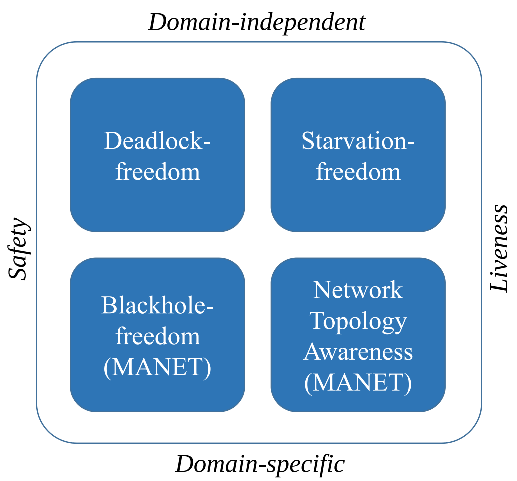
I protocolli N-AODV e B-AODV per MANET

Nota Preliminare

- La presente dispensa è una rivisitazione di uno dei seminari tenuti dal Dr. Gennaro Vessio per l'a.a. 2016-17, completato con considerazioni svolte negli articoli
  - Bianchi A., Pizzutilo S., Vessio G. Preliminary Description of NACK-based Ad-hoc On-demand Distance Vector Routing Protocol for MANETs. In: 9th International Conference on Software Engineering and Applications (ICSOFT-EA 2014), Vienna, Austria, pp. 500-505, SciTePress, 2014
  - Bianchi A., Pizzutilo S., Vessio G. CoreASM-based Evaluation of the N-AODV Protocol for Mobile Ad-hoc Networks. Journal of Mobile Multimedia, 12(1-2), pp. 31-51, Rinton Press, 2016
  - Bianchi A., Pizzutilo S., Vessio G., Intercepting Blackhole Attacks in MANETs: An ASM-based Model, Proc. of the 1st International Workshop on Formal Approaches for Advanced Computing Systems – FAACS2017, Trento - Italy, September 2017, to appear

Classi di proprietà



Le MANET

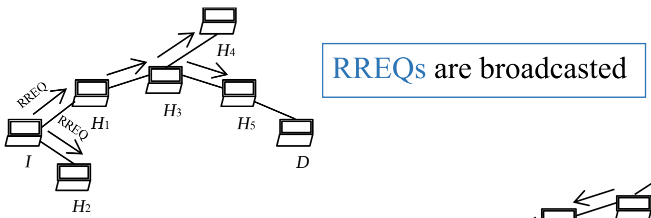
- Una rete mobile ad-hoc (MANET) è una rete wireless caratterizzata da:
  - Assenza di infrastruttura fisica fissa
  - Topologia dinamica
- Specifici protocolli di routing si basano sulla cooperazione fra più host per stabilire route fra coppie di end-point:
  - Proattivi
  - Reattivi
  - Ibridi

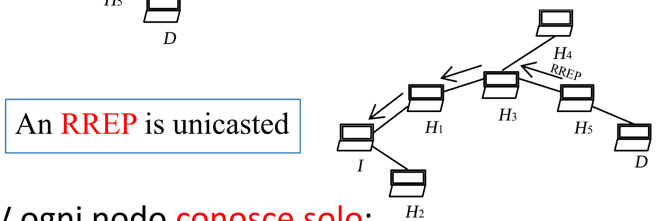
Network Topology Awareness

Concetto generale

- La network topology awareness (NTA) si riferisce alla conoscenza che ciascun host ha:
  - Degli altri host
  - Della loro raggiungibilità attraverso una route
- Tale aspetto è importante in svariate applicazioni:
  - Algoritmi di elezione del leader
  - Sicurezza
  - VANET
  - …
- *Come varia la NTA in funzione del protocollo di routing adottato?*

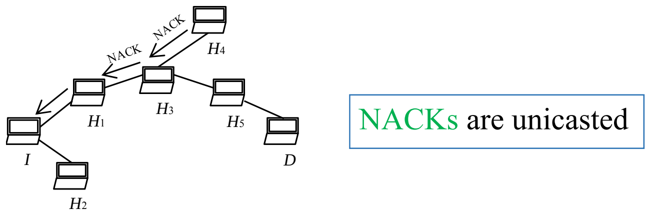
Limiti di AODV





- In AODV ogni nodo conosce solo:
  - I suoi vicini
  - Il prossimo nodo nel percorso verso nodi non vicini
- *Come migliorare tale aspetto?*

NACK-based AODV (N-AODV)



- Ogni nodo intermedio che ignora come raggiungere la destinazione comunica all'iniziatore del route discovery che «non sa nulla!»

Modellazione in ASM

- Una MANET che adotta N-AODV può essere modellata da una DASM composta di *Hosts* = $\{h_1, \ldots, h_n\}$
- Ogni ASM h_i è caratterizzata da:
  - *neighb*: *Hosts* → PowerSet(*Hosts*)
  - *wishToInitiate*: *Hosts* × *Hosts* → *boolean*
  - Tre code di pacchetti
  - Una tabella di routing

Host program

```
HostProgram(h_i) =
if ¬isEmpty(requests(self)) then {
  RREQ := top(requests(self))
  previousHop := sender of RREQ
  UpdateRoutingTable(self, RREQ)
  Router(RREQ, previousHop)
  dequeue RREQ from requests(self)
}
if wishToInitiate(self, dest) = true then
  Initiator(dest)

if ¬isEmpty(nacks(self)) then {
  NACK := top(nacks(self))
  if NACK.dest ≠ self then {
    previousHop := select r.nextHop ∈
      routingTable(self) with
        r.dest = NACK.dest
    UpdateRoutingTable(self, NACK)
    enqueue NACK into nacks(previousHop)
    dequeue NACK from nacks(self)
  }
}

if ¬isEmpty(replies(self)) then {
  RREP := top(replies(self))
  if RREP.init ≠ self then {
    previousHop := select r.nextHop ∈
      routingTable(self) with
        r.dest = RREP.init
    UpdateRoutingTable(self, RREP)
    enqueue RREP into replies(previousHop)
    dequeue RREP from replies(self)
  }
}
```


Predicati sugli stati

- Ciascuna ASM può trovarsi in uno fra più stati computazionali ciascuno caratterizzato da uno o più predicati sugli stati:
  - idle
  - routing
  - initiating
  - forwarding
- Per esempio, forwarding = *isEmpty*(*replies*(**self**)) = *false* ∨ *isEmpty*(*nacks*(**self**)) = *false*

Correttezza

Theorem. *The route discovery process always ends*

Sketch of proof. Each ASM can enter four different computational branches depending on which rule is executed. Every time a computational branch is entered the computation is reversible to the initial state. Therefore, the thesis holds □

Esperimento

- *Come si comporta N-AODV rispetto ad AODV?*
- Quesiti di ricerca:
  - Quale protocollo esibisce maggiore NTA?
  - Qual è il più efficace?
  - E il più efficiente?
- Metodo: simulazioni
- Tool: CoreASM
- Modello di mobilità: topology-based
- Parametri:
  - Dimensione della rete: 10, 25, 50 host
  - Livello di mobilità: alta, bassa
  - Numero di run: 500

Metriche

- Network Topology Awareness:
  - Routing tables size
  - Routing tables updates
  - Network awareness lag
  - Broadcast activations
- Efficacia:
  - Total rate of success
- Efficienza:
  - Control overhead

Formulazione delle ipotesi

- Per ogni metrica concernente NTA ed efficienza:
  - H_0: non c'è differenza statisticamente significativa fra i due protocolli
  - H_1: c'è differenza statisticamente significativa fra i due protocolli
- I valori relativi al total rate of success, invece, sono semplicemente confrontati

Risultati: Routing tables size

Scenario	AODV (media)	N-AODV (media)	p-value
10 host/Alta mobilità	13.092	21.13	< 0.0001
10 host/Bassa mobilità	25.772	28.544	< 0.0001
25 host/Alta mobilità	129.948	164.78	< 0.0001
25 host/Bassa mobilità	182.272	225.424	< 0.0001
50 host/Alta mobilità	575.566	809.966	< 0.0001
50 host/Bassa mobilità	1030.224	1135.414	< 0.0001

Risultati: Routing tables updates

Scenario	AODV (media)	N-AODV (media)	p-value
10 host/Alta mobilità	0.912	1.236	0.5661
10 host/Bassa mobilità	0.52	0.878	0.0016
25 host/Alta mobilità	3.404	5.182	0.0106
25 host/Bassa mobilità	2.202	3.46	0.0413
50 host/Alta mobilità	9.598	13.158	0.5598
50 host/Bassa mobilità	6.556	8.056	0.0369

Risultati: Network awareness lag

Scenario	AODV (media)	N-AODV (media)	p-value
10 host/Alta mobilità	6.103	2.6136	< 0.0001
10 host/Bassa mobilità	5.3769	2.8883	< 0.0001
25 host/Alta mobilità	3.8867	2.2163	< 0.0001
25 host/Bassa mobilità	3.3082	1.5847	< 0.0001
50 host/Alta mobilità	2.2764	1.047	< 0.0001
50 host/Bassa mobilità	1.5191	1.1705	< 0.0001

Risultati: Broadcast activations

Scenario	AODV (media)	N-AODV (media)	p-value
10 host/Alta mobilità	1.914	1.446	0.0002
10 host/Bassa mobilità	1.104	1.03	0.5848
25 host/Alta mobilità	4.24	3.052	0.0022
25 host/Bassa mobilità	2.722	2.226	0.0088
50 host/Alta mobilità	11.376	5.496	< 0.0001
50 host/Bassa mobilità	7.972	3.414	< 0.0001

Risultati: Total rate of success

Scenario	AODV (%)	N-AODV (%)
10 host/Alta mobilità	61.6	69
10 host/Bassa mobilità	66.8	71.2
25 host/Alta mobilità	71	76
25 host/Bassa mobilità	71.4	82.4
50 host/Alta mobilità	78.2	80.6
50 host/Bassa mobilità	85.4	90

Risultati: Control overhead

Scenario	AODV (media)	N-AODV (media)	p-value
10 host/Alta mobilità	1.07	1.32	0.6268
10 host/Bassa mobilità	1.05	1.27	0.3857
25 host/Alta mobilità	4.8	5.4	0.6457
25 host/Bassa mobilità	5.16	6.97	0.2728
50 host/Alta mobilità	11.15	10.768	0.0898
50 host/Bassa mobilità	8.846	7.74	0.4286

Analisi

- N-AODV:
  - –Fornisce agli host maggiore NTA rispetto ad AODV
  - –È più efficace
  - –Non è meno efficiente
- L'esperimento condotto soffre di minacce alla validità. *Quali?*

Blackhole-freedom

BACKGROUND

The context

- A Mobile Ad-hoc NETwork (MANET) is a network designed for wireless communications among mobile hosts:
  - It lacks of a fixed infrastructure
  - Hosts act both as initiator/destination and as intermediate router
  - Each host must be able to read all pcks for routing them to destination
- The intrinsic features of MANET make them vulnerable to security attacks
  - We focus on blackhole attack

Blackhole

- A blackhole is a malicious host in the network that sends fake routing information
- It claims to know the best route to reach a destination
  - Pcks are so routed to it
  - It can then misuse or discard them
- Sometimes there are several malicious hosts
  - the attack is cooperatively executed by the main blackhole and its colluders
- A Greyhole is a blackhole that alternates malicious and honest behaviour in an unpredictable way
  - Here we don't consider greyholes

Our research

- Formal specification of Blackhole-free N-AODV routing protocol – BN-AODV
  - The Ad-hoc On-demand Distance Vector (AODV) protocol is one of the most popular routing protocol for MANETs (*)
  - We proposed a variant of the protocol: the NACK-based AODV (N-AODV) (#); it improves the network topology awareness (°)
  - We now enrich N-AODV with the capability to intercept blackholes
- The protocol is formally specified by means of Abstract State Machines (ASMs)

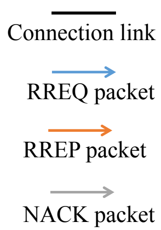
(*) C.E. Perkins et al. http://tools.ietf.org/html/rfc3561 (2003)

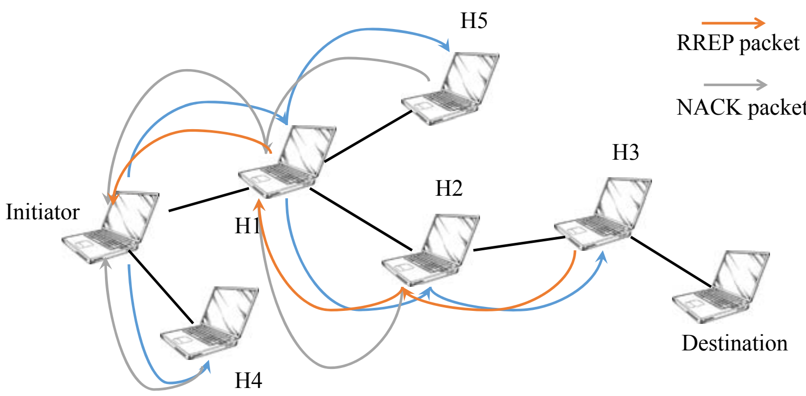
(#) A. Bianchi et al. Preliminary Description of NACK-based AODV for MANETs. ICSOFT (2014)

(°) A. Bianchi et al. CoreASM-based Evaluation of the N-AODV Protocol for Mobile Ad-hoc NETworks. J. of Mobile Multimedia, (2016)

Blackhole-free N-AODV

N-AODV



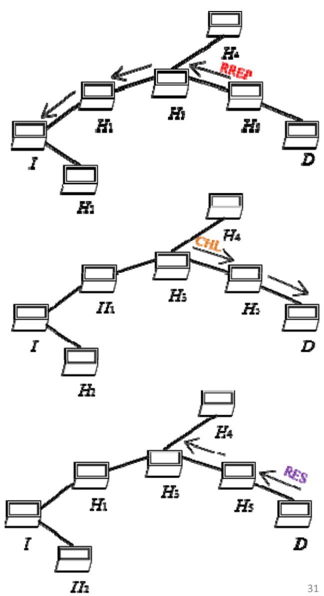


BN-AODV (1/2)

- The main idea is that each host receiving an RREP must verify the trustworthiness of the next host to dest
- It adds two control pcks:
  - Challenge (CHL) – is an encrypted nonce to be decremented by the destination
  - Response (RES) – is the encrypted decremented value

BN-AODV (2/2)

- H_3 receives an RREP from H_5
- H_3 unicasts CHL to D through H_5
  - If H_3 receives back the correct RES, then H_5 is considered trusted
  - Else H_5 is considered a blackhole



ASM model of BN-AODV

Generality

- The MANET is modeled by a DASM, including a set of Hosts={h_1, h_2, …, h_n}
  - each h_i models the behavior of the i-th host executing the protocol
- Each host behaves either as an honest node, or as a blackhole, or as a colluder
  - No greyhole
- Three ASMs describe the three different cases

Honest ASM - HostProgram

```
if ¬isEmpty(requests(self)) then {
 let RREQ = top(requests(self)), previousHop = RREQ.sender in
  UpdateRoutingTable(self, RREQ)
  Router(RREQ, previousHop)
  dequeue RREQ from requests(self)
}
if wishToInitiate(self) = true then
 forall dest ∈ Hosts with dest ≠ self do
  if initiateTo(self, dest) = true then
   Initiator(dest)
if ¬isEmpty(replies(self)) then
 let RREP = top(replies(self)), nextHop = RREP.sender in
  if RREP.init ≠ self then {
   choose entry ∈ routingTable(self) with entry.dest = RREP.init
   previousHop := entry.nextHop seq
   hasToVerify(previousHop, nextHop, RREP.dest):=true
  }
forall previousHop ∈ neighb(self) do
 forall nextHop ∈ neighb(self) do
  forall dest ∈ Hosts do
   if hasToVerify(previousHop, nextHop, dest) then
    Verify(top(replies(self)), previousHop, nextHop, dest)
if ¬isEmpty(responses(self)) then {
 let RES = top(responses(self)) in
  if RES.dest ≠ self then {
```

```
if ¬isEmpty(challenges(self)) then {
  let CHL = top(challenges(self)) in
   if CHL.dest = self then {
    let previousHop = CHL.sender in
     create_RES seq
     enqueue RES into responses(previousHop)
     dequeue CHL from challenges(self)
   }
   if CHL.dest ≠ self then {
    choose entry ∈ routingTable(self) with entry.dest = CHL.dest
    nextHop := entry.nextHop seq
    enqueue CHL into challenges(nextHop)
    dequeue CHL from challenges(self)
   }
 }
if ¬isEmpty(nacks(self)) then {
  let NACK = top(nacks(self)) in
   if NACK.dest ≠ self then {
    choose entry ∈ routingTable(self) with entry.dest = NACK.dest
    previousHop := entry.nextHop seq
    UpdateRoutingTable(self, NACK)
    enqueue NACK into nacks(previousHop)
    dequeue NACK from nacks(self)
   }
 }
```


Honest ASM - Initiator

```
if dest ∈ neighb(self) ∨ dest ∈ routingTable(self) then {
  CommunicationSession(dest)
  initiateTo(self, dest) := false
 }
 if dest ∉ neighb(self) ∧ dest ∉ routingTable(self) then {
  create_RREQ seq
  BroadcastRREQ(RREQ)
  initiator_waiting(self, dest) := true
  initiator_timeout(self, dest) := default_value
 }
 if initiator_waiting(self, dest) then
initiator_timeout(self,dest):=initiator_timeout(self,dest)
−1 seq
  if ¬isEmpty(replies(self)) then
   forall r∈replies(self) with r.init=self and r.dest =dest
```

```
if trusted(self, dest) then {
  CommunicationSession(dest)
  initiateTo(self, dest) := false
  initiator_waiting(self, dest) := false
}
if initiator_waiting(self, dest) ∧
¬isEmpty(nacks(self)) then {
  forall n ∈ nacks(self) with n.dest = self do {
   UpdateRoutingTable(self, n)
   dequeue n from nacks(self)
  }
}
if initiator_waiting(self, dest) ∧
initiator_timeout(self, dest)=0
then {
  initiateTo(self, dest) := false
  initiator_waiting(self, dest) := false
```


Honest ASM - Verify

```
if ¬verify_waiting(self, dest) then {
  create_CHL seq
  enqueue CHL into challenges(nextHop)
  verify_waiting(self, dest) := true
  verify_timeout(self, dest) := default_value
 }
 if verify_waiting(self, dest) then {
  if ¬isEmpty(responses(self)) then
   if ReliableRREP(self, top(responses(self))) then {
    trusted(self, dest) := true
    verify_waiting(self, dest) := false
    dequeue top(responses(self)) from
responses(self)
   }
  verify_timeout(self, dest) := verify_timeout(self,
dest) – 1
 }
```

```
 if verify_waiting(self, dest)∧verify_timeout(self,
dest) =0 then {
  trusted(self, dest) := false
  verify_waiting(self, dest) := false
 }
 if trusted(self, dest) then {
  UpdateRoutingTable(self, RREP)
  UpdateTrustTable(self, nextHop)
  dequeue RREP from replies(self)
  if previousHop ≠ null then
   enqueue RREP into replies(previousHop)
 }
 if ¬trusted(self, dest) then {
  UpdateTrustTable(self, nextHop)
  dequeue RREP from replies(self)
 }
```

Malicious ASM - Blackhole Program

```
if ¬isEmpty(requests(self)) then {
  let RREQ = top(requests(self)), previousHop = RREQ.sender in
    UpdateRoutingTable(self, RREQ)
    MaliciousRouter(RREQ, previousHop)
    dequeue RREQ from requests(self)
 }
```

where the MaliciousRouter submachine is simply:

```
MaliciousRouter(RREQ, previousHop) ≡
 create_RREP seq
 enqueue RREP into replies(previousHop)
```


Malicious ASM – Colluder Program

```
if ¬isEmpty(replies(self)) then {
  let RREP = top(replies(self)) in
   if RREP.init ≠ self then {
    choose entry ∈ routingTable(self) with entry.dest = RREP.init
    let previousHop := entry.nextHop seq
     enqueue RREP into replies(previousHop)
    dequeue RREP from replies(self)
   }
 }
```


Correctness (1/2)

- *The honest hosts intercept any single blackhole attack*
  - Let $n_0, n_1, \ldots, n_{k-1}, n_k, b$ be the route from the initiator to the blackhole
  - We prove that any fRREP is discarded by n_k, i.e. fRREP is not enqueued to n_{k-1}
  - The only rule allowing n_k to enqueue fRREP to n_{k-1} is guarded by a condition that verifies the trustworthiness of the route (red rule in Verify)

Correctness (2/2)

- *The honest hosts intercept any cooperative blackhole attack*
  - Let's consider the worst case: $n_0, c_1, c_2, \ldots, c_k, b$
  - We prove that if fRREP is enqueued to n_0 the communication does not start
  - In fact the communication starts (blue rule in Initiator) only after n_0 has verified the trustworthiness of the route
  - If at least one honest node is between n_0 and b is a special case of interception of single blackhole

Contribution

- From the application domain viewpoint:
  - Proposal of BN-AODV
  - Thanks to ASMs, the ability of the protocol to intercept all attacks is proved
- From the ASM viewpoint:
  - Application to a complex, real, up-to-date case study

Future work

- Investigation about greyhole
- Simulations for mitigating
  - the draconian approach of BN-AODV
  - the high overhead
- Implementation in ASMETA