Una rete Multi-hop Ad-Hoc anonima e sicura.

Roberto Rossi

Abstract

I sistemi P2P hanno subito uno sviluppo e una diffusione impressionante a partire dagli ultimi anni ’90 fino ad oggi. È fatto noto che una percentuale altissima del traffico internet è generato proprio da questi sistemi. Spesso viene naturale associare il concetto di P2P con quello di filesharing, in realtà bisognerebbe distinguere attentamente le due cose. Con P2P intendiamo semplicemente un modello di comunicazione punto – punto che non si appoggia a server e realizza una rete completamente decentralizzata e fortemente dinamica; Internet stesso, a mio avviso, se vogliamo è P2P: nasce P2P, proprio con lo scopo di essere una rete in grado di resistere ad attacchi militari e di realizzare quindi un sistema completamente decentralizzato. Nel corso degli anni sono nati sistemi, comunemente denominati P2P, che di P2P hanno solo il nome. Pensiamo a Napster, il P2P per antonomasia, tutto è fuorchè P2P, giacchè si appoggia su server centrali. Le sue sorti hanno ben mostrato ciò: un P2P, proprio perchè è P2P, non è e non può essere soggetto a controllo, Napster invece lo è attualmente, è controllato e i servizi vengono offerti a pagamento. I successori, Gnutella e OpenNap, realizzano invece qualcosa che a tutti gli effetti può essere chiamato P2P. Il cosiddetto P2P di seconda generazione non si appoggia effettivamente a server, ma presenta reti fortemente decentralizzate. Questi sistemi tipicamente realizzano un motore per la ricerca di risorse e sfruttano collegamenti diretti tra le sorgenti e i fruitori per lo scambio di informazioni, comprendiamo bene i limiti reali di questo approccio, internet è una realtà variegata fatta di reti e sottoreti che spesso non possono comunicare direttamente, una soluzione a ciò è data dal nuovo IPv6, che tuttavia non è attualmente in uso su Internet e non può offrire aiuto concreto.
Un modello nuovo di rete che sta prendendo piede dunque è quello che trasforma i singoli nodi delle reti P2P in veri e propri router capaci non solo di instradare le query, ma anche le risorse stesse. Con l’avvento di reti di comunicazione sempre più veloci e con l’assenza di vincoli di banda, sistemi come il JXTA di java o progetti come Anthill del Dipartimento di Informatica dell’Università di Bologna sono destinati al successo, visto che con il loro modello permettono di creare reti pervasive insensibili a barriere come NAT e router, in cui i nodi possono facilmente trovarsi e scambiarsi informazioni in qualsiasi situazione; reti dunque realmente decentralizzate, che realizzano infrastrutture al di sopra del TCP/IP e sulle quali, in modo trasparente, possono appoggiarsi le applicazioni (si pensi alle JXTASocket).
ANts si pone sulla scia di questi ultimi sistemi: realizza una rete Multi-Hop Ad-Hoc sfruttando algoritmi di intradamento tipici delle reti MANET. Le scelte adottate garantiscono alti livelli di privacy e sicurezza nelle comunicazioni. Nel seguito analizzeremo le politiche di comunicazione e gli algoritmi di routing adottati nello sviluppo del sistema, i suoi pregi, i difetti e le estensioni possibili.

Perchè un modello MANET ?

Una rete mobile multi-hop da-hoc è costituita da un gruppo di nodi che comunicano via radio e non hanno bisogno di alcuna infrastruttura. Sono reti flessibili, e hanno un gran numero di applicazioni pratiche, dato che permettono di stabilire comunicazioni senza bisogno di alcuna infrastruttura preinstallata. Dato che i dispositivi mobili hanno un raggio di azione limitato spesso le comunicazioni devono essere instradate da dispositivi intermedi, dunque ogni dispositivo funge anche da router. Punti critici in queste reti sono dunque la ricerca di nodi nella rete e il mantenimento di un canale di comunicazione tra i nodi a fronte di variazioni nella rete stessa.
Quale dunque il nesso con la rete ANts?
La rete ANts si pone come una struttura costruita al top dei protocolli internet TCP/IP. In ANts i nodi devono poter comunicare tra loro stabilendo (poche) connessioni iniziali dalle quali sarà possibile raggiungere l'intera rete. Le connessioni tra nodi avverranno sfruttando connessioni TCP e gli indrizzi IP, ma all'interno della rete ANts i nodi si troveranno tra loro sfruttando un'altro identificativo (ID), non necessariamente univoco. È interesse dei nodi mantenere segreto questo ID non permettendo ai nodi vicini (connessi via TCP) di poter associare l'indirizzo IP che loro vedono al relativo ID. Capiamo bene che in tal modo, dato che ogni nodo è sia agente, sia router, è possibile raggiungere un anonimato pressochè completo aggiungendo due livelli di crittografia: connessioni TCP criptate tra i singoli nodi e connessioni criptate tra gli endpoint che stanno comunicando (fig. 1); queste ultime giustificano il fatto che non sia una necessità stringente avere ID univoci in rete, in quanto il raro caso di un messaggio giunto all'ID sbagliato si risolverà in un impossibilità da parte del nodo di leggere il messaggio stesso, dunque (vedremo) ci sarà una ritrasmissione.

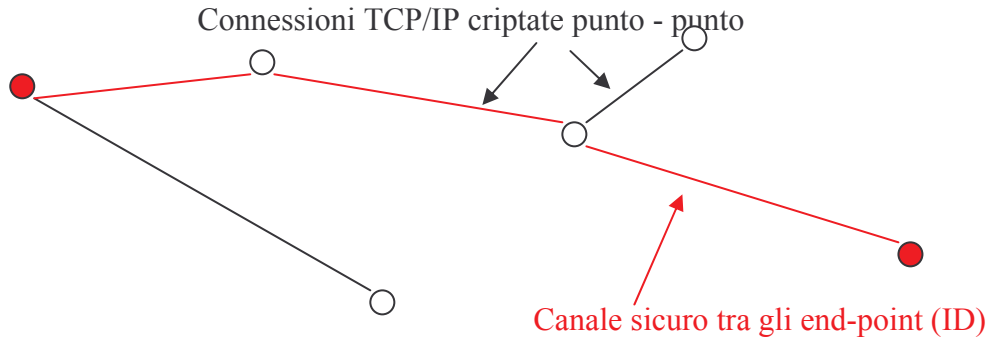

Fig.1

Non staremo ad approfondire in questa sede gli accorgimenti presi in termini di sicurezza per garantire questo "anonimato degli ID", cenni verranno fatti ove necessario in seguito, ma in uno scenario simile capiamo bene quanto gli algoritmi spesso usati nelle reti MANET si adattino alla situazione presentata. In particolare nel sistema è stato adottato un protocollo di routing che si ispira a due modelli:

- Ad Hoc On-Demand Distance-Vector Protocol (AODV)
- ARA – Ant colony Based Routing Algorithm for MANET[1]

Le caratteristiche di questi due modelli che hanno portato alla loro scelta e integrazione nella realizzazione del progetto sono:

[1] International Workshop on Ad Hoc Networking (IWAHN 2002), Vancouver, British Columbia, Canada, August 18-21, 2002

- *estrema capacità di adattamento a cambiamenti nella topologia della rete* in particolare apparzione di nuovi nodi e connessioni o in caso di disconnessioni
- *routing effettuato sulla base di informazioni strettamente locali* importante sia dal punto di vista dell'overhead (praticamente nullo visto che non serve scambio di informazioni tra i nodi), sia per mantenere anonimato relativamente all'ID
- *possibilità di qualificare la qualità di un percorso di routing*
- *supporto al routing multi-path*

Andiamo dunque ad analizzare i due protocolli, mostrando in seguito quali caratteristiche sono state impiegate nella rete ANts.

Ad Hoc On-Demand Distance-Vector Protocol

AODV è un protocollo che applica una politica on demand: questo significa che non servono scambi periodici di informazioni tra i nodi. Il protocollo è costituito da due fasi:

- *route discovery*
- *route maintenance*

Un nodo intenzionato a comunicare con un altro nodo cerca dapprima una strada nella sua tabella di routing, se la trova la comunicazione inizia immediatamente, altrimenti viene iniziata la *route discovery phase*.
Questa fase consiste nell'invio di un messaggio di route-request (in broadcast, vedremo che la scelta sarà per un net flooding). Se un nodo ha informazioni riguardo alla destinazione richiesta risponde con un route-reply message. In aggiunta, il nodo che effettua la replica crea una reverse-route entry nella sua routing table, in tale entry viene memorizzato l'indirizzo della sorgente che ha fatto la richiesta insieme ad altre informazioni aggiuntive (es. l'indirizzo del nodo da cui il messaggio è stato ricevuto, il timeToLive del messaggio etc.), alla entry è associato un time to live entro il quale può essere utilizzata e dopo il quale essa scade e viene eliminata.
La seconda fase del protocollo è quella di *route maintenance*, effettuata dal nodo sorgente e suddivisa in:

- movimenti della sorgente, a seguito dei quali la sorgente effettua un nuovo route discovery process
- movimenti di qualche nodo intermedio o della destinazione, che si risolvono nell'invio di un route error message alla sorgente. I nodi intermedi che ricevono tale messaggio eliminano le entry relative al nodo per cui si è verificato l'errore. La sorgente, ricevuto il messaggio di errore, effettua una nuova fase di route discovery. Per prevenire frequenti broadcast viene utilizzato l'espediente di inviare piccoli messaggi di HELLO, con tempo di vita limitato, che hanno l'effetto di segnalare la propria presenza ai nodi vicini

Ant Algorithms

Gli algoritmi Ant sono un sottinsieme degli algoritmi basati sulla swarm intelligence. Questi algoritmi modellano il comportamento di sciami di insetti per rsolvere problemi complessi (solitamente NP difficili) attraverso la cooperazione. Nelle loro forme tipiche assumono l'aspetto di sistemi ad agenti dove ogni singolo agente mostra il comportamento di una singola formica.

Concetti di base

L'idea alla base degli algoritmi Ants è presa dal modo in cui si comportano le formiche alla ricerca di cibo. Le formiche infatti all'inizio si muovono dirette verso il cibo, ma se un ostacolo si interpone tra loro e il cibo allora cambiano direzione, tipicamente in modo casuale. Mentre procedono esse lasciano però una scia di feromone, che traccia la via da loro percorsa. La concentrazione di feromone lungo un certo percorso indica quanto questo viene usato; nel tempo tale concentrazione decresce in quanto il feromone si disperde.
La nell'immagine (fig. 2) è possibile vedere un possibile scenario in cui una formica deve scegliere a fronte di una biforcazione una strada dal nido al cibo. La formica sceglie una direzione a caso, e se fosse sola non otterrebbe vantaggio, ma nel momento in cui uno sciame di formiche fa la stessa cosa, la situazione cambia: in media metà delle formiche andranno a destra e metà a sinistra, ma quelle che sceglieranno la via più breve torneranno più in fretta al nido e il relativo percorso

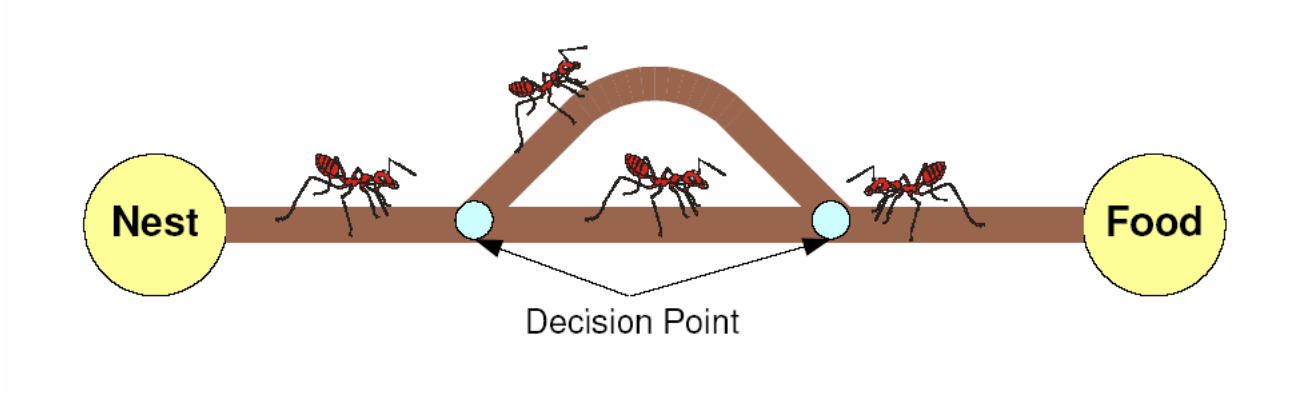

Fig. 2

mostrerà una concentrazione di feromone superiore rispetto alla via più lunga, dunque le formiche inizieranno a preferire tale percorso trovando in tal modo la via più breve. Dunque il comportamento delle formiche può di fatto essere imitato per trovare il percorso più breve in una rete. In particolare se la rete risulta molto dinamica un algoritmo ant based risulterà molto efficiente nell'adattarsi ai cambiamenti topologici della rete: interruzioni di link, apparizione e scomparsa di nodi.

*The **A**nt colony based **R**outing **A**lgorithm (ARA)*

Anche questo protocollo, che adatta le caratteristiche dei protocolli ant based alle reti Multi-Hop Ad-Hoc, consta di tre fasi:

1. *Route Discovery:* In questa fase vengono create nuove vie. La creazione di nuove vie richiede l'uso di una Forward Ant e di una Backward Ant. La Forward Ant è un agente che ha il compito di marcare il percorso verso la destinazione con una traccia di feromone e dualmente la Backward Ant traccia il percorso inverso fino alla sorgente. In pratica la Forward Ant non è altro che un piccolo pacchetto con un identificativo univoco (tipicamente idSorgente@numeroSequenza). In tal modo i nodi possono distinguere pacchetti duplicati sulla base dell'identificativo. Un nodo che riceve una Forward Ant per la prima volta crea un record nella sua tabella di routing (destinazione, hop successivo, feromone conc.). Il nodo interpreta l'indirizzo sorgente della Forward Ant come destinazione, e l'indirizzo del nodo precedente come hop successivo, la concentrazione di feromone è valutata sulla base del time to live rimanente del messaggio; successivamente il nodo passa la Forward Ant ai propri vicini. Forward Ant duplicate vengono riconosciute e rimosse. Il nodo di destinazione estrae le informazioni della Forward Ant, crea una Backward Ant e la invia verso il nodo

sorgente. Il comportamento nel caso della Backward Ant è il medesimo, una volta che la sorgente ha ricevuto indetro la Backward Ant il percorso è stato tracciato ed i pacchetti possono essere inviati.

2. *Route Maintenance:* Tale fase si occupa di mantenere attivo (tracciato) il percorso durante la comunicazione. Non servono particolari pacchetti per questo scopo: sono i dati scambiati stessi che vengono utilizzati per mantenere il percorso. Man mano che i dati passano nelle due direzioni la traccia di feromone viene rafforzata, se sono presenti più percorsi sarà la via dove i dati passano più spesso ad essere preferita, col trascorrere del tempo inoltre se non passano dati la traccia si attenua così come la memoria del percorso trovato.
3. *Route Failure Handling:* La terza ed ultima fase è dovuta alla dinamicità della rete e dunque ai cambiamenti che possono intervenire (comparsa/scomparsa di nodi e collegamenti). Se un nodo riceve un errore per un certo link allora setta il valore di feromone a 0 per quel lin, viene poi cercato un link alternativo nella tabella di routing, se lo trova lo usa, altrimenti si attiva il backtracking che nel caso estremo riporterà il pacchetto alla sorgente che dovrà rieseguire la fase di discovery.

Proprietà di ARA

- Operazioni inerenti al routing distribuite su ogni nodo della rete. Ogni nodo opera in maniera indipendente dagli altri e ha una memoria autonoma (i.e. traccia di feromone per ogni sorgente/destinazione)
- È loop-free a causa dell'utilizzo degli identificativi univoci
- I percorsi vengono stabiliti on-demand, quindi si evita l'overead inutile che si avrebbe con il routing tradizionale che stabilisce percorsi anche se non vengono usati
- Le parti della rete non attive non vengono sollecitate inutilmente

Altre proprietà importanti ai nostri fini

- Località: ogni nodo non ha cognizione di dove si trovi un altro nodo (i.e. distanza effettiva) sa solo che per raggiungerlo deve instradare il messaggio in una certa direzione e che alcune direzioni sono meglio di altre. Queste informazioni sono locali e non vengono scambiate tra i nodi.
- Multi-path: ogni nodo può conservare informazioni riguardanti diversi percorsi possibili (i.e. direzioni verso cui instradare un messaggio) per raggiungere una certa destinazione, può inoltre stabilire qual'è attulamente il migliore fra tali percorsi.

Considerazioni sull'efficienza

L'overead di protocolli come ARA è bassissimo ed è limitato alla banda necessaria per trasferire le Forward e le Backward Ant. Comprendiamo che se un percorso viene usato per trasferimenti corposi e viene riutilizzato spesso (ovvero prima che se ne perda la traccia) l'overead è in pratica nullo.

La rete ANts

Il progetto di tale rete, come spiegato, è stato sviluppato integrando i due protocolli illustrati.
La rete di base presenta poche ed essenziali funzionalità sulle quali poi vengono costruiti i servizi a livello superiore. Ci troviamo quindi di fronte ad uno stack analogo a quello OSI. Possiamo identificare tre livelli nello stack di ANts:

- Link/Rete (ID)
- Trasporto/Sessione (Canali sicuri tra end-point)
- Applicazione (Servizi di alto livello)

Idealmente il livello fisico, che manca nella lista, è costituito dai servizi internet su cui ci appoggiamo (TCP/IP). In pratica ci troviamo di fronte ad una rete sulla rete.

Analizziamo dunque i singoli livelli:

Link/Rete

Esattamente come avviene in internet anche qua abbiamo un indirizzo (ID).
Ogni nodo della rete (identificato dal proprio indirizzo) è un agente attivo in grado di:

- Fare connessioni con altri nodi (a livello TCP/IP)
- Inviare messaggi in broadcast sulla rete (tramite flooding)
- Inviare messaggi ad altri nodi
- Instradare messaggi diretti verso altri nodi

I messaggi sono strutturati e ad ogni livello è possibile accedere solo alla parte del messaggio relativa a quel livello (il tutto avviene esattamente come nello stack TCP/IP dove il messaggio è "spacchettato" dal router solo fino al livello necessario), da notare che in ANts tutto quanto è a livello superiore rispetto al livello corrente è criptato e quindi non può essere letto (al contrario di quanto avviene nel TCP/IP) se non dal legittimo destinatario con il quale (vedremo) a livello di sessione è stabilito il canale sicuro di comunicazione. Dunque ciò implica che osservando da fuori la rete ANTs si potrà osservare solo un traffico completamente criptato (connessioni punto-punto tra i nodi criptate), dall'interno, a livello di Link/Rete, si osserveranno messaggi con header in chiaro, ma body criptati, dunque solo a livello di Sessione si potrà accedere al corpo vero e proprio dei messaggi.

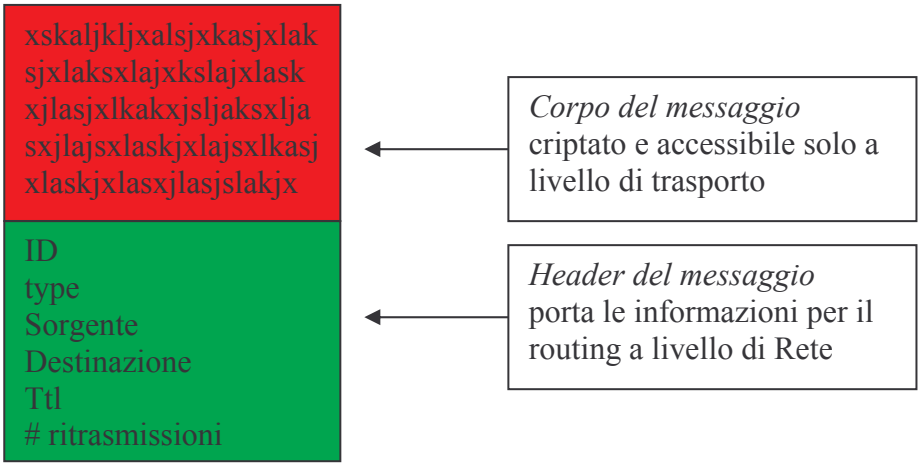

Fig. 3

Un discorso particolare va speso sull'indirizzo usato nella rete ANts (ID). Abbiamo già spiegato come l'ID possa non essere univoco, a causa di attacchi voluti o di casualità nella generazione (rara). Vediamo meglio la cosa. Sulla rete le comunicazioni vere e proprie (quelle che scambiano messaggi con body non vuoto) prevedono *sempre* che sia stabilito un canale sicuro tra i due end-point a livello di sessione, tale canale può prevedere o meno anche l'identificazione tra i due end-point (ad esempio tramite un protocollo challenge-response basato su chivi PGP, purtroppo non ancora implementato, ma previsto). Dunque l'avere ID duplicati non può portare problemi, se è

richiesta identificazione, questa verrà fatta a livello di sessione e una connessione con un ID (duplicato) non autorizzato porterà a fallimento, se non è richiesta identificazione il canale sarà comunque sempre stabilito con uno solo dei due ID - il fatto che sia quello giusto o meno non è rilevante visto che non è richiesta identificazione in questo caso - quindi l'altro non potrà leggere i messaggi e non potrà fare altro che reinstradarli o bloccarli, DoS quest'ultima, che difficilmente potrà avere effetto visto che l'algoritmo di routing prevede percorsi multipli.

Il protocollo di routing a livello di Link/Rete

Il core di tutto il sistema è la politica di routing applicata, che garantisce l'isolamento dell'informazione ID (solo il nodo stesso conosce il proprio ID) ed efficienza nell'instradamento dei messaggi. Come spiegato esso integra i due algoritmi visti in precedenza, vediamo dunque le caratteristiche che risultano da questa integrazione:

Nel protocollo di ANts esistono solo due fasi, queste fasi sono entrambe integrate in servizi effettivi, dunque l'overhead di protocollo è minimo.

Al solito:

1. *Route Discovery:* la fase di RouteDiscovery è integrata nei messaggi di broadcast, che vengono utilizzati per trovare risorse sulla rete.
2. *Route Maintenance:* come per gli altri due protocolli tale fase è integrata nel trasferimento dati.

I **messaggi** usati nel protocollo di routing a livello di Rete sono di tre tipi:

- Broadcast (net flooding)
- Unicast
- Acknowledge

Qualificati dal campo *type*.

Non è prevista una fase di errore vera e propria che riattiva una ricerca della sorgente a livello di rete come visto nei protocolli mostrati, i messaggi unicast nella fase di *Route Maintenance* vengono inviati con semantica *At Least Once*. Ciò significa che il messaggio è inviato più volte (fino a NMAX con un certo time out di ritrasmissione) fintanto che non si riceve il relativo Ack, se non lo si riceve entro NMAX ritrasmissioni allora il messaggio finisce tra quelli dichiarati falliti. Sarà compito degli strati superiori valutare la situazione e decidere.
I messaggi in broadcast e gli ack vengono sempre inviati solo una volta.

I messaggi possiedono due tipi di timeout: timeout in termini di hops (salti tra vicini) e sulla sorgente (in secondi). Quindi il messaggio può essere eliminato in rete perchè il suo time to live è a 0 su un nodo qualunque, oppure sulla sorgente, dopo un certo intervallo di tempo, dove è rimosso dai propri messaggi ed è inserito in quelli falliti, eventuali acknowledge sono a questo punto ignorati. In realtà la cosa è più complicata, soprattutto per quel che riguarda il TTL degli hops, a causa di problemi legati all'anonimato che qui non tocchiamo; l'idea è che il TTL viene decrementato ad ogni salto di un valore casuale (spiegazione molto riduttiva ripetto a quello che avviene in realtà) in modo da evitare analisi (anche statistiche) fatte sui ttl dei messaggi, TTL pari a 0 non causano l'eliminazione immediata del messaggio, ma attivano su ogni nodo una procedura che lo rimuove con probabilità pari al 10%, quindi l'eliminazione in media avverrà sopo 10 salti.

La fase di *Route Discovery* è di fatto attuata solo a livelli superiori rispetto a quello di rete e contestualmente a servizi (ad esempio query).
Le query sono messaggi di broadcast molto particolari, che verranno approfonditi nella sezioneche tratta il livello di Sessione, quando un nodo processa una query e scopre di avere una risorsa che fa match allora invia un acknowledge per la query con i risultati della stessa in piggybacking. Tale acknowledge effettua di fatto la fase di *Route Discovery* tracciando la strada all'indietro esattamente come fa la Backward Ant nel protocollo ARA o il Route Reply in AODV. Per ogni nodo attraverso cui passa, il messaggio di acknowledge fa memorizzare una tupla del tipo: [sorgente, destinazione, hop from, hop to]. Sorgente è l'ID sorgente del messaggio, Destinazione è l'ID di destinazione, hop from è il vicino da cui il messaggio proviene, hop to è il vicino a cui è stato instradato il messaggio. Sfruttando tali informazioni la strada è ovviamente tracciata nei due sensi.

Vediamo quindi l'altra fase: anche per il *Route Maintenance* sono integrate le politiche AODV e ARA, le strutture impiegate per attuare questi protocolli sono:

- Lista dei vicini con punteggi di efficienza (generica) nell'instradamento di messaggi.
- Lista dei messaggi consegnati (tupla: sorgente, destinazione, hop from, hop to);
- Lista dei messaggi in transito
- Lista dei propri messaggi
- Lista dei messaggi falliti

La posizione di una tupla nella lista dei messaggi consegnati indica da quanto tempo la via non è usata, dunque posizioni alte indicano vie poco usate (inefficienti o interrotte), oltre un certo limite le tuple sono scartate.

Ogni volta che un messaggio arriva al router di un nodo e viene processato si valutano le seguenti cose:

- Se il messaggio è unicast si valuta se si possiedono informazioni per instradarlo verso la sua destinazione, se le si hanno lo si instrada, altrimenti lo si instrada verso il vicino valutato più efficiente nella relativa lista dei vicini.
  In pratica la prima valutazione deriva da un integrazione tra AODV e ARA infatti la tupla è analoga a quelle di AODV e la posizione nella lista (che può contenere più tuple differenti per una stessa destinazione) richiama ARA per il multi path.
  Se poi un messaggio ritorna su un nodo dove era già passato (lo si verifica guardando tra i messaggi in transito) si decrementa l'efficienza del vicino verso cui era stato instradato e si tentano le altre vie o, in assenza di tuple relative, lo si instrada verso il vicino valutato più efficiente (strategia *protocolli ants di base* e ARA) sperando che lui possieda tuple che gli permettano di trovare una via verso la destinazione. Quando in un nodo tutte le vie sono state tentate il messaggio viene eliminato.

- Se il messaggio è di tipo Ack ed è relativo ad uno dei messaggi nella lista dei messaggi transitati allora si trasferisce la tupla nella lista dei consegnati e si aggiorna l'effficienza del vicino a cui il messaggio era stato instradato.

- I messaggi di broadcast, sebbene prevedano ack, non causano aggiornamenti di efficienza dei vicini.

Il Router presente in ogni nodo, oltre a fare tali valutazioni sui messaggi che riceve e a provvedere all'instradamento realizza due politiche per garantire efficienza nella rete:

- Routing sulla base della ***priorità del messaggio***
- ***Leaky Bucket*** per messaggi rimasti nella coda di attesa per un tempo > timeout
- ***Leaky Bucket*** per messaggi che arrivano quando già NMAX messaggi stanno venendo processati

I messaggi oltre alla tipologia determinata dal campo type sono divisi in classi. Ogni classe ha una sua priorità di instradamento stabilita sulla base della dimensione media (politica ***Max-min share***).

1. Messaggi semplici (vuoti) e Messaggi di Controllo sui Servizi
2. Messaggi di Richieste di Servizio
3. Messaggi di Dati

dimensione crescente

La politica di priorità tra i messaggi di una stessa classe è la stessa che Java applica a Thread con medesima priorità bloccati su un lock (tipicamente nondeterminismo).

I servizi del livello Link/Rete

Implementati sfruttando il protocollo descritto, i seguenti servizi sono messi a disposizione dei livelli superiori:

- Invio di messaggi di broadcast (net flooding)
- Invio di messaggi unicast (*At Least Once*)
- Connessione sicura a nodi vicini (Scambio DH – AES su connessioni TCP)

Per garantire che la rete possa rimanere connessa è stato sviluppato un servizio di background che periodicamente esegue query sulla rete per richiedere ip di nodi appartenenti alla stessa. Tale servizio ha due effetti: innanzi tutto ogni nodo (che può effettuare al massimo NMAX connessioni con altri vicini) ha sempre una lista di possibili sostituti a nodi vicini che si disconnettono; il fatto di sfruttare query sulla rete stessa garantisce che gli indirizzi vengano distribuiti tra tutti i nodi, ciò produce una metaconoscenza della rete su se stessa, dunque la rete tiene traccia dei suoi nodi e li recupera anche se essi vengono totalmente isolati a causa di disconnessioni. Il secondo effetto, collaterale, ma desiderabile, dell'esecuzione di tali query è la creazione e il mantenimento di percorsi verso altri nodi, ciò permette alla rete di dimenticare in fretta nodi che si sono disconnessi e di creare sempre nuove vie alternative verso nodi attivi.

Il protocollo a livello di Trasporto/Sessione

A tale livello vengono usati i servizi messi a disposizione da quello sottostante per cercare risorse sulla rete e realizzare connessioni sicure tra gli end-point (Scambio DH – AES) con le quali effettuare comunicazioni sulla rete.
Come già spiegato, inizialmente un nodo fa una o più query (broadcast) tali query tracciano le vie verso le risorse trovate. A questo punto, un nodo può richiedere una connessione sicura con una delle sorgenti delle risorse.

I messaggi di query sono strutturati in modo particolare e costituiscono una classe a parte:

l'header è composto da una chiave pubblica e dalla stringa di query. Chiunque voglia inviare risposte alla query dovrà criptare i risultati con la chiave pubblica e mandarli in piggybacking ad un acknowledge per il messaggio di query verso l'ID che ha generato la query.

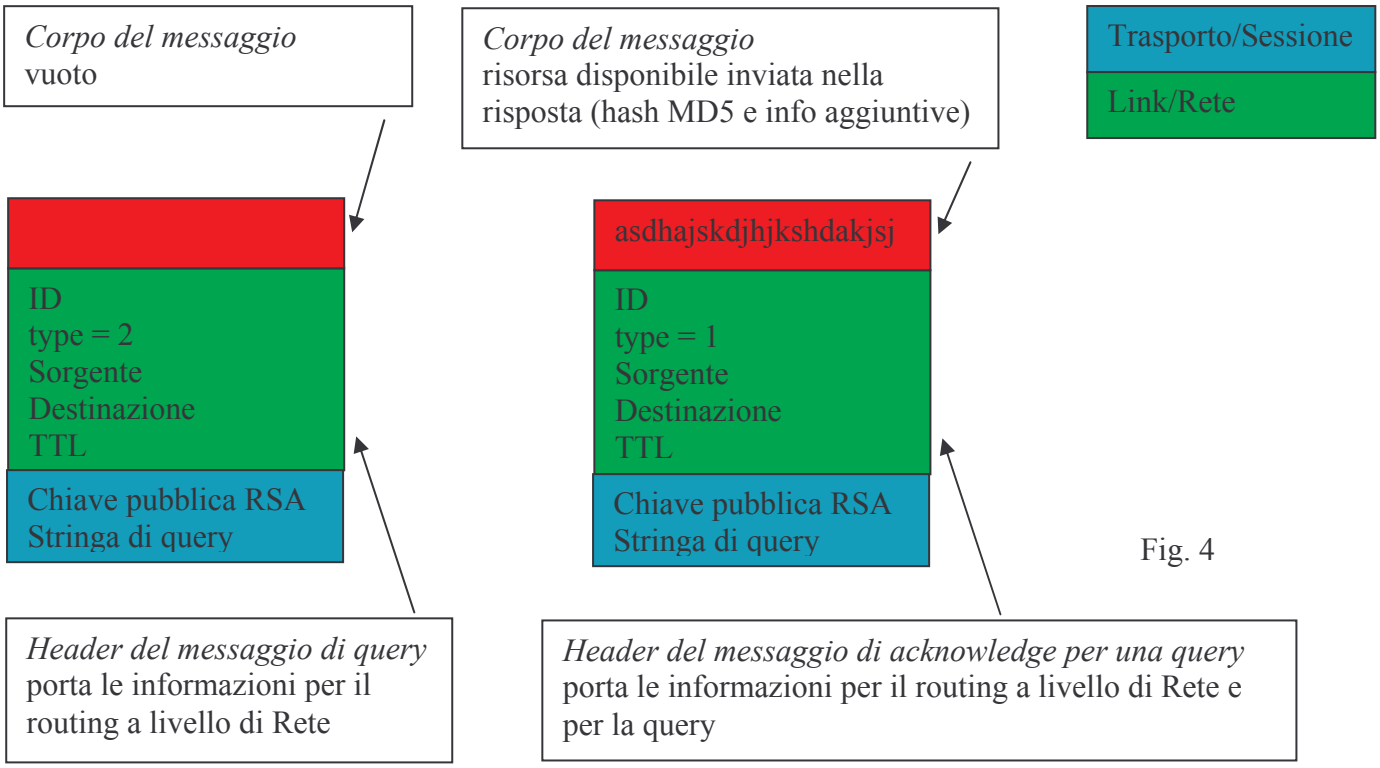

Fig. 4

Tracciata la strada all'indietro grazie all'acknowledge inviato (procedimento analogo a quanto visto nell' ARA e nell AODV) si ha a disposizione un percorso per inviare messaggi bidirezionali tra due nodi.

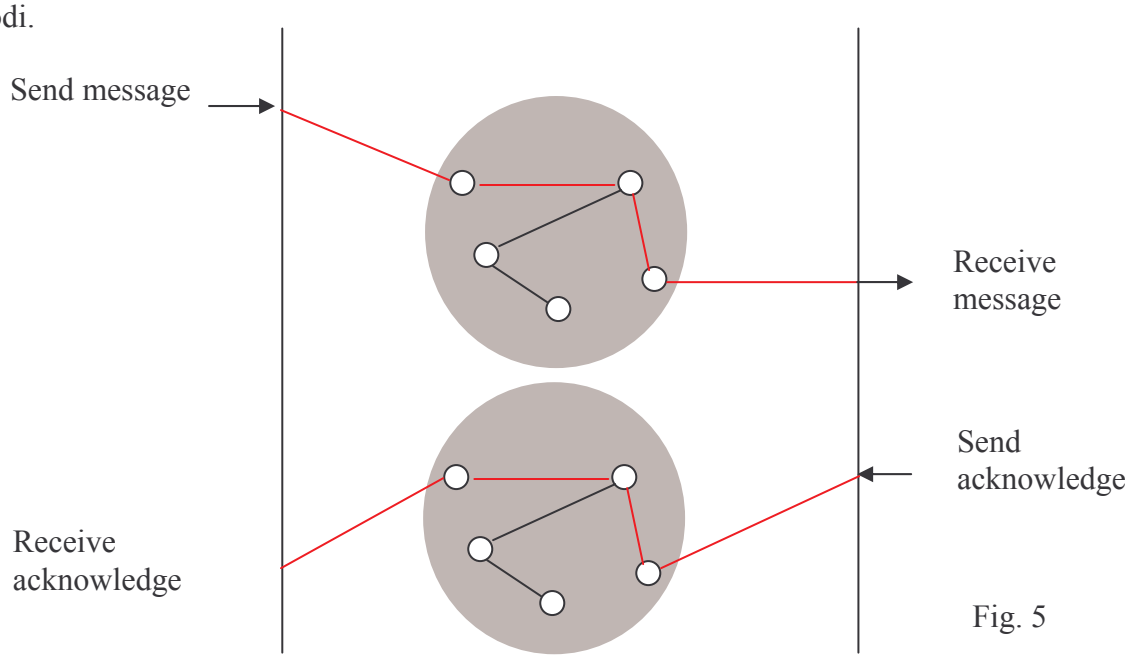

Fig. 5

Da notare che il percorso, proprio grazie agli algoritmi di routing implementati, non è fisso, ma varia dinamicamente con la rete: *politica multi-path.*

Per attuare la comunicazione vera e propria viene attuato il protocollo seguente per la richiesta di una connessione sicura; ogni messaggio richiede l'arrivo alla sorgente dell'acknowledge, come

visto sopra, per essere considerato *delivered*, nel caso in cui si perda l'acknowledge, si procede alla ritrasmissione, di qui la ha politica *At Least Once*:

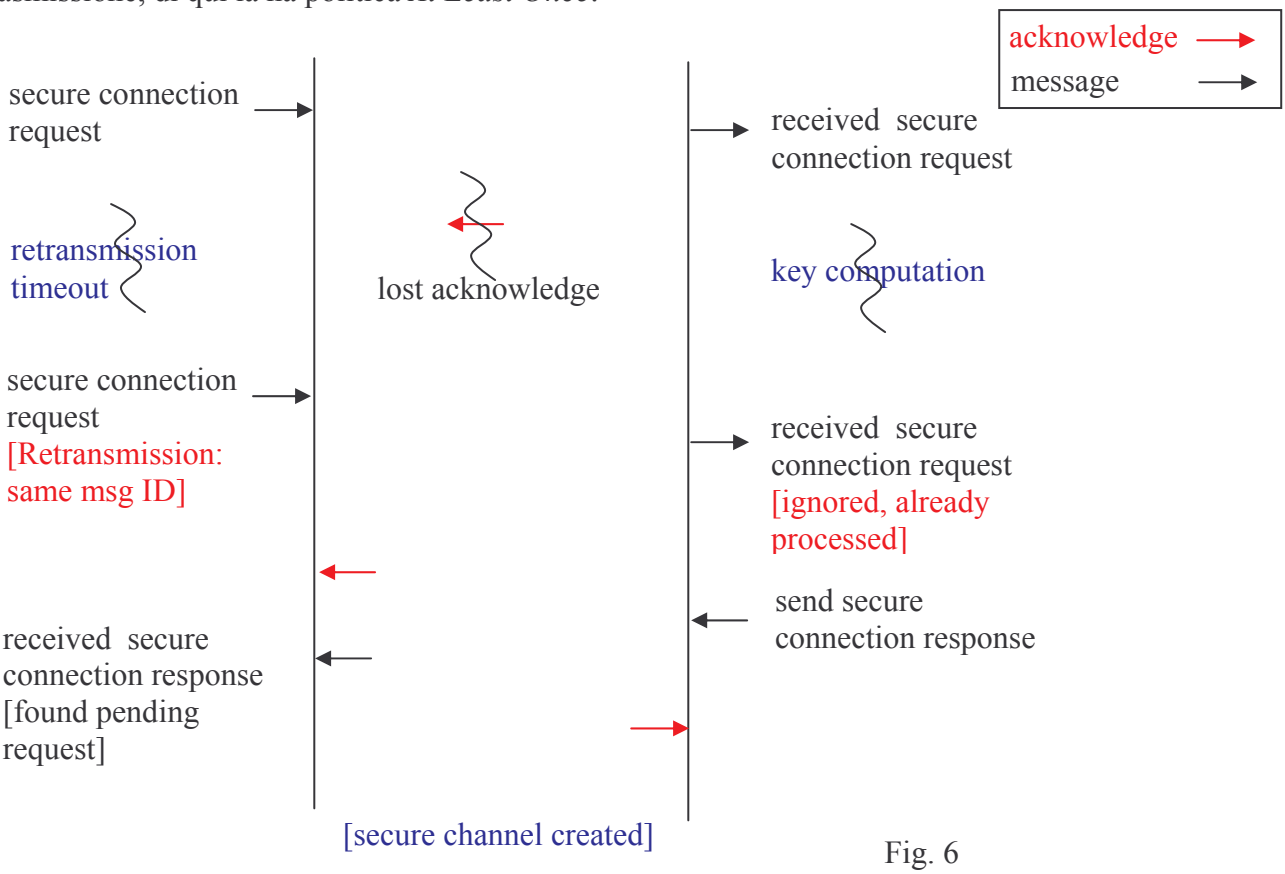

Fig. 6

Il protocollo mostrato è facilmente estendibile ed è possibile introdurre un'identificazione sicura nella creazione del canale ad esempio utilizzando certificati X.509.

Una volta stabilita una connessione sicura si ha a disposizione un canale per la comunicazione con l'altro end-point. La comunicazione avviene sempre a scambio di messaggi e sfruttando le API ANts è possibile definire nuovi tipo di messaggio che estendono la classe Message.

Si hanno a disposizione tutti gli strumenti per implementare politiche consistenti ed affidabili di comunicazione: in particolare è possibile sapere se un messaggio arriva con successo (scompare dalla lista dei propri messaggi) o fallisce (compare nella lista dei messaggi falliti: *no route found* oppure *timeout*). Come già spiegato la politica di invio è At Least Once, quindi a livello di Trasporto/Sessione bisognerà gestire la possibilità che un messaggio venga recapitato più volte ed eliminare i duplicati.

I servizi sviluppati e resi disponibili sopra il livello di Sessione

Stabilito il canale sicuro di comunicazione tra due end-point, che, esattamente come le Socket BSD, esiste solo sugli end-point stessi, è possibile usufruire dei servizi già sviluppati a livello di Sessione, oppure è possibile costruirne altri estendendo le API ANts: esattamente come per le Socket BSD è pensabile un servizio di invio di un byte all'altro endpoint, ma anche servizi di livello più alto:

Servizi esistenti sviluppati sull'infrastruttura di rete con protocolli a scambio di messaggi:

- *Invio di una query/Invio di una risposta ad una query*
  servizio che si appoggia sul livello di rete ed in particolare sulla possibilità di inviare messaggi in broadcast e relativi acknowledge con informazioni in piggybacking. L'header a livello di rete è esteso con informazioni aggiuntive che caratterizzano la query, il corpo delle risposte è criptato e accessibile solo a livello del nodo che ha emesso la rete (end-point a livello di Sessione) [Fig. 4]

- *Pulling di una risorsa [file] o di parte di essa [numero definito di byte ad un certo offset] presente sull'altro end-point*
  Il protocollo per questo servizio prevede l'invio di un messaggio di file pull, la ricezione di tale messaggio da parte del destinatario attiva la procedura di servizio che trasmette il file al richiedente.

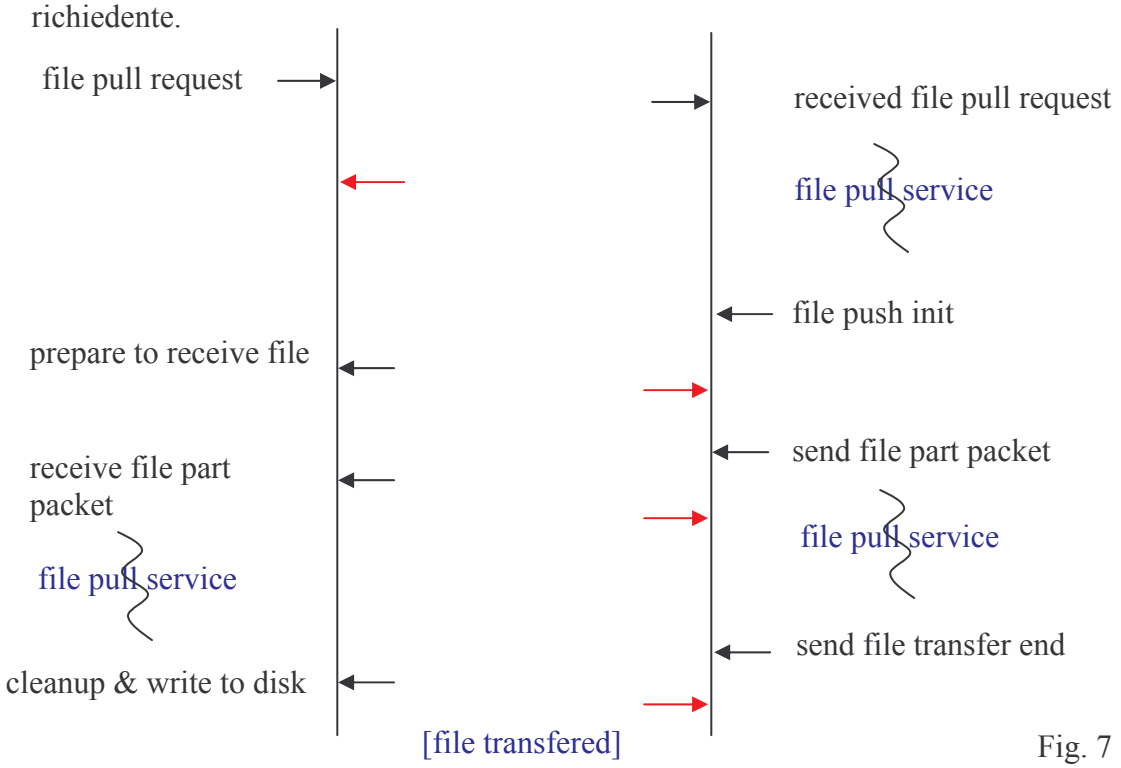

Fig. 7

  È necessario fare alcune precisazioni: il messaggio di *file pull request* non viene mai ritrasmesso, si deve richiedere un nuovo servizio a fronte di un fallimento. Gli altri messaggi scambiati possono subire duplicazioni senza side effects: ad esempio la ricezione di due *file push init* da parte del richiedente non darà problemi in quanto alla seconda ricezione egli sarà già in attesa di parti del file e ignorerà il messaggio. Ricezioni di duplicati di parti della risorsa non avrannno neppure esse effetti collaterali in quanto non verranno memorizzate in presenza di dati già ricevuti.

- *Pulling della dimensione di una risorsa [file] presente sull'altro end-point*
  Scambio di due messaggi: richiesta della dimensione, risposta.

Estensioni facilmente sviluppabili sull'infrastruttura di rete con protocolli a scambio di messaggi:

- *Invio di messaggi di testo tra nodi della rete:* in questo caso la fase di *route discovery* tramite query in broadcast non serve a reperire risorse ma a trovare il nodo con cui abbiamo intenzione di comunicare. Ciò ci permette anche di stabilire se un nodo è on-line oppure no,

dunque anche in questo caso il protocollo di discovery non sarebbe fine a se stesso, ma sarebbe comunque ottenuto come side-effect di un servizio.

I servizi sviluppati e resi disponibili a livello di Applicazione

Sfruttando quanto messo a disposizione sopra il livello di sessione è stato possibile sviluppare una rete distribuita per la condivisione di risorse. La rete, sebbene l'implementazione attuale non sviluppi ciò, non si limita alla condivisione di file, ma attraverso la creazione di nuovi servizi a livello di sessione può essere estesa per offrire ad esempio un supporto trasparente al protocollo http, dunque un accesso tramite browser alle risorse presenti in essa.

Servizi esistenti a livello di Applicazione:

- *Motore per recuperare risorse da sorgenti multiple:* gestisce i download da più sorgenti delle parti della risorsa che deve essere recuperata, si appoggia principalmente al servizio di *pulling di parti di una risorsa* disponibile sopra al livello di *Sessione,* coordina il recupero generando piani di download non sequenziali. In pratica vengono recuperati frammenti che stanno ad una distanza generata casualmente e relativamente prima con il numero totale di frammenti che compongono la risorsa, in tal modo si genera un piano di donwload che scarica tutti i frammenti seguendo una sequenza dipendente dalla distanza scelta. La generazione casuale della distanza nella fase iniziale rende uniforme la probabilità che un frammento sia scaricato all' n-esima richiesta di download e in tal modo rende efficiente il sistema di condivisione parziale di risorse.
- *Condivisione di risorse parziali durante il recupero:* a mano a mano che il servizio al punto precedente recupera parti della risorsa, queste vengono messe a disposizione in rete, in modo che anche altri nodi possano riceverle facendo richieste al nodo che sta recuperando la risorsa stessa.
- *Ricerca automatica di sorgenti per la risorsa che deve essere recuperata:* è attuata effettuando query periodiche che favoriscono al solito anche la formazione di persorsi alternativi per l'invio dei messaggi verso il richiedente.
- *Supporto a query strutturate testuali:* operatori logici AND e OR sui termini da ricercare nel nome della risorsa stessa. Possibilità di ricerca per hash MD5 del contenuto della risorsa.

Estensioni future

Come illustrato precedentemente possono essere sviluppate estensioni a vari livelli:

- *Sviluppare estensioni a livello di Sessione:* identificazione sicura tra nodi tramite certificati X.509

- *Supportare nuovi protocolli sopra il livello di sessione:* il protocollo supportato attualmente può essere a buon diritto considerato concettualmente un'estensione dell'FTP. Abbiamo parlato della possibilità di implementare l'HTTP sulla connessione sicura che viene stabilita a livello di Sessione, ma nulla vieta l'implementazione di qualsiasi protocollo su tale connessione.

- *Aggiungere funzionalità a livello applicativo:* es. supporto a messaggi di testo tra end-point connessi.

Bibliografia

Mesut Güneş, Udo Sorges, Imed Bouazizi *ARA – The Ant-Colony Based Routing Algorithm for MANETs* Department fo Computer Science - Aachen University of Technology. Aechen, Germany 2002

Mesut Güneş, Otto Spaniol *Routing Algorithms for Mobile Multi-Hop Ad-Hoc Networks* International Workshop NGNT

Hein Meling, Alberto Montresor, Özalp Babaoğlu *Peer-to-Peer Document Sharing using the Ant Paradigm* Department of Computer Science, University of Bologna; Department of Telematics, Norwegian University of Science and Technology,

M. Heissenbüttel, T. Braun *Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks* Institute of Computer Science and Applied Mathematics University of Berne

Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, Ion Stoica *Towards a Common API for Structured Peer-to-Peer Overlay* MIT Laboratory for Computer Science, Cambridge, MA. University of California, Berkeley, CA. Rice University, Houston, TX.

Gianni Di Caro, Marco Dorigo *AntsNet: A Mobile Agents Approach to Adaptive Routing* IRIDIA, Université Libre de Brusselles, Brusselles – Belgium.

Contatti

Roberto Rossi

Email: robertross@libero.it
Sito ufficiale: http://www.myjavaserver.com/~gwren/home.jsp?page=custom&xmlName=ants
SourceForge: https://sourceforge.net/projects/antsp2p