URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks

Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang

1

Introduction

- Problem
  - How to provide access control for a mobile ad hoc network
  - Network-layer access for routing and packet forwarding
- Goal
  - Grant access to well-behaving nodes and deny access to misbehaving nodes
- Misbehaving node is defined as being selfish and/or malicious

2

Challenges

- Ad hoc networks cannot perform access control in the same way as other wired/ wireless networks.
  - Infrastructureless, No well-defined line of defense
- Users/devices are allowed to roam freely
  - Access control services must be available everywhere

3

Challenges

- Insider attacks are a higher risk
  - Nodes are active participants in access control
- Dynamic node membership
  - Nodes can join, leave, and fail

4

Introducing URSA

- URSA is a fully localized ticket-based approach to access control
- Well-behaved nodes use a certified ticket to participate in routing and packet forwarding
- No valid ticket = Misbehaving = No access

5

URSA Characteristics

- Multiple-node consensus and fully localized instantiation
- Every node locally contributes to the access control system
- All nodes collectively secure the network
- Soft states
  - Regular ticket renewal is required
  - Certification function is refreshed periodically
  - Enables dynamic join and leave

6

Related Work

- SPINACH
  - Users are authenticated through Kerberos-enabled telnet. Routers provide MAC-based access control
- NetBar
  - Separates public LANs for configuration and authentication via port-based control.

➢ These techniques require infrastructure (e.g. routers) not found in mobile ad hoc networks !

7

Related Work (cont'd)

- Network firewalls
- Remote-access VPN
- GSM

➢ All of these systems have a clear boundary between users and services where access policy can be enforced

8

Related Work (cont'd)

- COCA
  - Multiple certificate management servers are deployed for ticket issuance
  - Requires lots of message overhead for ticket renewal and revocation
  - More on this later
- 802.11
  - WEP-based access control is subject to a number of attacks
  - Infrastructure requirements (e.g. APs)

9

System Models

- Network Model
- Localized Group Trust Model
- Attack Model

10

Network Model

- Wireless MANET
  - limited bandwidth
  - error-prone insecure wireless links
- Reliable multi-hop transmission is not assumed
- Nodes join, leave or fail over time
- Nodes are capable of:
  - Neighbor discovery and monitoring

11

Localized Group Trust Model

- Node must be trusted by *k* trusted nodes
  - Results in local and network-wide trust
- Trust relation is soft-state (T_{cert})

➢ Trust management and maintenance are distributed in both spatial (*k*) and temporal (T_{cert}) domains to support large, dynamic ad hoc networks

12

Attack Model

- Network layer attacks only (no PHY/MAC)
  - Routing and Packet Forwarding
- Single or Multiple misbehaving nodes
- Wireless link attack
  - Eavesdrop, record, inject, reorder, resend, DoS
- Direct Node attack
  - Compromise/control via software bugs or system backdoors

13

Attack Model (cont'd)

- Multiple nodes could conspire (e.g. joint accusation)
  - Assume < *k*-1 collaborative malicious nodes over network lifetime and any time interval
- Insider attack is a primary concern
  - Nodes are active participants in access control mechanisms
  - Malicious nodes can roam to extend impact

14

URSA Design

- Overview
- Ticket
- Ticket Services via Local Collaboration
- Self-Organized Bootstrapping
- Resisting Attacks
- Soft States to Improve Robustness

15

URSA Design - Overview

- Valid ticket is required to participate in network
  - Valid = Certified and Unexpired
- Nodes establish mutual trust relationship with one-hop neighbors via ticket exchange
- The neighbors will monitor for bad behavior

16

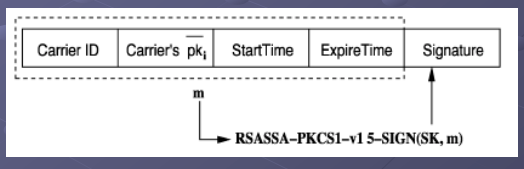
URSA Design – Overview (cont'd)

- Prior to ticket expiration, node must request neighbors to collectively renew his ticket
- Initial tickets are issued
  - By a coalition of existing nodes after external authenticity verification
  - Tentative admittance for a closely-monitored trial period, allowing only packet forwarding
  - Pay to play

17

URSA Design – Ticket

- Carrier ID = node's MAC or IP address
- Carrier pk_i = node's personal public key
- Start/Expire Time = T_{cert}
- Signature = integrity verification based on system RSA secret key (SK)



18

URSA Design – Ticket (cont'd)

- No single node has complete SK exponent that is used to sign tickets
  - Each node (v_i) has a partial share (P_{v_i})
  - P_{v_i} is used to sign partial tickets
- Ticket exchange validation
  - Verify ticket signature with system public key (PK, N)
  - Challenge/response to confirm private key (corresponding to pk_i) is held by claimant

19

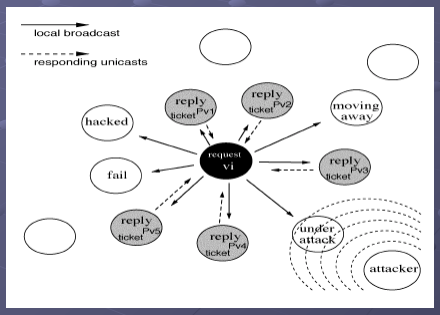
URSA Design – Ticket (cont'd)

- After certification, nodes help each other in routing and forwarding packets
- Nodes without valid tickets are considered misbehaving and denied participation
- Neighbors keep an eye out for misbehavior
  - specific detection mechanisms are left to individual nodes

20

Ticket Services via Local Collaboration

- URSA Ticket Services
  - Ticket Renewal
  - Mobility Impact
  - Ticket Revocation
- Performed locally to maximize availability and resilience



21

Ticket Services – Ticket Renewal

- Prior to expiration, a node sends a ticket renewal request to its one-hop neighbors
- Each neighbor checks its records to determine any misbehavior during T_{mon}
  - T_{mon} = avg time a node remains in one-hop range
- If well-behaved, each neighbor provides a partial ticket

22

Ticket Services – Ticket Renewal (cont'd)

- *k* partial tickets are combined to construct a renewed complete ticket
- Timeout value
  - Set to allow *k* neighbors to process/transmit *k* partial tickets
  - Processing delay + transmission time + channel access time
- Improper partial tickets will be dropped and noted as misbehavior

23

Ticket Services – Mobility Impact

- Benefit
  - In case of sparse neighborhoods (< *k* nodes)
    - nodes can move around to accumulate additional partial tickets
    - nodes can anticipate other nodes moving into the neighborhood on a frequent basis
- Detriment
  - Misbehaving nodes can take advantage of $T_{cert} > T_{mon}$ and launch an "intermittent moving attack"

24

Ticket Services - Ticket Revocation

- Helps mitigate intermittent moving attacks
- Node records
  - Direct monitoring records
  - Ticket Revocation List (TRL)
- TRL entry = node ID + accusation list
  - If < *k* accusers → node is "suspect"
  - Else → node is "convicted"
- A node can also be "convicted" if misbehavior is directly detected

25

Ticket Revocation (cont'd)

- How a node is accused
  - A neighbor node, v_j, determines misbehavior and enters node into its TRL as "convicted"
  - v_j floods a signed accusation
  - Other nodes receive accusation
    1. Verifies accuser is not a convicted node
    2. If not, updates its TRL accordingly
    3. If # of accusers = k, node is marked as "convicted"

26

Ticket Revocation (cont'd)

- Accusation propagation range

$$TTL \geq \left\lceil \frac{T_{\text{cert}} \cdot 2S_{\max}}{D} \right\rceil .$$

TTL = Time To Live
T_{cert} = Ticket validity period
S_{max} = Max node moving speed
D = Max one-hop transmission range

27

Ticket Revocation (cont'd)

- Assuming TTL = m, a particular node's TRL contains nodes at most m+1 hops away
- Nodes only hold each TRL entry for T_{cert}
  - After which, the convicted node's ticket is expired away.

28

URSA Design – Self Organized Bootstrapping

- During bootstrapping, an authority has to privately send each node its share of the ticket signing key (SK)
- Self-Initialization
  - Authority is only responsible for x*k nodes
  - Those nodes then collaboratively initialize other nodes
  - Similar to ticket renewal

29

URSA Design – Resisting Attacks

- Single node attacks
  - Single false accusations can only label a node as "suspect", and are pruned after T_{cert}
  - Roaming attacks should be detected and result in ticket revocation
  - Attacks on routing and forwarding must be caught by local monitoring mechanisms
    - URSA ticket services can isolate the offending nodes
    - False negatives are minimized by neighborhood involvement

30

URSA Design – Resisting Attacks

- Multiple node attacks
  - False accusation protection is provided assuming < *k*-1 attackers over T_{cert}
  - TRL exploit (e.g. potential attack partners) mitigators
    - Accusation flooding is local not global
    - TRL entries are purged after T_{cert}
  - T_{cert} reduction can help reduce conspired attacks, but at the cost of ticket renewal frequency

31

URSA Design – Soft States

- Periodic refreshing of each node's secret share
  - Strengthens security of ticket signing key (SK)
  - Mitigates long-term attacks against *k*-1 (or more) victim nodes' secret shares
- Mechanism is based on self-initialization
  - Coalition of *k* nodes update their shares
  - Then, the neighbors are updated… and so on

32

Implementation

- Cryptographic Implementation
- Protocol Implementation

33

Cryptographic Implementation

- Ticket Renewal
  - Distribution of the exponent SK of the ticket signing key
  - Multisignature generation mechanism based on the SK distribution
- Secret sharing options: polynomial or additive
  - Polynomial is chosen due to its ability to handle dynamic grouping

34

Cryptographic Implementation

- Each node hold a polynomial share
  $P_{v_i} = f(v_i) \bmod N$,
  $f(x) = SK + f_1 x + \ldots + f_{k-1} x^{k-1}$ is a uniformly distributed random polynomial
- Partial ticket generation

$$TICKET_{v_j} = (ticket)^{(P_{v_j} \cdot l_{v_j}(0) \bmod N)} \bmod N$$

where $l_{v_j}(0) = \prod_{r=1, r \neq j}^{k} (v_r / (v_r - v_j)) \bmod N$.

35

Cryptographic Implementation

- Candidate ticket generation (TICKET')
  - Partial tickets are combined together

$$TICKET' = \prod_{r=1}^{k} TICKET_{v_r} \bmod N.$$

Note that

$$\begin{aligned} TICKET' &= \prod_{r=1}^{k} TICKET_{v_r} \\ &= (ticket)^{\sum_{r=1}^{k} (P_{v_r} l_{v_r}(0) \bmod N)} \\ &= (ticket)^{t \cdot N + SK} \\ &= TICKET \cdot (ticket)^{t \cdot N} \bmod N \end{aligned}$$

where t is an integer bounded by k : $0 \leq t < k$.

36

Cryptographic Implementation

- *k*-bounded coalition offsetting is used to recover the new ticket, TICKET

Inputs: $TICKET'$: the candidate ticket
  $ticket$: statement of the ticket, to be signed
Output: $TICKET$: ticket
1: $Z := (ticket)^{-N} \bmod N$
2: $r := 0,\ Y := TICKET'$
3: **while** $r < k$ **do**
4: $\quad Y := Y \cdot Z \bmod N,\ r := r + 1$
5: $\quad$ **if** $(ticket = Y^{PK} \bmod N)$ **then**
6: $\quad\quad$ break **while**
7: $\quad$ **end if**
8: **end while**
9: **output** $Y = TICKET$

37

Cryptographic Implementation

- Self-Initialization and Share Update
  - Trusted authority
    - Broadcasts secret sharing polynomial *f(x)*
    - Initializes the first *k* nodes with their shares
    - Destroys *f(x)* and quits
  - Initialized nodes
    - Collaboratively initialize their neighbor nodes using their partial shares
  - Secret share update
    - Similar to self-initialization

38

Protocol Implementation

- Communication protocols implemented in *ns-2* network simulator
- Ticket renewal requests are broadcasted, replies are unicasted
- Reply message collision avoidance reduction via simple backoff mechanism
  - Node generates random backoff value
    - [0,m] in time unit Δt, m = # of neighbors

      Δt = reply transmission time + propagation delay

39

Protocol Implementation

- Insufficient replies may be received
  - Lack of neighbors
  - Replies lost due to collisions or corruptions
- Therefore, nodes will initiate their ticket renewal requests starting at $T_{cert} - 3T_{det}$
- Node roams to another area if insufficient replies are received during the initial $1.5T_{det}$

40

Protocol Implementation

- Misbehavior detection algorithm
  - Listen in on the channel during the detection period T_{det}
  - In order to conserve energy, use statistical sampling
  - Node randomly samples behavior of its neighbors and sleeps the rest of the time

41

Performance Evaluation

- Computation Cost
- Communication Performance

42

Performance Evaluation

- Computation Cost
  - UNIX/C, 10 000 lines of code
  - RSASSA-PSS signature scheme used to certify tickets
  - Three platforms
    - Compaq iPAQ3670 Pocket PC
    - Laptop – Pentium II
    - Laptop – Pentium III

RSASSA-PSS =RSA Signature Scheme with Appendix – Probabilistic Signature Scheme

43

Performance – Ticket Certification

TABLE I
RSA AND URSA TICKET CERTIFICATION ($k = 5$, POCKET PC IPAQ3670, STRONGARM 206 MHz CPU)

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.15	0.29	0.39
1280	0.01	0.26	0.50	0.57
1536	0.01	0.41	0.79	0.88
1792	0.01	0.61	1.18	1.29
2048	0.01	0.85	1.71	1.79

TABLE II
RSA AND URSA TICKET CERTIFICATION ($k = 5$, LAPTOP, PENTIUMII 300 MHz CPU)

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.07	0.16	0.18
1280	0.01	0.13	0.27	0.31
1536	0.01	0.21	0.46	0.51
1792	0.01	0.31	0.67	0.74
2048	0.01	0.44	1.01	1.08

TABLE III
RSA AND URSA TICKET CERTIFICATION PERFORMANCE ($k = 5$, LAPTOP, PENTIUMIII 850 MHz CPU)

key (bit)	RSA-PK (sec)	RSA-SK (sec)	URSA-PTC (sec)	URSA-Combine (sec)
1024	0.01	0.02	0.05	0.06
1280	0.01	0.04	0.10	0.11
1536	0.01	0.08	0.15	0.18
1792	0.01	0.11	0.24	0.26
2048	0.01	0.16	0.36	0.38

- RSA-PK = standard RSA PK verification
- RSA-SK = standard RSA SK verification
- URSA-PTC = partial ticket computation
- URSA-Combine = delay caused by combining k partial tickets

44

Performance – Ticket Service and Self Initialization

TABLE IV
URSA TICKET SERVICE PERFORMANCE VERSUS k (AVERAGE VALUE ON 10 RUNS, RSA KEY LENGTH 1024 b, TIME UNIT: SECOND)

k	iPAQ3670, ARM 206MHz		Laptop, PIII 850MHz	
	URSA-PTC	URSA-Combine	URSA-PTC	URSA-Combine
2	0.290	0.397	0.059	0.063
3	0.291	0.391	0.057	0.062
5	0.293	0.394	0.057	0.062
7	0.291	0.393	0.056	0.062
10	0.292	0.392	0.058	0.061
20	0.291	0.393	0.059	0.060
30	0.291	0.396	0.056	0.063

TABLE V
URSA SELF-INITIALIZATION PERFORMANCE ($k = 5$, TIME UNIT: SECOND)

key (bit)	iPAQ3670,ARM 206MHz		Laptop, PIII 850MHz	
	URSA-PSS	URSA-Sum	URSA-PSS	URSA-Sum
1024	0.09	0.10	0.01	0.01
1280	0.10	0.11	0.01	0.01
1536	0.10	0.11	0.01	0.01
2048	0.10	0.11	0.01	0.01

- *k* does affect system performance significantly
  - Partial tickets are computed in parallel
  - Increase in *k* does not significantly increase the overhead in combining *k* partial tickets
- Processing latency associated with self-initialization is not significantly affected by key length

45

Performance Evaluation

- Communication Performance
  - UDP-like transport agent, one-hop IP broadcast
  - Network sizes 50 to 100 nodes
  - Node moving speed 1 to 15 meters/sec
  - Mobility model – random way-point
    - Max range [s_{max}-s_{min}] allows more randomness in speed setting
  - T_{cert} = 300 seconds
  - k = 5

46

Performance Evaluation

- Communication Performance
  - Success ratio
    - Ratio of the # of successful ticket renewals performed by all nodes, over the total # of renewals that should take place during simulation
  - Average number of retries
    - The number of retries before a node successfully receives the ticket service

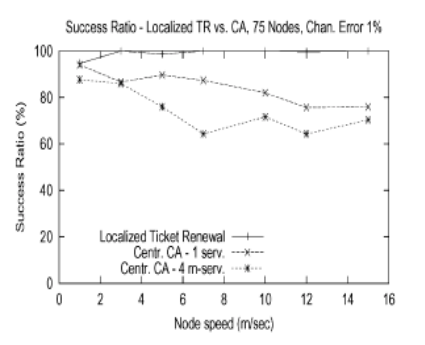
47

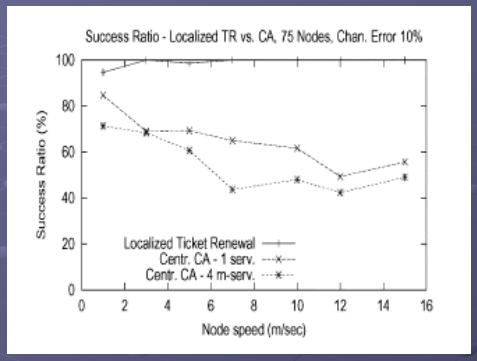
Performance Evaluation

- Communication Performance
  - Average delay
    - Average latency to successfully renew a ticket
  - Normalized overhead
    - Aggregate communication overhead over the success ratio

48

Success Ratio

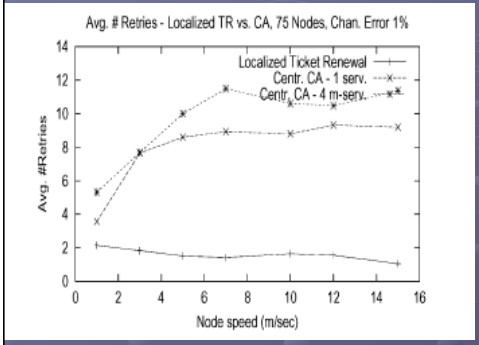


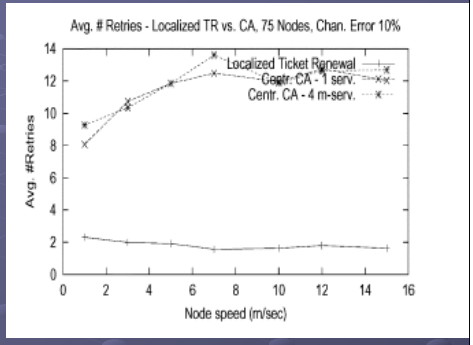


- Centralized solutions have lower success ratios than localized ticket renewal
- Increase in channel error does not appear to impact URSA's performance

49

Average # of Retries

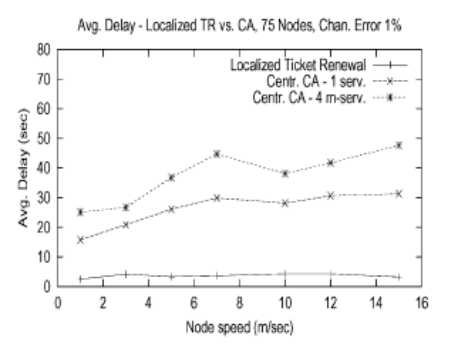


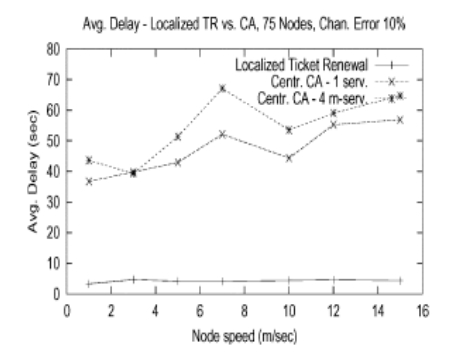


- URSA requires less avg. # of retries than centralized services
- Again, increase in channel error does not appear to impact URSA's performance

50

Average Delay

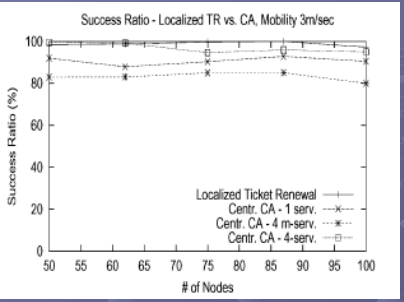


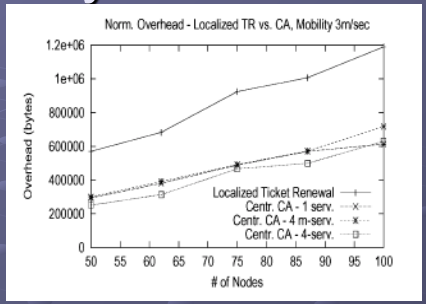


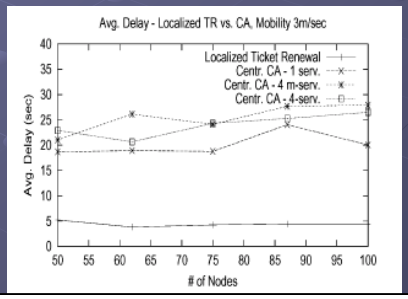
- URSA has lower avg. delay than centralized services
- Again, increase in channel error does not appear to impact URSA's performance

51

Scalability

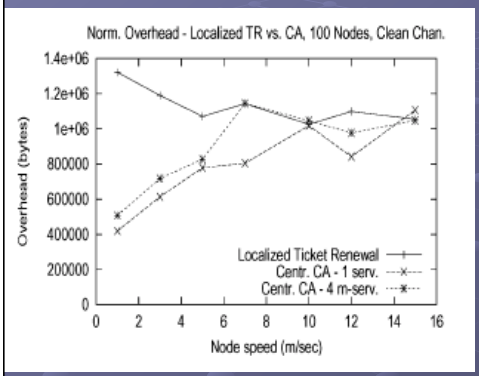


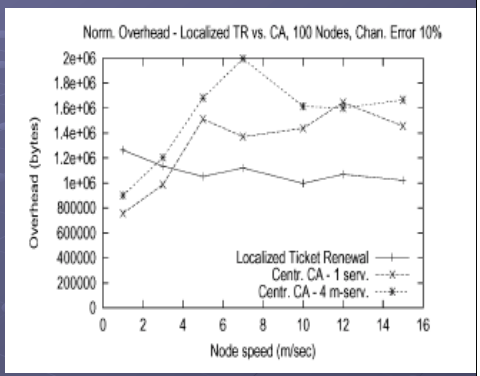




52

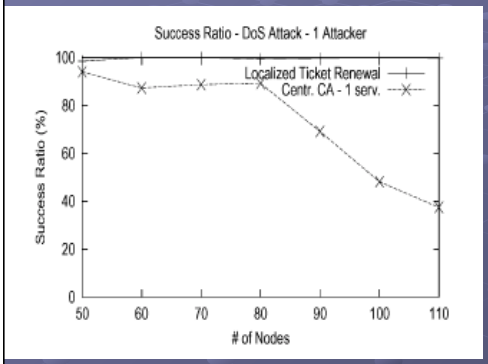
Communication Overhead

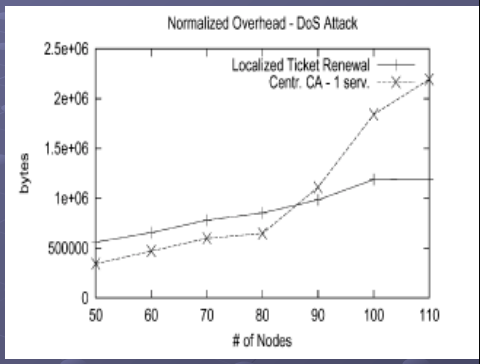




53

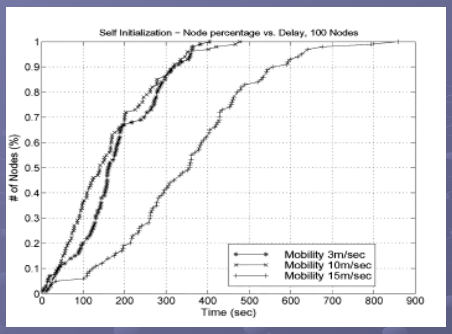
Resilience to Attack

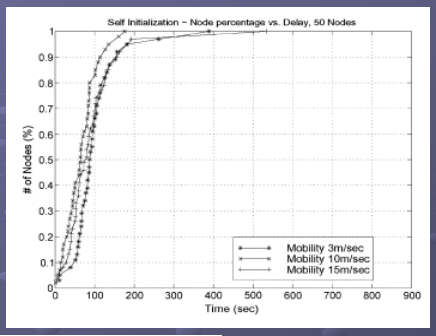


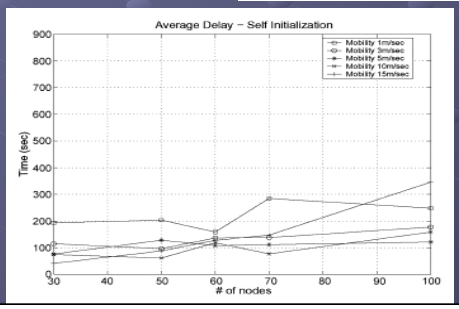


54

Self Initialization







55

Strengths

- Ticket services are fully localized to each node's one-hop neighborhood
  - Provides service ubiquity, eliminates single point of failures, and improves performance
- Self-initialization
  - A central authority is only needed to initialize x**k* nodes (2*k* in their simulations)
- Resilience to conspired attacks
  - < *k*-1 attackers (e.g. false accusations)
  - Mitigators: *k* good nodes and T_{cert} soft state

56

Extensions

- "The choice of specific detection mechanism is left to individual nodes"
  - Specific misbehavior detection mechanisms and signatures should be a network design requirement

57