CAR Approach for the Internet of Things

By Fadi Al-Turjman and Melih Gunay

Presented by Mak Kolybabi

Outline

We're going to follow along with the paper:

1. Introduction
2. Related Work
3. Models and Assumptions
4. Context-Aware Routing (CAR) Approach
5. Theoretical Analysis
6. Performance Evaluation
7. Conclusion
8. Criticism (bonus!)
9. Questions

Introduction

- Smart cities will have lots of different kinds of sensors, many moving
- Need to balance bandwidth usage vs. delivery delay
- We can spend CPU resources to find better routes, others do this

Introduction

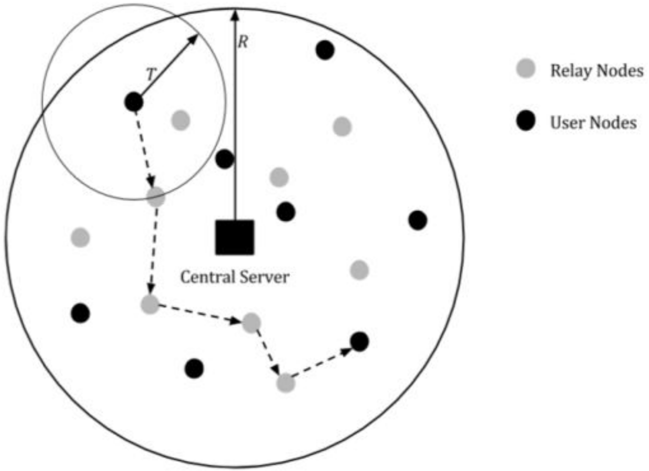
- This paper classifies nodes as
  - User nodes, which act as sinks and sources
  - Relay nodes, which transfer
- Other approaches randomly choose paths
- CAR approach centrally and dynamically chooses paths
  - Skype: tolerates no delays, some loss
  - Alarms: tolerates some delay, no loss
  - Temperature: tolerates delays, loss
- Cloud has the data needed to make routing decisions
  - Relay node properties
  - Application properties

Related Work

- *Lots* of work has been done on routing algorithms
- Other work does not address data delivery and delay management challenges together
- DSR and AODV are mentioned as contenders
    - DSR is good for small networks with low mobility
    - AODV is good for large networks, has large overhead
- Our comparison is going to use AODV

Models and Assumptions

- User nodes have applications and are mobile
- User nodes generate messages
- Relay nodes support some applications



- Modeled for hundreds of nodes
- Radii: T = Transmission, R = Mobility
- Uniform RNG used to choose everything
    - Start location, movement, source, destination, application

Context-Aware Routing Approach: Source

Algorithm 1 For Source Node S

1. **If** S has a new *data* msg & no route to D
2. **Then** forward a *setup* msg to the cloud server.
3. **If** S receives the *setup* response msg,
4. **Then** choose *best* path $\boldsymbol{p_i}$ and send the new *data* msg.
5. **If** S doesn't receive a response for a RS period,
6. **Then** go to line 2.

Context-Aware Routing Approach: Destination

Algorithm 4 For Destination Node D

1. **If** *next_neighbor* is not reachable towards S for the *Ack*,
2. **Then** send a *setup* msg back to the Cloud server and update $\boldsymbol{p_i}$.
3. **If** there exist path $\boldsymbol{p_i}$ and still active,
4. **Then** send the *data* msg (if any).
5. **If** no paths found
6. **Then** go to line 2.

Context-Aware Routing Approach: Cloud Server

Algorithm 3 For Intermediate Node $\boldsymbol{i}! = D$

1. **If** i receives *data* msg from the *source*,
2. **Then** use compatible *App* and forward *data* msg.
3. **If** *next_neighbor* is not reachable,
4. **Then** send a *setup* msg back to the cloud server and update $\boldsymbol{p_i}$.
5. **If** a new active path was established
6. **Then** check the compatible *App*, update RT and forward *data*.
7. **Else** buffer *data* **and** send another *setup* to the cloud server and update $\boldsymbol{p_i}$.

Context-Aware Routing Approach: Relay Node

Algorithm 2 For the Cloud Server

1. **If** no msg's are exchanged with server for *hello_interval* time units,
2. **Then** send a *hello* msg and update RT.
3. **If** *server* receives *setup* msg from S,
4. **Then** based on *App* requirements, send a list of recommended paths to S.
5. **If** *server* receives *setup* msg from D,
6. **Then** based on *App* requirements, send a list of recommended paths to S and D.

Theoretical Analysis

- We’ve not discussed how the Cloud Server recommends routes
- Mobile relay nodes are riskier because they might move somewhere undesirable after they’re recommended but before a packet is sent
- CAR aims to handle static and mobile relays

Performance Evaluation

- Simulation performed in MATLAB comparing CAR and AODV
- Built a packet-level simulator to measure key metrics
- Modeled networks as using 802.11
- Three types of random events:
    a. Information requests (reversal of Algorithm 1)
    b. Node enters or exits network
    c. Data transmission fails
- Poisson processes are used to generate events

Performance Evaluation: Performance Metrics

What we're looking at:

1. Average End-to-End Delay
    - Average time that each data packet spends in the network
2. Average Queuing Delay
    - Average time that each data packet spends in a relay node's queue
3. Average Dropped Packets
    - Average percentage of transmitted data packets failing to reach their destination

Performance Evaluation: Performance Metrics

What we’re varying:

1. Size of network
   - In terms of relay nodes
2. Network load
   - In terms of request arrival rate

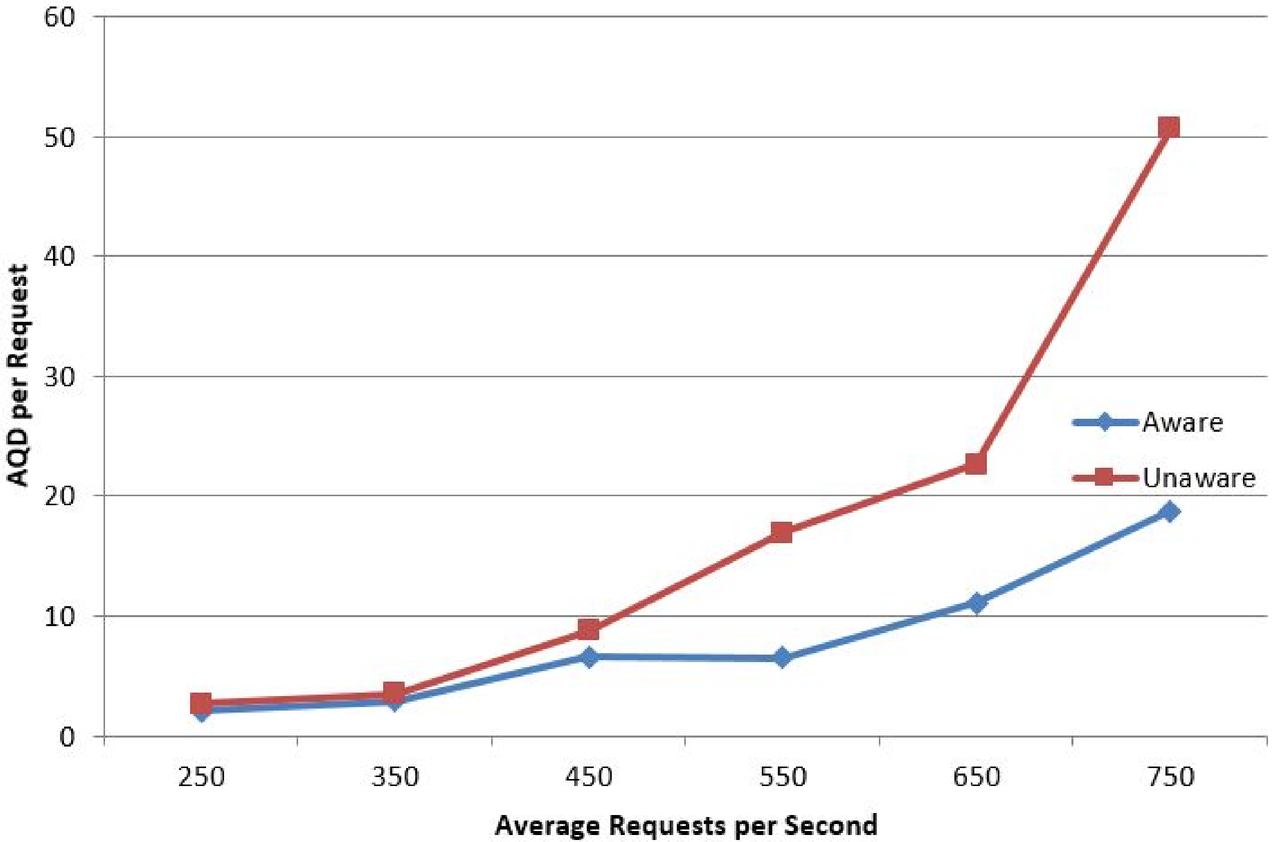
Two simulations are used, each with one aspect constant and the other varying

Performance Evaluation: Performance Metrics

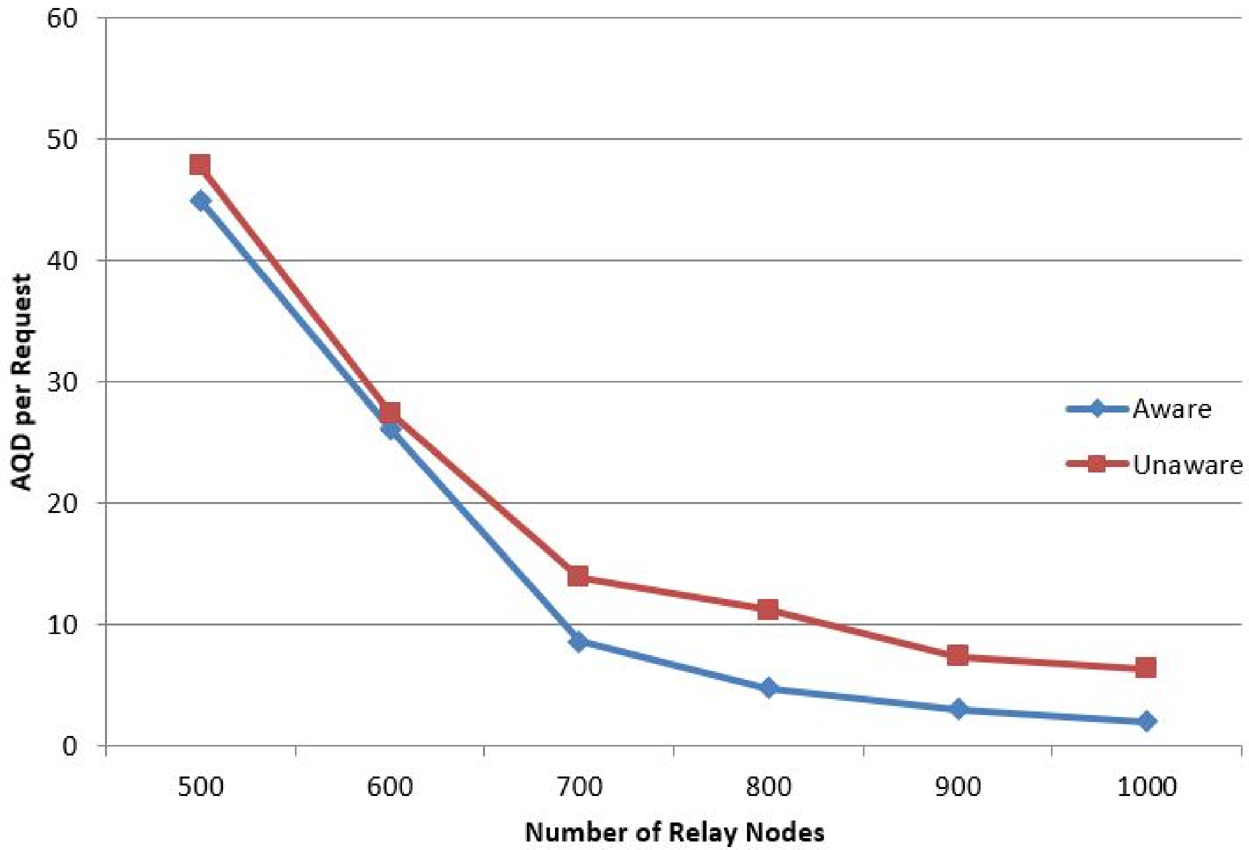
What we’re keeping constant:

1. Number of user nodes
2. Network radius
3. Transmission radius

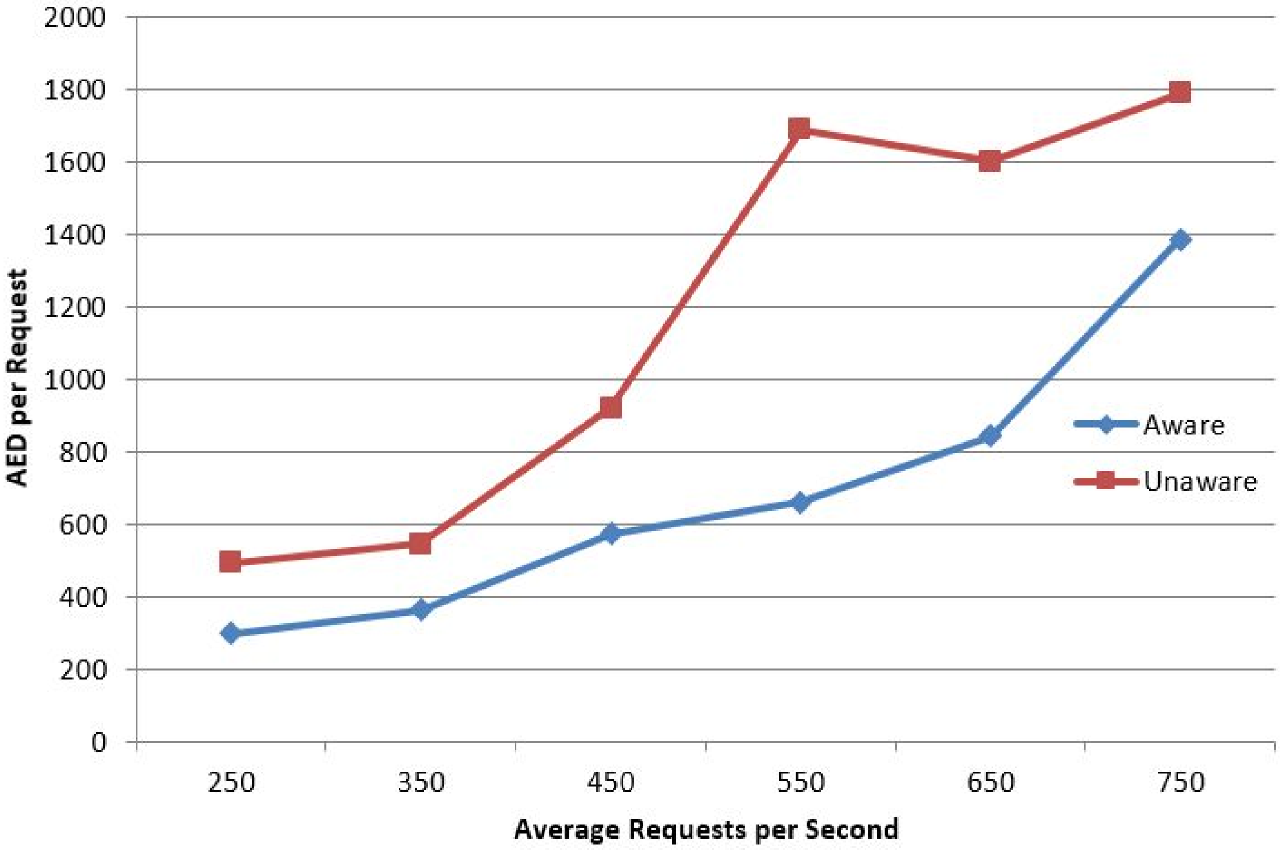
Performance Evaluation: Simulation Results



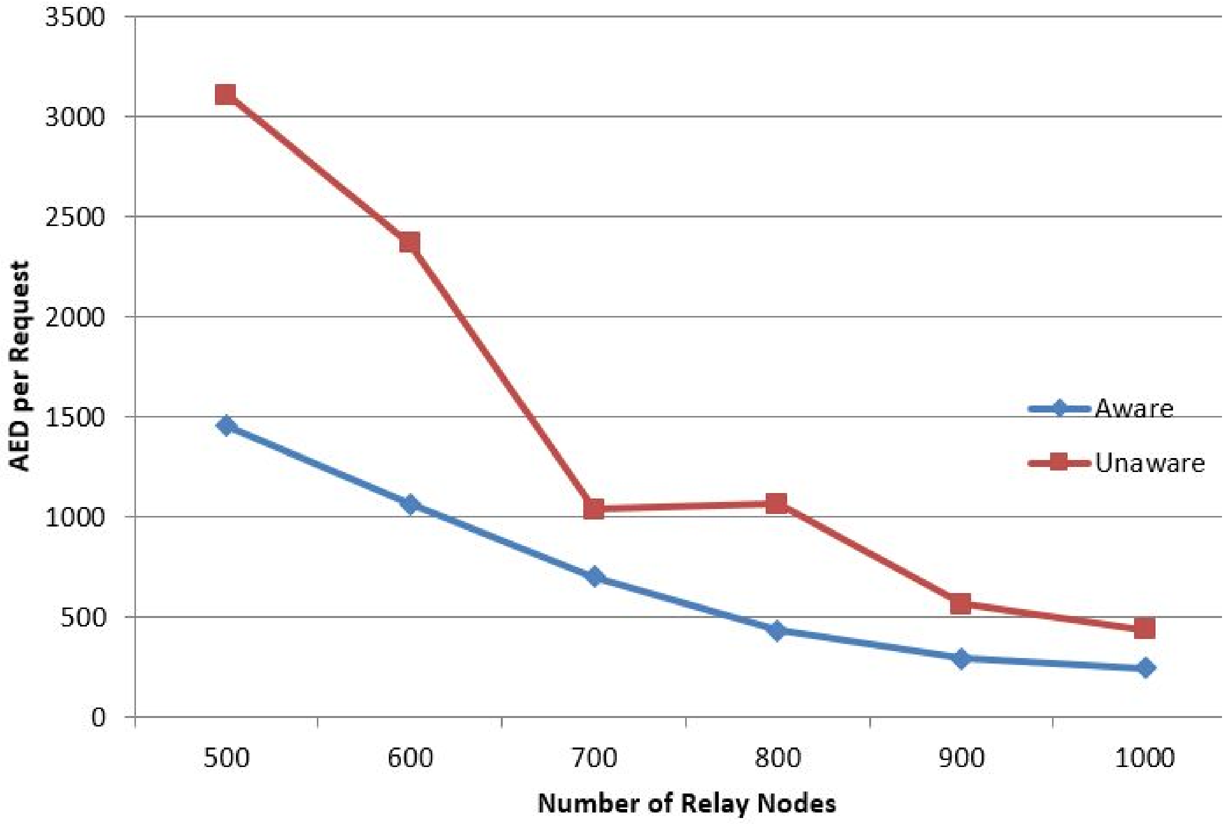
Performance Evaluation: Simulation Results



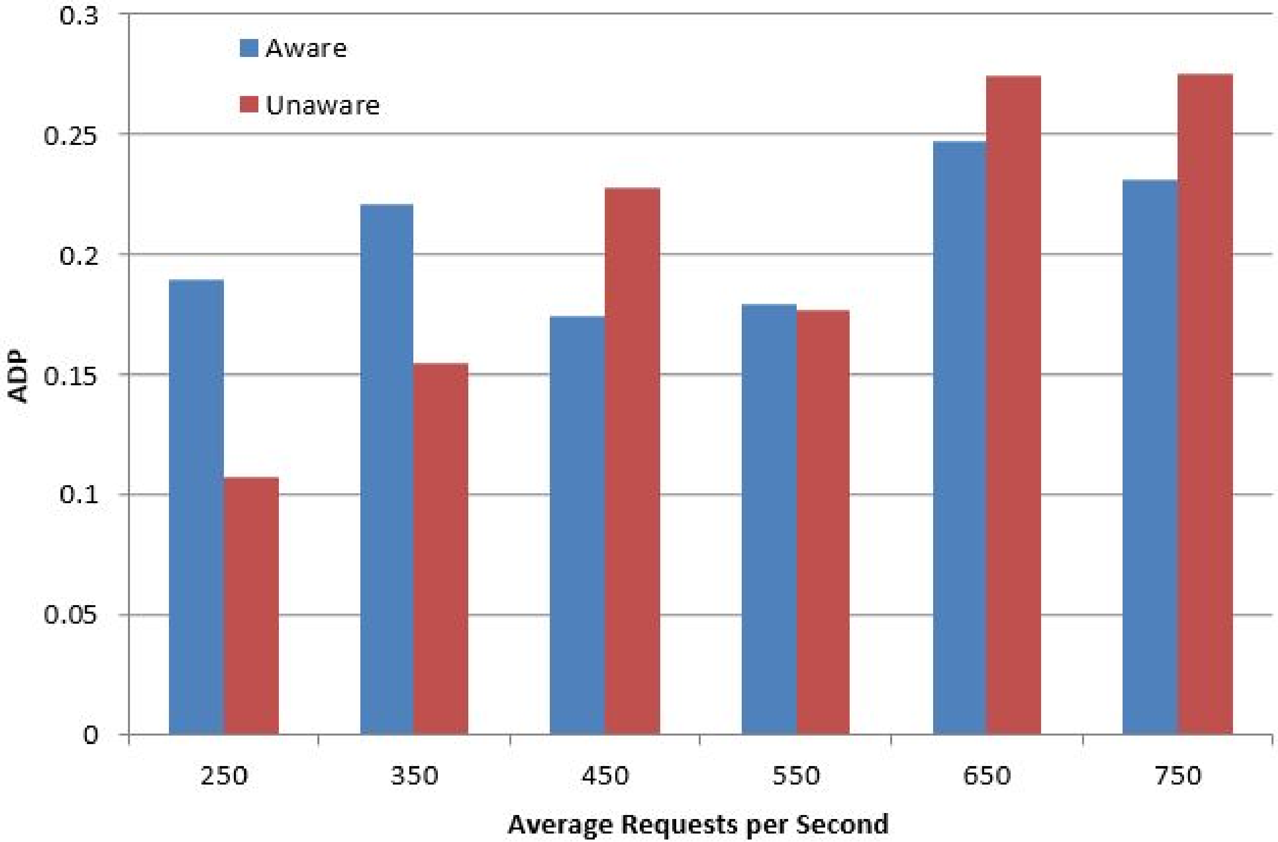
Performance Evaluation: Simulation Results



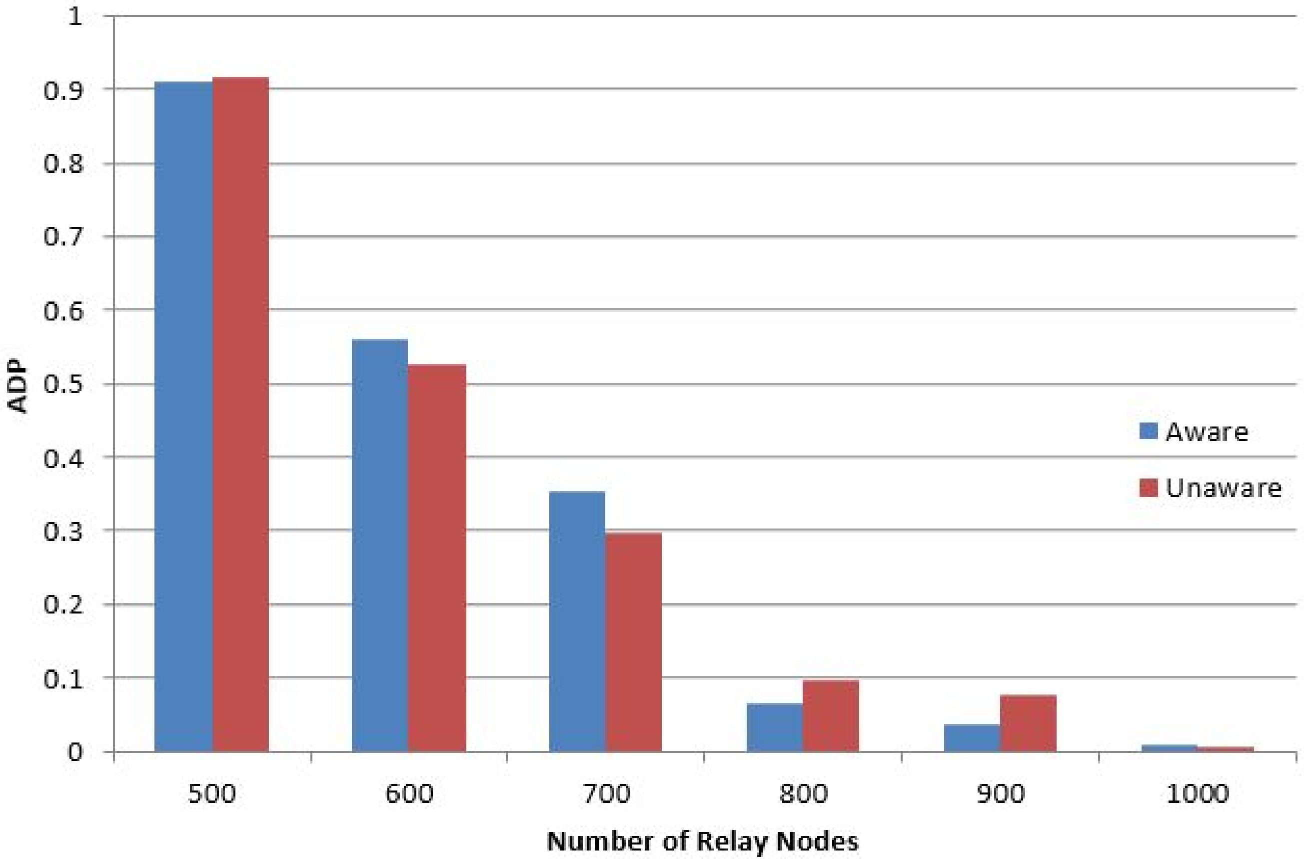
Performance Evaluation: Simulation Results



Performance Evaluation: Simulation Results



Performance Evaluation: Simulation Results



Conclusion

- Goal was to compare CAR and non-CAR network in terms of:
  - Average data request in-queue delay
  - In-network delay
  - Drop rate
- CAR outperformed non-CAR:
  - As the network grew in number of nodes
  - As requests grew in frequency

Criticism

- Only simulation was done, no actual implementation
- Only compared against a single other algorithm, known to have high overhead
- No discussion of resource needs of any device, especially central processing server
- No discussion of sensors moving in groups, such as vehicles or people
    - Affects choosing mobile relays if the mobility is only relative to the central processing server
- No discussion of method of choosing relays other than shortest-path
    - Despite talking about static vs mobile relay choice early in the paper
- No discussion of saturation of relays near central processing server
    - *Every* node might need to talk to it *any* point
- No discussion of considering paths sharing collision domains for high-bandwidth applications
- Unclear what role the ‘cloud’ has since the central processing server appears entirely local
    - Does it have a backhaul uplink to a large database and lots of CPU power?
- What if sources want to multicast to any available sink?
    - They talk about redundancy, but never address it in terms of user nodes

Questions?