Mobile Distributed Computing in the Cloud (NSF CNS-1409523)

New Jersey Institute of Technology

Cristian Borcea (borcea@njit.edu), Narain Gehani (narain.gehani@njit.edu),
Reza Curtmola (reza.curtmola@njit.edu), Xiaoning Ding (xiaoning.ding@njit.edu)

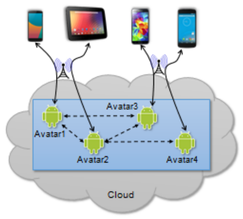
Background and Motivation

- Mobile distributed computing enables direct collaboration among mobile users
  - Novel apps based on huge amounts of data generated every day by mobile devices
  - Users are willing to share data with certain communities
- Mobile distributed computing over ad hoc and Internet-based P2P networks suffers from
  - Poor availability
  - Limited energy and computational resources
  - High latency
  - Costly bandwidth

Avatar System Overview

Avatars in the Cloud Augment Mobile Devices

- An avatar is a per-user, always-on software entity in the cloud
  - Executes app components on behalf of user's mobile devices
- Avatars execute mobile distributed apps
  - Collaborate with each other
  - Collaborate with mobile devices
- Inherent features of mobile distributed computing supported by avatars:
  - High availability for user apps/devices
  - Energy savings on mobile devices
  - Low latency/response time for apps
  - Isolation and effective resource management for mobile apps in the cloud



Applications and Benefits

- Finding people of interest in the crowd
  - Example app: Lost Child - finding a missing child by searching photos taken by nearby mobile users during same time period
  - Benefits: fast response time, energy savings on mobiles
- Healthcare and wellbeing:
  - Example app: monitoring the spread of epidemic diseases
  - Benefits: improved availability and scalability
- Collaborative mobile sensing and filtering:
  - Example app: mobile people-centric sensing
  - Benefits: fast response time, energy savings on mobiles
- Mobile multiplayer gaming:
  - Example app: Quake 3 (first person shooter game)
  - Benefits: improved user experience, energy savings on mobiles, reduced wireless bandwidth usage

System Challenges

- Programmability: provide programming abstractions to shield programmers from complex distributed execution environment
  - Environment: distributed and synchronized (mobile, avatar) pairs
  - Apps have multiple options to divide components between mobiles and avatars
- Storage/Sensor Mechanisms: provide scalable and energy efficient cloud services for mobile cloud apps accessing mobile files and sensors
  - Distributed computing across avatars and mobile devices of multiple users
  - Accesses to files and sensors must be enabled and managed dynamically
- Privacy: efficiently storing data and executing apps while protecting sensitive data and results from cloud providers
  - Data needs to be used by Avatars: not good enough to just encrypt the data

Main Research Contributions and Ongoing Work

- Moitree middleware to support the development and execution of Avatar applications
- OFS: an overaly file system to support file I/O in cloud-assisted mobile apps
- Sentio: a Sensor Virtualization Middleware to facilitate mobile cloud apps to access sensors
- P2F2: a system that allows cloud-based photo matching while preserving the privacy of the photos from the cloud provider
- FaceDate: a mobile, location-based app developed for testing Avatar system and user study
- **Ongoing/future work**
  - User study with FaceDate and Moitree with students on our campus
  - CASINO: Collaborative Sensor Driven Offloading System for offloading computation to the cloud in a collaborative way.

Moitree Middleware

Provides an API that implements a high-level distributed programming model for developing Avatar apps

- Communication end-points are mapped transparently to mobile-avatar pairs
- Common app execution environment on mobiles and avatars allows seamless offloading of app components
- Context-aware group-based abstractions to manage user collaboration
- Abstract communication channels to offload mobile-to-mobile communicating to the cloud

Programming Abstraction: Context-aware Groups

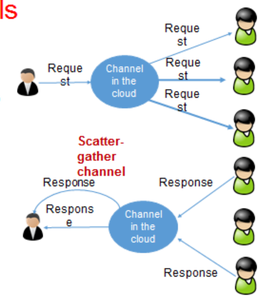
- Human collaboration breaks down into groups and sub-groups
- Moitree's programming model structures user collaboration within apps over context-aware groups
  - Formed and maintained dynamically based on context parameters
  - Structured hierarchically

Lost child app: group defined by location and time (find people who took photos in this location since 1pm)

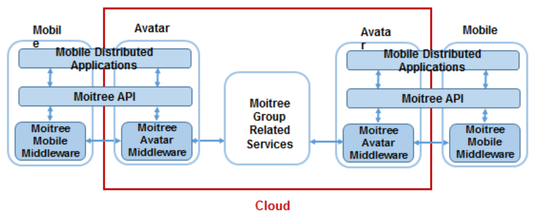
Free parking spot app: hierarchical groups of drivers (hierarchy: city, district, block, street)

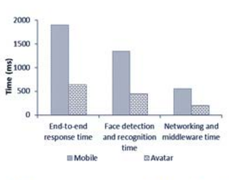
Programming Abstraction: Communication Channels

- Communication channels
  - Communication abstraction for group collaboration
  - Mobile-mobile communication offloaded to the cloud
  - Hides programming complexity for distributed communications
- Four asynchronous communication channels
  - broadcast
  - point2point
  - anycast
  - scatter-gather

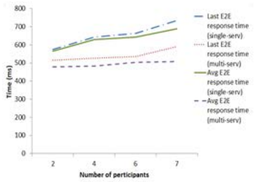


Moitree Architecture





Moitree can improve latency by 3.5 times compared to mobile only solution for this experiment



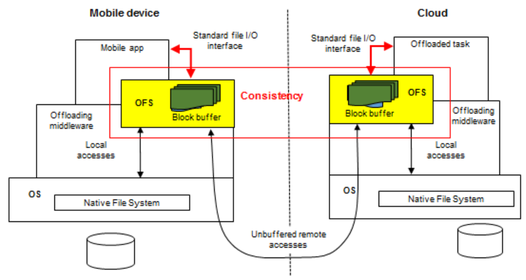
Application scales well for number of participants

Overlay File System

OFS: An Overlay File System for Mobile Cloud Applications

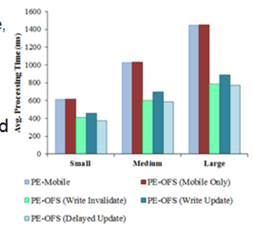
- To support file I/O in cloud-assisted mobile apps:
  - Read and write files on both mobile and cloud
  - Require strong consistency
  - Long I/O latency due to transferring the file over network
- Application-level file system for cloud assisted mobile apps
  - Doesn't need system-wide management
  - Can work with any native file system
  - Doesn't incur costly context switches
- Advantages:
  - Strong consistency (delayed-update policy)
  - Location transparency (file session management)
  - Low overhead (low latency file access, low overhead consistency maintenance)
  - Ease of deployment (application level)

Overall System Architecture



Case Study and Evaluation w/ a Photo Enhancement App

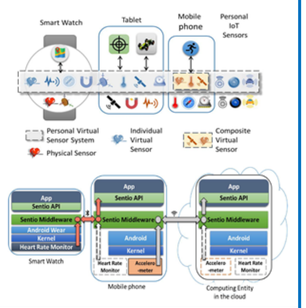
- The app processes the photos selected by user
  - applying color reduction, adding salt noise, applying sharpening filters, and adding watermarks
- Mobile only (PE-Mobile): low performance
- Photo processing offloaded to the cloud
  - Photos saved in Dropbox for accesses from the cloud: crash/correct results
  - OFS used to support accessing photos from the cloud: correct results w/ lower processing time



Sensor Virtualization

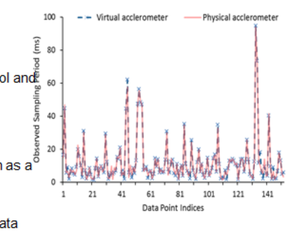
Sentio: A Sensor Virtualization Middleware

- Two objectives:
  - To allow tasks offloaded in the cloud by Avatar (or other offloading systems) to access sensors on mobile devices seamlessly
  - To allow apps running on any mobile personal device to seamlessly access sensors located on other mobile personal devices.
- Personal virtual sensor system (PVSS)
  - A set of virtual sensors built with physical sensors
    - Physical sensors in local/remote smart devices and/or IoT devices
    - Multiple virtual sensors can form a composite sensor to support data aggregation and fusion
- Sentio provides
  - A high level API to create and access virtual sensors
  - A distributed middleware to handle low-level communication and sensor management tasks



Case Study and Evaluation

- Android-based prototype
- Test bed: smart phones and smart watches.
- Apps
  - New apps: SentioApp and SentioFit.
  - Open source games modified: Tilt Control and Space Shooter.
- Various testing cases with the apps.
  - E.g., ran the Space Shooter game on a mobile device and used a virtual accelerometer mapped to a smart watch as a game controller.
- Results show that Sentio can
  - guarantee real-time delivery of sensor data
  - impose a minimal overhead.

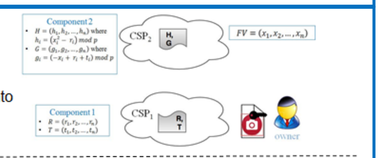


Privacy-Preserving Face Finder

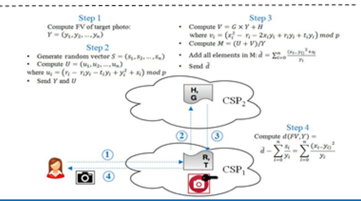
- Privacy-Preserving Face Finder (P2F2) allows
  - Cloud-based photo matching
  - Preserving the privacy of photos from the cloud provider
- Designed for face matching using Local Binary Patterns (LBP)
- Guarantees
  - Match accuracy
  - Photo privacy
- Assumes
  - Honest-but-curious CSPs
    - Follow the protocol
    - Try to learn additional information
  - CSPs do not collude or share the data stored by owners

Key Ideas

Setup Protocol: split feature vectors of the photos into two components and store them into two non-colluding CSPs

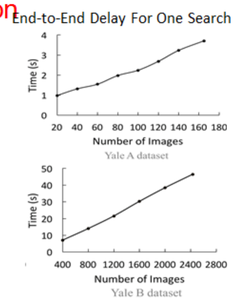


Searching protocol: use a privacy-preserving protocol to compute the Chi-square distance between the feature vectors of two photos.



Security Analysis and Performance Evaluation

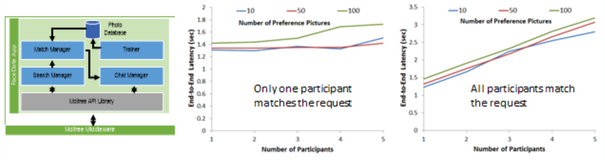
- Image Privacy
  - Image encryption
    - CSPs cannot access the content of images
    - Encrypted with a semantically-secure encryption scheme
  - Security analysis of data stored at and exchanged between CSPs
    - The cloud cannot derive any useful information from the variables introduced by the system
- Match Accuracy
  - Immediate from step 3 of the Search protocol
  - Achieve the exact Chi-square distance that is obtained over unencrypted data



FaceDate Mobile Cloud App

- FaceDate is a mobile, location-based app that matches people based on their face preferences in real-time.
- Developed mainly to test the Avatar system
- We plan to use it for a large-scale, on-campus user study of Avatar

Implementation and Evaluation



- End-to-end latency is almost constant when only one participant responds
- End-to-end latency increases sub-linearly when all participants respond due to parallel execution at participants and serial execution at the requestor.