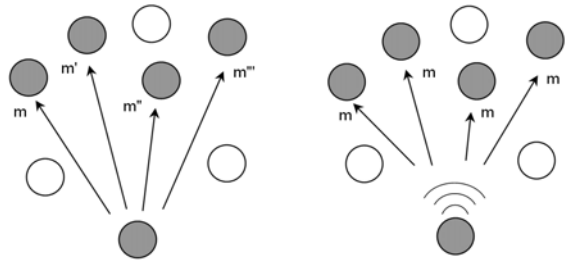

Figure 2. Standard vs. Local broadcast delivery

It must be considered that, even taking as a basis a collaboration scenario, nodes may change its location unexpectedly. Therefore, it is possible that packets sent via local broadcast do not reach their final destination due to mobility issues, e.g. virtual neighbor information is temporally inaccurate. In order to minimize this loss, broadcast delivery will be performed only under certain conditions:

- The number of virtual neighbors that are ready to receive the message must be greater than a threshold MIN_BC_NEIGH.
- A virtual neighbor is considered to be ready to receive broadcast messages if the source node has received one link state message from this neighbor in the last BC_PERIOD seconds. In this way, we ensure that a node will send broadcast messages just to nodes that were located at one physical hop in a close period of time. Thus, we prevent sending messages to nodes that have potentially changed its location.

Another problem caused by node mobility occurs when a node is supposed to forward a packet to a receiver that does not appear in its tree. This may happen since source node is responsible for assigning first subgroup nodes. OMCAST solves this problem by forcing forwarder nodes to unicast these packets to the destination whenever the intended receiver does not appear in the tree.

4. Performance evaluation

We have evaluated OMCAST performance by completing numerous emulations targeted to realistic collaboration environments. Therefore, we assume a scenario of 30 nodes, where group size varies between 5, 10, 15 and 20 members. Furthermore, in order to approximate the emulation to a real environment we have developed an emulation environment consisting of VMware images together with an improved version of MobiEmu [11]. APE Mackill [1] has been used together with MobiEmu to emulate connectivity. In the emulation environment we use an actual implementation of DYMO protocol, named DYMOUM.

Since we aim to demonstrate OMCAST performance in a real collaborative environment, we cannot take the random waypoint as mobility model. To achieve this, we have specially designed several scenarios where nodes are located in three non-connected areas. In each scenario, there exist one or two nodes linking the first with the second area, and the second with the third area, so connectivity is provided for all current members. Emulation lasts for 140 seconds, where nodes may switch from one area to

another and remain static for a pause time of 30 seconds. Members of the multicast group send packets at a constant bit rate of 4 Kb/s if not specified.

4.1. Impact of broadcast delivery

In first place we evaluate the benefit obtained from using broadcast delivery for data messages. The emulation compares the network traffic induced by group members' multicast messages, considering on the one hand unicast delivery and, on the other hand broadcast delivery mechanisms. With this simulation, we aim to prove the efficiency of broadcast delivery in areas with collocated nodes where multicast is performed.

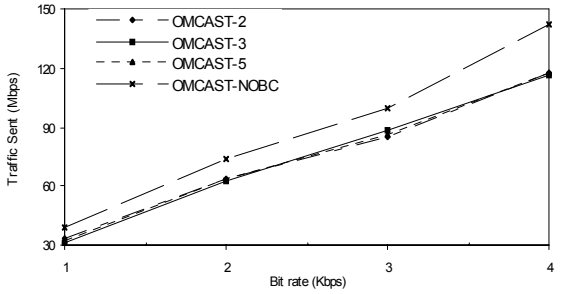

Figure 3 Overall network traffic

We have chosen a scenario of 20 group members, which will send messages at different bit rates. The MIN_BC_NEIGH parameter is changed to restrict the use of broadcast delivery at different values. This implies that broadcast messages will only be sent if there are at least MIN_BC_NEIGH virtual neighbors ready to receive the message. OMCAST-2, OMCAST-3, OMCAST-5 indicate that the current value of MIN_BC_NEIGH is set to 2, 3 and 5, respectively. On the other hand, OMCAST-NOBC indicates no use of local broadcast delivery, that is, all messages are sent via unicast.

As we can see in Figure 3, overall network traffic has been reduced significantly. We observe that by taking benefit from broadcast medium capability almost 20 percent traffic is reduced when local broadcast delivery is used. This can be explained since one message sent to the network is taken by multiple receivers; therefore, fewer packets are induced in the network. The extra traffic created by non using broadcast may produce longer end-to-end delay, as well as an increase of packet loss, as it will be shown in the next figure. We can also see that the three variants that use local broadcast perform in a similar way, since most members have more than 5 nodes in the device's range in the selected scenarios.

4.2. Data Packet Delivery Ratio

The second evaluation is focused on the influence of broadcast transmission in packet delivery ratio. It must be noted that unicast messages are specifically addressed to a destination node, so that unicast routing is able to route the packet to its destination. In contrast, broadcast messages are addressed to all nodes located at one physical hop distance, so it is only delivered to the nodes that are in the source node range. Therefore, a node may not receive a broadcast message that it is supposed to receive, due to inconsistencies in virtual neighbor information. However, as seen in the previous figure, the overall network traffic is reduced using broadcast retransmission, so the congestion may decrease assuring a high packet delivery ratio.

Taking the same parameters of the previous simulation, Figure 4 shows that OMCAST with capability of transmitting broadcast data packets performs almost a 15% better than the non-broadcast-enabled version.

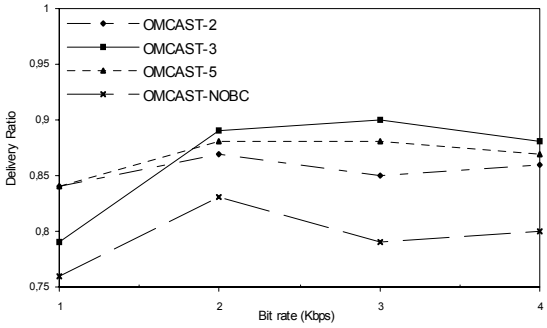

Figure 4 Packet Delivery Ratio

4.3. Round trip time delay

The last figure shows the effect of broadcast retransmission in end-to-end delay. As we have seen, network traffic was substantially reduced when enabling broadcast forwarding in dense environments. Therefore, delay should be reduced as well. In this figure we evaluate the delay decrease for different group sizes.

The delay is computed by sending several multicast request messages with a timestamp. Nodes receiving these messages reply with a unicast ACK message to the source. This message contains the original

timestamp sent in the request, so sources can establish the round-trip-time to all nodes. The global mean value is computed and shown in Figure 5.

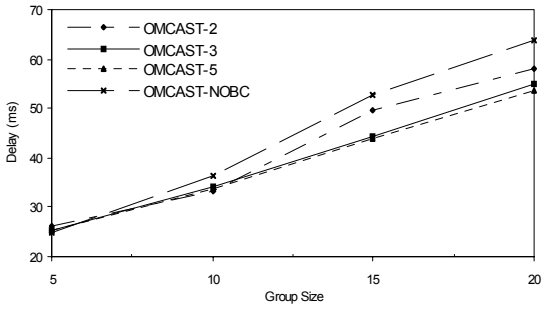

Figure 5 Round Trip Time

As we can see, Round trip time is higher when using non-broadcast delivery. Broadcast approaches present lower delay, which can be explained by the reduction of traffic in the network. Thus, we can conclude that by using local broadcast delivery, OMCAST ensures higher data packet delivery ratio and reduces notably global delay.

5. Conclusion

In this paper, we have introduced OMCAST, an overlay multicast for mobile ad hoc networks (MANET) especially targeted to collaborative scenarios. OMCAST benefits from broadcast medium capability in order to reduce global traffic load. Therefore, it minimizes packet loss and reduces end-to-end delay. OMCAST also offers membership information which higher level layers may use in collaborative applications.

Evaluation shows how OMCAST performs well in these scenarios, where nodes are located close together in different areas, which are connected by one or two nodes. We have seen how end-to-end delay as well as traffic induced in the network is reduced for data packet delivery.

We are currently using OMCAST in a collaborative middleware developed for ad hoc networks. We are performing real tests with a 15-node testbed, with nodes located in different rooms. Nodes periodically switch areas in order to test stability in mobility situations. Chat and shared photo album applications have been tested and performed in a satisfactory manner. Future work is focused on minimizing size of control packets by enhancing link state table exchange mechanism, together with multiple multicast group improvements.

6. Acknowledgements

This work is partly supported by the European Commission, under the FP6-2006-IST-034241 POPEYE project.

7. References

[1] M. Ge, S. V. Krishnamurthy and M. Faloutsos. “Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol”, Ad Hoc Networks Volume 4, Issue 2, March 2006, 283-300.

[2] C. Gui and P. Mohapatra, “Efficient Overlay Multicast for Mobile Ad Hoc Networks,” in Proc. IEEE Wireless Comm. and Networking Conf., vol. 2, IEEE Press, 2003, pp. 1118-1123.

[3] M. A. Kaafar, C. Mrabet, T. Turletti, “A Topology-Aware Overlay Multicast Approach for Mobile Ad-Hoc Networks”, AINTEC, Bangkok, Thailand, November, 2006.

[4] S. Blödt, “Efficient End System Multicast for Mobile Ad Hoc Networks”, in Proc. of PERCOMW 2004, Orlando, Florida, USA, March 2004.

[5] J. Xie, et al., “AMRoute: ad hoc multicast routing protocol”, ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks 7 (6) (2002).

[6] C.M. Cordeiro, H. Gossain, and D.P. Agrawal, “Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions”, IEEE Network, vol. 17, no. 1, 2003, pp. 52-59.

[7] L. Xiao, et al., “Prioritized Overlay Multicast in Mobile Ad Hoc Environments”, IEEE Computer, vol. 37, no. 2, pp. 67-74, Feb. 2004.

[8] K. Chen and K. Nahrstedt, “Effective location–guided tree construction algorithm for small group multicast in MANET”, in Proceedings of IEEE Infocom’02 (2002).

[9] P. Yao, E. Krohne, and T. Camp, “Performance Comparison of Geocast Routing Protocols for a MANET”, Proceedings of IC3N 2004, pp. 213-220, 2004.

[10] B. Williams, T. Camp, “Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks”, In Proceedings of ACM MobiHoc 2002, Lausanne, Switzerland, June 2002.

[11] Y. Zhang and W. Li, “An Integrated Environment for Testing Mobile Ad-Hoc Networks”, Proceedings of ACM MobiHoc 2002, Lausanne, Switzerland, June 2002.

[12] http://apetestbed.sourceforge.net/#APE_Tools The APE testbed

Annex 4: Article enviat a l'International Middleware Conference, MIDDLEWARE 2008

OMOLSR: improving MANET middleware with application layer routing protocols

Pedro Garcia Lopez[1], Marcel Arrufat[1], Gerard Paris[1] and Marc Sánchez Artigas[1]

[1] Departament d'Enginyeria Informàtica i Matemàtiques,
Universitat Rovira i Virgili, Av. dels Països Catalans 26,
43007 Tarragona, Spain
{pedro.garcia, marcel.arrufat, gerard.paris, marc.sanchez}@urv.cat

Abstract. We believe that any MANET middleware should be aware of the underlying multi-hop routing protocol to improve communication efficiency. In general, existing MANET middleware either ignore the underlying routing protocol or create specific cross-layer solutions that break the strict layering of the network stack. Like many peer-to-peer protocols, we propose to move the routing logic to the application layer. In this line, we have implemented the OLSR protocol in the application layer using UDP (jOLSR). On top of it, we present OMOLSR, an overlay Multicast protocol that directly benefits from the OLSR protocol to improve communication efficiency. We validate our approach both in simulation and in a real test-bed to demonstrate the feasibility and efficiency of our MANET middleware.

Keywords: MANET, middleware, routing, application layer multicast.

1 Introduction

Mobile Ad-Hoc Networks (MANETs) represent an interesting substrate for many types of applications that do not require a fixed network infrastructure (Access Points). When every participant is a routing node in the network, several interesting multi-hop scenarios may arise. For example, spontaneous collaborative applications are of particular interest for settings where many users can interact with close-by participants (conferences, campus, stadium, popular events, city, games). In these settings, it is very important for MANET middleware and applications to be aware of the underlying routing protocols and topology. If the middleware just uses the underlying transport protocol as a black box, it can incur in communication inefficiencies due to the multi-hop nature of the medium.

A typical solution has been to implement cross-layer solutions enabling the construction of more efficient middleware (using topology information). However, we believe that this approach is flawed because it implies dirty ad-hoc hacks that break the clean network layering model. These approaches solve specific problems but are difficult to port or adapt to different scenarios.

We believe that in this setting is completely justified to move the routing logic to the application layer. In fact, peer-to-peer technologies are a classic example of

moving routing logic to the application layer. In this line, content distribution networks or application layer multicast are peer-to-peer technologies created to overcome the limitations of the transport layer. In the case of MANET settings, there exist several reasons to move the routing logic to the application layer:

- Clean interaction between middleware and routing protocols.
- Flexibility and adaptation to specific requirements.
- Simplicity of development, testing and deployment.
- Topology-Aware overlay

In the context of the European project IST-POPEYE (Peer to Peer Collaborative Working Environments over Mobile AdHoc Networks), we have developed a Java implementation of the OLSR protocol using UDP sockets (jOLSR). To address the group communication issues of the collaborative middleware, we have created OMOLSR, an overlay multicast channel that directly benefits from the OLSR routing protocol. We will demonstrate by simulation and in a real test-bed that this multicast protocol is efficient, adaptive and fault tolerant. Furthermore, thanks to the tight integration with OLSR at the application layer, the protocol outperforms others like SMOLSR or MOLSR in terms of communication efficiency.

The structure of this paper is as follows: in section 2 we present related work, in section 3 we describe POPEYE's OMOLSR and jOLSR middleware architecture. In section 4 we present prospective applications of our middleware. In section 5 we present validation results using both simulation and experimentation, and finally we draw some conclusions in section 6.

2 Related work

It is widely accepted in MANET settings that cross-layer solutions are required for achieving performance and adaptation in many scenarios [1, 2, 4]. Nevertheless, many researchers also recognize that strict layering enables controlled interaction among layers (each layer is independent developed and maintained). Besides, many cross-layer solutions generate spaghetti-like code that is impossible to maintain efficiently because unexpected dependencies between layers may arise.

To avoid this coupled cross-layer solutions, authors in [1] proposed a vertical component named Network Status that would avoid nasty inter-layer communication. All the layers (MAC, Network, Transport, Middleware and Application) would then communicate directly with Network status and thus avoiding direct coupling between layers. This design is more elegant than the classical cross-layer solutions but still face important problems. On the one hand, it is not a standardized operative system component so it is difficult to port and maintain. The community should agree to such transversal solution with an open standard. This is clearly difficult to achieve in the short term. On the other hand, network status seems appropriate for offering information between layers, but not as a clear medium for upcalls and downcalls. In specific settings, a tight coupling between some layers can be even desirable to achieve optimal performance.

If we focus in the middleware and application layers, it is also obvious that cross-layer solutions clearly present better performance. Both [10] and [11] modify existing

peer-to-peer middleware (FreePastry, Gnutella) to benefit from a cross-layer communication with underlying MANET protocols. It is quite clear that if middleware ignores the underlying topology it can even hurt or degrade the overall performance of other nodes. In conclusion, we can summarize the existing approaches to cross-layer in the following groups: including information in packet headers, using the Internet Control Message Protocol, offering device or kernel ad-hoc connections, creating transversal components for all layers [1], standardizing interfaces and loopback addresses [7], or even allowing direct signaling among layers [8]. Even some projects have considered a complete rethinking of the network stack for MANET environments [9].

As we will explain in the following sections, our approach for MANET middleware is simpler: moving the routing logic to the application layer. This is probably not a solution for all kind of problems and we neither aim to replace existing transport layer MANET protocols. But we will demonstrate that this simple solution is very appropriate for many scenarios where cross-layer information flows are required by middleware and applications.

Before entering into deeper details we will introduce briefly some of the current contributions on MANET routing

Unicast Routing Protocols

DYMO (Dynamic MANET On-Demand) [16] is a reactive unicast protocol, successor of the popular AODV (Ad-Hoc On-Demand Distance Vector) and shares many of its functionalities. Routes are created on-demand by sending request and response control packets. In consequence, no global topology information is available. On the other hand, this means that when nodes stop sending messages, there is no overhead traffic in the network. DYMO seems more suitable for sparse communications, with mobility but may not work as well in case of congestion.

OLSR [12] is a proactive unicast protocol, so it maintains routing table information up to date continuously. Topology information is exchanged by means of controlled flooding of topology messages. Hello messages provide information about the two-hop neighborhood in a way that each node selects a neighbor as MPR (Multi-Point Relay). These MPRs are in charge of sending topology messages to the entire network performing controlled flooding. With the topology information, each node can build the routing table in order to be able to send messages to the other nodes. OLSR performs well in small-medium sized networks where node density is relatively high. The knowledge of the topology, together with its good performance with dense communication patterns turns OLSR in a good candidate for performing group communication in MANETs.

Application Level Multicast Protocols

ALMA (Application Layer Multicast Algorithm) [17] creates a tree of logical links between the group members. The aim of this protocol is to reduce the cost of each

link in the tree by reconfiguring the tree under mobility and congestion situations. When a node joins the network it must select a node as a parent, so as to become part of the tree. If tree performance drops below a defined threshold, the node must reconfigure the tree by switching the parent or freeing children. This mechanism leads to a complex loop avoiding and detection system, since synchronous switching can occur. ALMA also considers the existence of a rendezvous host for obtaining the structure of the logical tree as well as neighbor information in the bootstrapping process.

PAST-DM (Progressively Adaptive Subtree in Dynamic Mesh) [18] is an overlay multicast protocol based on the construction of a dynamic virtual mesh. The mesh is maintained dynamically through the exchange of link state packets, thus adapting to network topology changes. These packets provide link state table information, that is, a partial view of the network. With the topology information extracted from the mesh, nodes compute a source-based Steiner tree to deliver information to all members in the multicast group. Logical and physical hop distances are used as heuristics to compute the Steiner tree. Each child of the source tree is responsible for delivering the multicast message to all nodes in the subtree. This process is repeated through every node until the subtree becomes empty. The decision of packet delivery path is computed at each receiver, so path selection is performed always with the most up-to-date information. Although this is an efficient way of delivering data, some packets may be lost if nodes change location, once the source node has computed its corresponding subtree.

In conclusion, both protocols, as most application level protocols, need to send control packets to keep their structures up to date: using periodic exchange of link packets or communicating with other nodes in the tree. In both cases, the protocols ignore the underlying network routing protocol so they perform redundant communication and, in consequence, bandwidth misuse.

3 Advantages and drawbacks of Application-layer MANET routing

The idea of moving routing logic to the application layer is not new. The peer-to-peer paradigm is frequently used to overcome the limitations of the transport layer. For example, Application Layer Multicast (ALM) solutions have emerged due to the problems with IP multicast in the Internet. ALM solutions create overlay topologies that permit efficient one-to-many communications using unicast transport communications. One of the major issues with these overlay networks is to correctly reflect the underlying topology to be efficient in terms of latency. In this line, the routing latency between two nodes on the overlay network can be different from the unicast latency between those two nodes on the underlying network. The ratio of these two quantities is called the latency stretch. If this ratio is high, there is a

considerable penalty in using the overlay. In fact, it is still an open problem in peer-to-peer research to create topology-aware overlays.

Hereafter, we will present the main advantages of our approach:

Topology-Aware overlay. In MANET environments, nodes are connected in a peer-to-peer network created in the transport layer using protocols like OLSR, AODV or DYMO. If we just move these protocols to the application layer (OLSR for example), the topology does not change. The application layer peer-to-peer network will then reflect exactly the physical connections. Furthermore, the application layer protocol will route packets using UDP unicast connections in the same way that the transport protocol would operate. This is a remarkable advantage of MANET application layer routing and avoids many problems found on Internet topology-unaware overlays.

Clean interaction between middleware and routing protocols. When MANET routing is just another middleware component we can use well-established software engineering practices to specify module interaction. With standardized APIs in the application layer it is not necessary to break the strict layering of the network stack for MANET middleware.

Flexibility and adaptation to specific requirements. Different applications may have completely different requirements for the routing layer and underlying topology. If it is a matter of changing a software component, it is feasible to develop specific adaptive routing layers for different settings. Emergency scenarios, military applications or collaborative work settings may develop specific underlying routing components adapted to their particular scenarios.

Simplicity of development, testing and deployment. It is easier to develop and test routing protocols in the application layer than in the transport layer. The middleware can be created transparent from the operative system and thus simplifying kernel compilation, library dependencies and hardware settings. Furthermore, the key point is easy deployment and portability to different platforms. Mobile phones could easily install an application that creates its own MANET adapted to its specific requirements. It is far more difficult to motivate users to install new transport protocols.

On the contrary, we find these major drawbacks:

Specific routing service. At the time present, our approach does not offer a generic service for all the applications running in a OS (like transport protocols do). Only the application using this middleware will benefit from the MANET routing services. If we have for example existing videoconferencing or desktop sharing tools using standard transport protocols, they wont be able to benefit from our middleware. In any case, this problem could be overcome in the same way that peer-to-peer Virtual Private Networks (VPNs) are working today. If necessary, the middleware component (P2P VPN) could then even provide standard transport level services to legacy applications.

Lower Layer interconnection. We do not address the problem of interconnecting lower layers like Physical, MAC, network and transport. In those cases it could be necessary that the community agreed on a standardized component or interlayer protocols. In the case of our middleware, it would be also possible to access specific OS, device or driver APIs to improve the performance of the middleware. Of course, this would make the middleware less portable and transparent, but it could be necessary for specific solutions.

4 Middleware Architecture: jOLSR and OMOLSR

This middleware is based in offering a channel to enable multi-hop communication. As depicted in Fig. 1, the multi-hop channel (M-Channel) uses both unicast and multicast functionalities provided by the two underlying protocols: jOLSR and OMOLSR. In first place, OMOLSR computes multicast routing thanks to the information received via events coming from jOLSR. These events allow OMOLSR to update the OMOLSR Network Graph (ONG), which will be used for routing computation as well as for providing membership information.

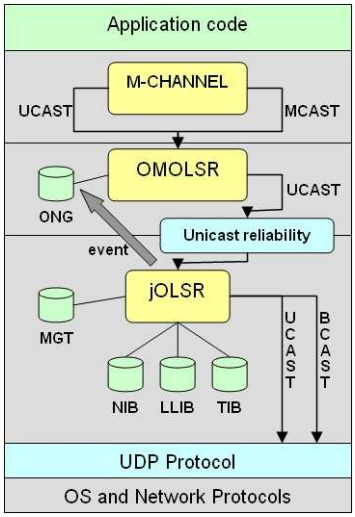

Fig. 1. Middleware architecture

Since OMOLSR is an application level multicast protocol, it will split a multicast message into multiple unicast messages. In order to do this, messages are sent to jOLSR so they can be routed to the other members of the group. Furthermore, message delivery is provided with unicast reliability, so nearly full delivery ratio is achieved.

Besides, jOLSR stores network information in different tables similarly to OLSR specification: Neighborhood information base (NIB) stores neighbor information; Local Link Information Base (LLIB) keeps updated information about the state of links to the neighbors; Topology Information Base (TIB) maintains information of the network topology to perform routing calculation. Apart from these tables, OMOLSR needs a new table called Multicast Group Table (MGT) which stores information about which nodes are present in each multicast group.

Finally, on the bottom layers, UDP Protocol allows jOLSR to send control (broadcast) and data (unicast) packets.

4.1 jOLSR

jOLSR is an application level implementation of the OLSR routing protocol [12] written in Java. Of course, the goal of jOLSR is not to be a fully compliant implementation of the standard, but an implementation which follows its basic functionalities. jOLSR implements nearly all components of the core functionality of OLSR. Although the core functionality also includes support for multiple interface addresses, this feature is not provided in the current version of jOLSR in order to simplify the implementation.

Some modifications have been added to the basic specification of OLSR to provide topology and group membership information to the upper multicast protocol. The multicast protocol, OMOLSR, will benefit from this information in order to avoid

flooding the network with unnecessary control packets. The main changes are the following:

- TC messages now include a list of multicast groups that are joined by the sending node.
- A new table called Multicast Groups Table stores the information about multicast groups received in TC messages. With this information, a partial representation of the network, similar to a graph, is computed when a change is detected in the different tables.

TC Message Modification. OLSR and jOLSR send two different types of control messages, HELLO and TC (Topology Control) messages. HELLO messages permit a node to know its one-hop and two-hop neighbors, since each node sends information about its local neighborhood. Based on this information, the node can select its multipoint relays (MPR) which will be in charge of performing controlled flooding. TC Messages are sent to all the nodes in the network thanks to this controlled flooding mechanism, and disseminate topology information of the local node to all nodes in the network.

In both OLSR and jOLSR, TC messages are sent periodically from one node to the rest of the network, so all nodes can compute its topology table. We have realized that by adding little information in TC messages, we can disseminate information easily to all nodes in the network. This is in fact really useful since the multicast routing protocol needs information about the multicast groups joined by each node.

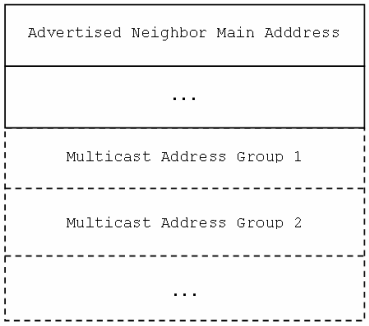

Fig. 2. TC message modification

Therefore, we propose to attach the multicast address of the groups joined by the local node in TC messages, as we can see in Fig. 2. The attached information about the groups is retrieved from the multicast group table.

Multicast Group Table. The Multicast Group Table keeps the information about the multicast groups joined by each node. This information is updated when the local node decides to join a new group or when it receives a new TC Message.

The table keeps a set of the joined multicast groups for each node in the network. The information in this table is used by the upper multicast routing protocol and changes in the table are reported as membership events to the application. When a change is detected in the neighbor table, in the topology table or in the multicast

groups table, a graph containing the members of the group is computed. The multicast protocol will receive a new event with the information of this graph.

In order to get the graph with the members of the group, in first place we obtain an approximate representation of the network by creating a network graph from the information stored in the topology table. Then we check in the Multicast Group Table which nodes belong to which group, so we create an event for each different group in the table. The graph can then be used by the multicast protocol and is able to provide membership information.

4.2 OMOLSR

OMOLSR (Overlay Multicast over OLSR) is a new application level multicast routing protocol, designed to work on top of jOLSR. OMOLSR computes locally minimum spanning trees by benefiting from the topology information gathered by jOLSR. The main characteristic of OMOLSR is that it does not need to send additional control packets to perform multicast delivery. The unicast routing protocol already provides all necessary information.

Basic Operations. The basic operations of OMOLSR are the dynamic computation of the Minimum Spanning Tree and the routing of multicast packets. In first place, OMOLSR computes a virtual mesh that connects all members of the multicast group. This mesh is a contraction of the network graph that jOLSR generates from the Neighbor Table and the Topology Table. By using this contracted graph, which contains only the members of the group, the local node can easily know which members are located at one hop in the membership. These nodes located at one logical hop are also known as virtual neighbors of the local node. The procedure of computing the contraction of the graph is performed for each multicast group the local node is member of.

Then, once the graph is ready, we compute a minimum spanning tree with the local node as the source of the tree, as depicted in Fig. 3. The tree will now be used for routing the packet to all the members of the group.

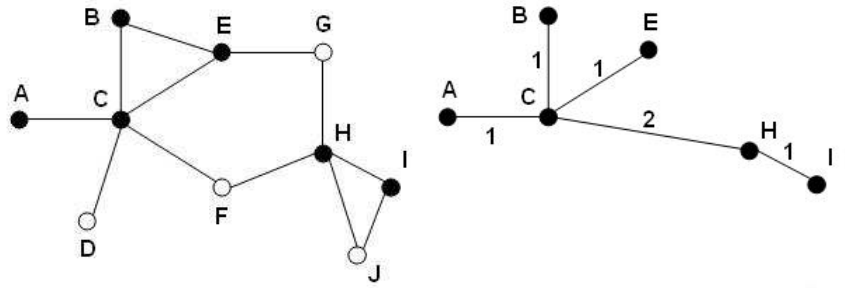

Fig. 3. The spanning tree for the node A. Colored nodes belong to the multicast group. Edges have a weight that measures the number of physical hops in a logical link

Multicast routing. In order to route multicast packets, OMOLSR uses an explicit multi-unicast scheme. When the application generates a new multicast packet, OMOLSR routes the message based on the tree computed for that multicast group. A copy of the packet is sent to each virtual neighbor, which is responsible for delivering the message to a certain subset of nodes. This subset is defined in each node by using the source-created tree and consists of all the nodes that are in the subtree of each virtual neighbor. This information is then attached to the header of the data packet. When the virtual neighbor receives the message, it computes a tree with the subset of nodes contained in the header. Again, it sends a copy of the data message to its virtual neighbors with new header content. The process is repeated until the subset which must receive the message is empty.

Reliability and integration. In order to ease and clarify the development of both jOLSR and OMOLSR, we have extended an existing toolkit for reliable communication: JGroups. The key feature of JGroups is its flexible protocol stack, which can be configured and extended depending on the communication needs. Each protocol in the stack provides different functionalities: ordering, reliability, membership, state transfer, etc. In our case, we have implemented both routing protocols (jOLSR and OMOLSR) as JGroups protocols so we can benefit from unicast reliability and ordering by adding the UNICAST protocol to our stack. This unicast reliability layer uses an acknowledgement scheme to provide lossless transmission of unicast messages. Therefore, multicast reliability is also ensured by adding this layer between both protocols: OMOLSR splits each multicast packet in several unicast packets that will be sent under the acknowledgement scheme.

The flexibility of the stack is also useful for validation purposes. For instance, we have also changed the lowest JGroups protocol that creates UDP sockets for another one that virtualizes communication so we could easily create an emulation layer.

4.3 MChannel

On top of the routing protocols we have developed a channel which enables communication in the mobile ad-hoc network: the MChannel. The main characteristic of the MChannel is that allows the user to send messages to a single member or to all the members in the group even if they are not in range. In consequence, a MChannel is bound to a single group, so if we want to communicate in two groups, we should create two different channels. Furthermore, the channel provides other services through a well-defined API:

- Get the list of members in the group: the channel permits registering listeners to know which users joined or left the group.
- Get the nodes located at 1-hop (neighbors) at 2-hop or at N-hop
- Get a well connected neighbor: from the information retrieved from OLSR, the MChannel can provide a list of neighbors which have many neighbors and give access to many other members in the neighbor.

Apart from these functionalities, the MChannel could offer more complex services in order to build MANET applications: quality of service (QoS) considering multiple parameters at a time (bandwidth, battery, CPU,...) to perform routing, hence providing adaptive middleware; scoped multicast delivery enabling TTL parameters related to MANET routing hops. In this way, multicast messages could be restricted to a certain groups of (closer) nodes; and MANET anycast services enabling filtering of events in nodes.

We believe that the definition of these services could help to build a standard common API for MANET routing components. Therefore, applications could benefit from the same functionalities just by defining APIs for upcalls and downcalls.

5 Validation

The work presented in this paper is not an architectural draft of a middleware for communication in MANETs, but a real implementation of this middleware. Thus, our objective is not to show performance simulations of the application layer routing protocols showing improvements over other protocols. Our main goal is to provide a ready-to-use middleware to build group applications in MANET networks. Nevertheless, we need to validate our approach both in emulation and real environments, to verify that our middleware is able to be the basis of group applications over ad-hoc networks.

5.1 Emulation

As aforementioned, an emulation transport layer was implemented in order to validate our approach. We replace the UDP layer of the JGroups protocol stack for a virtualized layer: socket message delivery is replaced by local message queues. It is worth saying that we used the final implementation of the middleware with only a change in the transport layer, so we could validate the middleware under emulated and real environments without changing a line of code.

The emulator works as following:

1. It loads a dynamic network graph from a text file. This file follows the Pajek format [15], describing the nodes of the network and the edges that link the nodes.
2. The emulator applies the connectivity constraints in each simulation step.
3. Then, the emulation layer simply adds all outgoing unicast packets (from the jOLSR protocol) to the receiver queue of the destination node of the message if this node has a direct connection with the source node. Similarly, broadcast packets are delivered to the queues of all nodes that have a direct connection with the source node in that moment.

The emulation layer can also simulate the loss of broadcast packets, a useful feature to test the convergence of the routing protocols when broadcast control packets are not received

We tested both protocols (jOLSR and OMOLSR) in different topologies, varying the number of nodes and the density of the network. The applications used in the validation generate unicast traffic to all the nodes in the network and multicast traffic for different multicast groups. We could see that the performance was acceptable and the data delivery ratio was almost the optimum thanks to the unicast retransmission layer.

5.2 Real test-bed

As pointed out in [14], there is a serious lack of real-world experiences in MANET research. Usually, researchers focus in solving problems that are not observed in real world experiences. However, since our middleware is intended to be used by real applications, we believe that a real-world test should be done to verify the performance of the routing protocols.

In order to test our middleware, 40 computer science students were told to participate in a MANET test with their laptops (Fig. 4). They were distributed in several groups along the campus, establishing different ad-hoc networks of maximum diameter Ø = 3. Each node run a test application that was able to join a chat with the other members of the group. The application kept track of all sent and received packets as well as periodical pictures of the underlying topology graph. The test involved heterogeneous devices with different O.S. (Windows XP, Vista and Linux) and different wireless cards. Test results were successful, showing that both unicast and multicast traffic was sent and received correctly. In our tests we achieved that 20 laptops participated in the same multicast group with a network diameter of 3 hops.

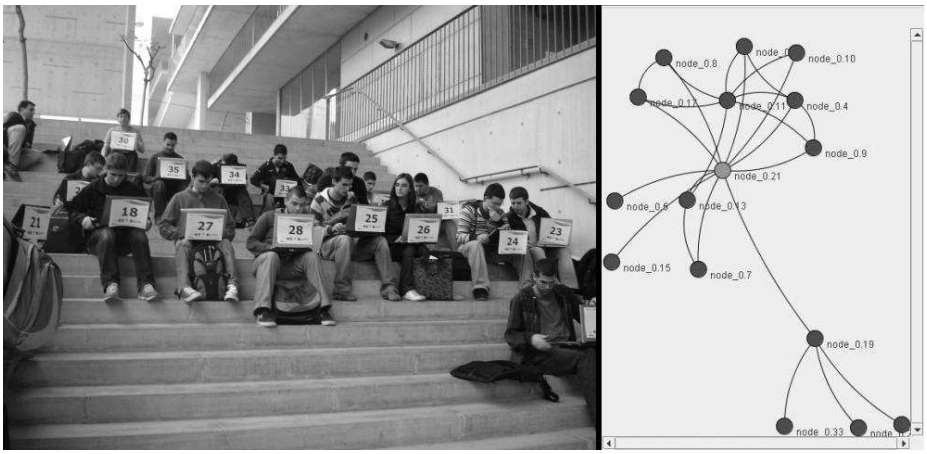

Fig. 4. Picture of the real test and screenshot of the network graph as seen by one of the participant nodes.

5.3 Comparison between MOLSR and OMOLSR

First of all, it is clear that application level multicast approaches are far from being efficient since they completely ignore underlying unicast routing protocols. This implies that they generate redundant traffic in order to create and maintain the necessary structures for multicast delivery. By moving the routing protocols to the application layer, we state that our approach clearly outperforms all current application level multicast protocols, which rely on the underlying unicast routing protocol. In consequence, we believe that the best existing approaches to analyze are the integrated multicast protocols over OLSR.

Secondly, most of the network multicast routing protocols for MANETs are also totally decoupled from the underlying unicast routing protocols, consequently flooding the network with new topology discovery packets. On the other hand, other protocols such as SMOLSR and MOLSR are implemented as extensions of a unicast protocol in order to take advantage of the already available information. SMOLSR (Simple Multicast OLSR) is a straightforward multicast protocol that simply floods the network using the OLSR Multipoint Relays (MPR). Since this protocol does not build any tree, all the nodes receive all the messages even if they are not in the multicast group. Hence, SMOLSR is not efficient for multi-group networks although it does not send any control message to the network. MOLSR (Multicast extension for the Optimized Link State Routing protocol) [13] is a multicast routing protocol for MANETs designed to work in top of OLSR nodes. This protocol shares similarities with OMOLSR, because it provides multicast routing benefiting from the topology knowledge of OLSR. Nevertheless, there are significant differences between them due to the fact of being a network layer multicast routing protocol. The table 1 shows some compared aspects of both protocols.

Table 1. Comparison between MOLSR and OMOLSR.

	MOLSR	OMOLSR
Layer	Network	Application
Underlying unicast protocol	OLSR	jOLSR
MAC layer forwarding	Broadcast	Unicast
Control messages	YES	NO
Routing headers	NO	YES
Stateless multicast	NO	YES
Use of unicast protocol information	YES	YES
Modifications in OLSR messages	NO	YES

The main characteristics of OMOLSR compared with MOLSR are the following:

- Quick multicast tree readjustment: When a node leaves the ad-hoc network abruptly, other nodes no longer receive HELLO messages from the leaving node. Then, OLSR neighbor table of nearby nodes is updated and the leaving node is removed. However, while in OMOLSR the network graph is locally recomputed given this change, in MOLSR new control messages must be generated to fix the tree.
- Unicast reliability at MAC layer: In traditional multicast routing protocols for MANETs like MOLSR, multicast packets are generally forwarded as 1-hop broadcast packets. Whereas MAC layer can detect if a node has received a unicast transmission by means of ACK messages, it is unable to detect if some nodes have not received a broadcast packet. This effect causes the loss of multicast packets in its complete downstream tree, and hence increases the retransmissions in the upper layers. In contrast, OMOLSR multicast forwarding is entirely based in unicast transmissions, that are more reliable at MAC layer, since if no ACK message is received, the packet can be retransmitted again.
- Stateless multicast: Application level multicast protocols like OMOLSR are stateless, i.e. do not need to hold state information in each node of the network. Stateless multicast reduces the control signaling need to maintain this information up-to-date. On the contrary, MOLSR has to maintain updated information about the forwarding branch for each pair (multicast group, source node).
- The overhead introduced in data packets may be an issue to take into account, since each data packet carries little routing information. This problem could be easily solved by providing a mixed approach between stateless multicast and stateful multicast: OMOLSR could keep track of intensive multicast traffic coming from a specific pair (source node, multicast group) and forward most of these packets without routing headers.

After evaluating exhaustively our approach against other approaches, we believe that application level routing together with the different provided services (ttl-multicast, neighbor and topology information, etc.) turn our middleware into a powerful tool for developing applications for mobile networks. Furthermore, this is provided without binding the application to a specific platform or any other transport protocol.

6 Conclusions

The overall idea of our middleware is to move the routing logic to the application layer in MANET environments. In such settings, there exist clear advantages that justify our approach like: clean interaction between middleware and routing protocols, flexibility and adaptation to specific requirements, simplicity of development, testing and deployment, and finally the creation of a topology aware overlay. Apart from these clear contributions, we present a more elegant model for MANET middleware applications than the cross-layer solutions that break the strict layering of the network stack.

We present a working proof of concept of our approach in the Java implementation of the OLSR protocol (jOLSR). On top of jOLSR, we also present an overlay multicast protocol (OMOLSR) that cleanly integrates with jOLSR to improve communication efficiency. We tested our middleware by emulation and in a real testbed (40 campus users).

We foresee more work in this line to create specific routing middleware depending on application requirements and scenarios. As future work, we are also experimenting with flooding, gossiping and ant algorithms to create simpler routing protocols than OLSR for collaborative applications. We also believe that mobile devices will clearly benefit from such application layer approaches that are neither intrusive nor dependent with the underlying mobile device OS. Every application will then be able to construct a MANET network adapted to its communication requirements.

Both OMOLSR and jOLSR source code is freely available for download at http://ast-deim.urv.cat/wiki/OMOLSR.

Acknowledgments. This work is partly supported by the European Commission, under the FP6-2006-IST-034241 POPEYE project.

7 References

1. Conti, M., Maselli, G., Turi, G., Giordano, S.: Cross-Layering in Mobile Ad Hoc Network Design. IEEE Computer Vol. 37, Num. 2, pp 48-51, Feb. 2004.
2. Goldsmith, A.J., Wicker, S.B.: Design Challenges for Energy-Constrained Ad Hoc Wireless Networks. IEEE Wireless Comm., vol. 9, no. 4, 2002, pp. 8-27.
3. Kawadia, V., Kumar, P.R.: A cautionary perspective on cross-layer design. Wireless Communications, IEEE Volume 12, Issue 1, Feb. 2005.
4. Lin, X., Shroff, N.B., Srikant, R.: A Tutorial on Cross-Layer Optimization in Wireless Networks. IEEE Journal on Selected Areas in Communications. Aug. 2006. Volume: 24.
5. Zhou, B. et al.: A Cross-Layer Route Discovery Framework for Mobile Ad Hoc Networks. EURASIP Journal on Wireless Communications and Networking, 2005.
6. Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. Communications Magazine, IEEE Volume 43. 2005.

7. Stine, J.A.: Cross-Layer Design of MANETs: The Only Option. Military Communications Conference, 2006. MILCOM 2006, pp. 1-7. 23-25 Oct. 2006.
8. Wang, Q., Abu-Rgheff, M.A.: Cross-layer signalling for next generation wireless systems. In: IEEE Wireless Commun. And Net. Conf., 2003, pp. 1084 -1089.
9. BAA05-42, PIP for DARPA ATO, Control-Based Mobile Ad-Hoc Networking Program.
10. Delmastro, F.: From Pastry to CrossROAD: CROSS-layer Ring Overlay for AD hoc networks. Conf. on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops).
11. Turi, G., Conti, M., Gregori, E.: A Cross Layer Optimization of Gnutella for Mobile Ad hoc Networks. Proc. ACM MobiHoc Symposium, Urbana-Champain, May 2005.
12. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). Request for Comments 3626, October 2003.
13. Jacquet P., et al.: Multicast Optimized Link State Routing. Internet-Draft, draft-ietf-manet-olsr-molsr-01.txt, November 2001.
14. Gunningberg, P., Lundgren, H., Nordstrom, E., Tschudin, C.: Lessons from Experimental MANET Research. Ad Hoc Networks Journal, special issue on "Ad Hoc Networking for Pervasive Systems", Vol. 3, Number 2, March 2005.
15. Batagelj, V., Mrvar, A.: PAJEK: Program for large network analysis. Connections, 21:47-57, 1998.
16. Chakeres, I., Perkins, C.: Dynamic MANET On-demand (DYMO) Routing. Internet draft: draft-ietf-manet-dymo-12.
17. Ge, M., Krishnamurthy, S.V., Faloutsos, M.: Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol. Elsevier Ad Hoc Networks Journal, vol. 4, no. 2, pp. 283-300, 2006.
18. Gui, C., Mohapatra, P.: Efficient overlay multicast for mobile ad hoc networks. In: The Wireless Communications and Networking Conference (WCNC), New Orleans, Louisiana, USA, Mar. 2003