VShare: A Wireless Social Network Aided Vehicle Sharing System Using Hierarchical Cloud Architecture

Yuhua Lin and Haiying Shen
Dept. of Electrical and Computer Engineering
Clemson University, SC, USA
Outline

• Introduction
• System Design
  • Overview of Vshare
  • Design of VShare
• Performance Evaluation
• Conclusions
Introduction

Carpool commuting: multiple travelers with similar schedules and itineraries share one vehicle

Introduction

Carpool commuting: benefits

• Alleviate traffic congestion, parking space tension
• Mitigate air pollution from vehicle emissions
• Privilege to use high occupancy vehicle (HOV) lanes

Introduction

How to match carpoolers?

Build carpool lanes in airports, bus stops:

wait in queues, make carpools spontaneously, first-come-first-service basis

- Cannot schedule carpooling in advance
- Small-scale user population in designated locations
Introduction

How to match carpoolers?

Utilize prior user mobility knowledge:

portable devices (e.g., smartphones) to collect individual trips, identify carpoolers based on travel routes and mobility models

• Cannot adapt to real time scenario
Introduction

How to match carpoolers?

Dynamic carpooling system

riders and drivers provide preferred travel information, calculate carpooling schedules based on objectives

• Using a centralized server generates long computation latency
Introduction

Our proposed method: VShare

- Identifies carpoolers through the wireless social network
- Uses a hierarchical cloud server architecture to identify carpoolers

Advantage: matching latency is reduced
Outline

• Introduction
• System Design
  • Overview of Vshare
  • Design of VShare
• Performance Evaluation
• Conclusions
Overview of VShare

Step 1:
- request broadcasted to neighbors in nearby locations
- neighbors check travel schedule, respond to request

Step 2:
- cloud servers form a hierarchical architecture
- requests with the same departure location and destination location are stored in the same server

Goal: match carpoolers with short latency
Transformation of Travel Requests

**Travel request**

- **Departure**: JFK airport
- **Destination**: 5th Ave
- **Travel time**: 9:30am
- **Max wait time**: 30min

**Address code**

- 000: Park Ave
- 001: 5th Ave
- 002: JFK airport
- 003: 1st Ave

**Numerical string**

<table>
<thead>
<tr>
<th>Dep ID</th>
<th>Dest ID</th>
<th>Time</th>
<th>Max wait time</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>001</td>
<td>0930</td>
<td>30</td>
</tr>
</tbody>
</table>

**Dep ID**: address code of departure location  
**Dest ID**: address code of destination location  
**Time**: departure time  
**Maximum wait time**
Matching of Potential Carpoolers: Two carpoolers

1. Compare Dep ID and Dest ID sequentially, unmatched if different Dep ID or Dest ID
2. Calculate matching score

\[ m_{ij} = 1 - \frac{(t_i - t_j)}{w_j} \]

- \( t \): departure time
- \( w_j \): maximum wait time
- \( m_{ij} \): degree of how long one needs to wait for another
Matching of Potential Carpoolers: Multiple Carpoolers

Input: a list of travel requests, $R = (r_1, r_2, ... r_u)$
Output: a carpool

Algorithm 1
• Select one candidate from R at a time
• Calculate new carpool travel schedule
• Check if wait time of each passenger is within his/her maximum wait time
• Add candidate to the carpool if satisfied
Matching Via the Wireless Social Network

TTL: maximal hops a travel request is forwarded
- Sends a request to neighbors with TTL=2
- Receives multiple replies from its neighbors
- uses Algorithm 1 to make a carpool from multiple passengers
- Starts instant conversation with carpoolers
Matching Via Hierarchical Cloud Architecture

- Centralized server (CServer): distributes requests
- Departure managers (DepM): handles requests with the same departure ID
- Destination managers (DesM): handles and stores requests with the same departure ID and destination ID

Three-level hierarchy structure
Matching Via Hierarchical Cloud Architecture

• CServer passes a new request to DepM that is responsible for the request’s departure ID

• DepM forwards the request to DesM that is responsible for the request’s destination ID

• DesM only needs to match the travel time and maximum wait time using Algorithm 1
Outline

• Introduction
• System Design
  • Overview of Vshare
  • Design of VShare
• Performance Evaluation
• Conclusions
Performance Evaluation: Settings

• Simulation using Cab mobility trace dataset [3]
  – GPS coordinates of 536 taxis over 30 days in San Francisco Bay Area
  – Average # of travel requests/day: 14000
  – Each taxi’s capacity is 4

• Comparison methods
  – Cloud: user travel requests are gathered and processed by a centralized cloud server
  – No-Sharing: each user occupies a single cab

Performance Evaluation: Results

• Average travel expense
  Setting: single trip costs [20,40] dollars, evenly split among carpoolers

• Observation: VShare < No-Sharing, average travel expense drops as the numbers of users increases
• Reason: users are more likely to be potential carpoolers when user density is high
Performance Evaluation: Results

- Average matching latency

Variants of Cloud systems:

- **Cloud-D**: travel requests are stored in random cloud servers, matching carpoolers by a centralized server

- **Cloud-C**: a centralized server stores all travel requests, matching carpoolers

- **Cloud-H**: travel requests stored in hierarchical cloud architecture
Performance Evaluation: Results

- **Average matching latency**

![Graph](image)

(a) Performance with different number of users.  
(b) Performance with different number of cabs.

- **Observation:** Cloud-D > Cloud-C > Cloud-H > VShare
- **Reason:** VShare first matches carpoolers among nearby users using the wireless social network within a short latency; hierarchical cloud architecture stores requests with the same departure and destination locations in the same server;
Performance Evaluation: Results

• Success rate of catching a taxi within maximum wait time

(a) Performance with different number of users.
(b) Performance with different number of cabs.

• Observation: VShare > No-Sharing
• Reason: multiple users heading to the same destination can share one taxi. Given the same number of taxis, more passengers are transported.
Performance Evaluation: Results

- Number of taxis needed to transport all users within their maximum wait times

- Observation: VShare < No-Sharing
- Reason: each user in No-Sharing takes one cab; users in Vshare identify carpoolers nearby and share cabs with each other
Outline

• Introduction
• System Design
  • Overview of Vshare
  • Design of VShare
• Performance Evaluation
• Conclusions
Conclusion

- **VShare**: dynamic vehicle sharing system
  - Leverages the wireless social network and hierarchical cloud server architecture
- **Trace-driven simulations show:**
  - Reduce user travel expense
  - Reduce carpool matching latency
  - Increase success rate of catching a taxi
  - Reduce # of taxis needed to transport a specific # of users
- **Future work**: identify carpoolers with different departure and destination locations
Thank you!

Questions & Comments?

Haiying Shen
shenh@clemson.edu
Associate Professor
Electrical and Computer Engineering
Clemson University