SWAPEROO

A Simple Wallet Architecture for Payments, Exchanges, Refunds, and Other Operations

Neil Daswani, Dan Boneh, Hector Garcia-Molina, Steven Ketchpel, Andreas Paepcke
Stanford University

Goals

- Desirable wallet properties / features
- Define wallet interaction model
- Define Clean APIs for wallet and its components
- Build Prototype

Wallet Features

- Extensible: support multiple existing and newly developed instruments and protocols
- Non-Web-Centric: can be implemented in non-web environments; extensibility across devices
- Symmetric: common services across commerce applications; extensibility across commerce applications
- Client-Driven: user initiates all operations, including wallet invocation

An Example

- Session Initiation
  - Dilbert -> Amazon.com
- Instrument Class Negotiation
  - Dilbert: MasterCard, PonyCash, CyberCoin
  - Amazon.Com: MasterCard, VISA, CyberCoin
  - ==> MasterCard, CyberCoin

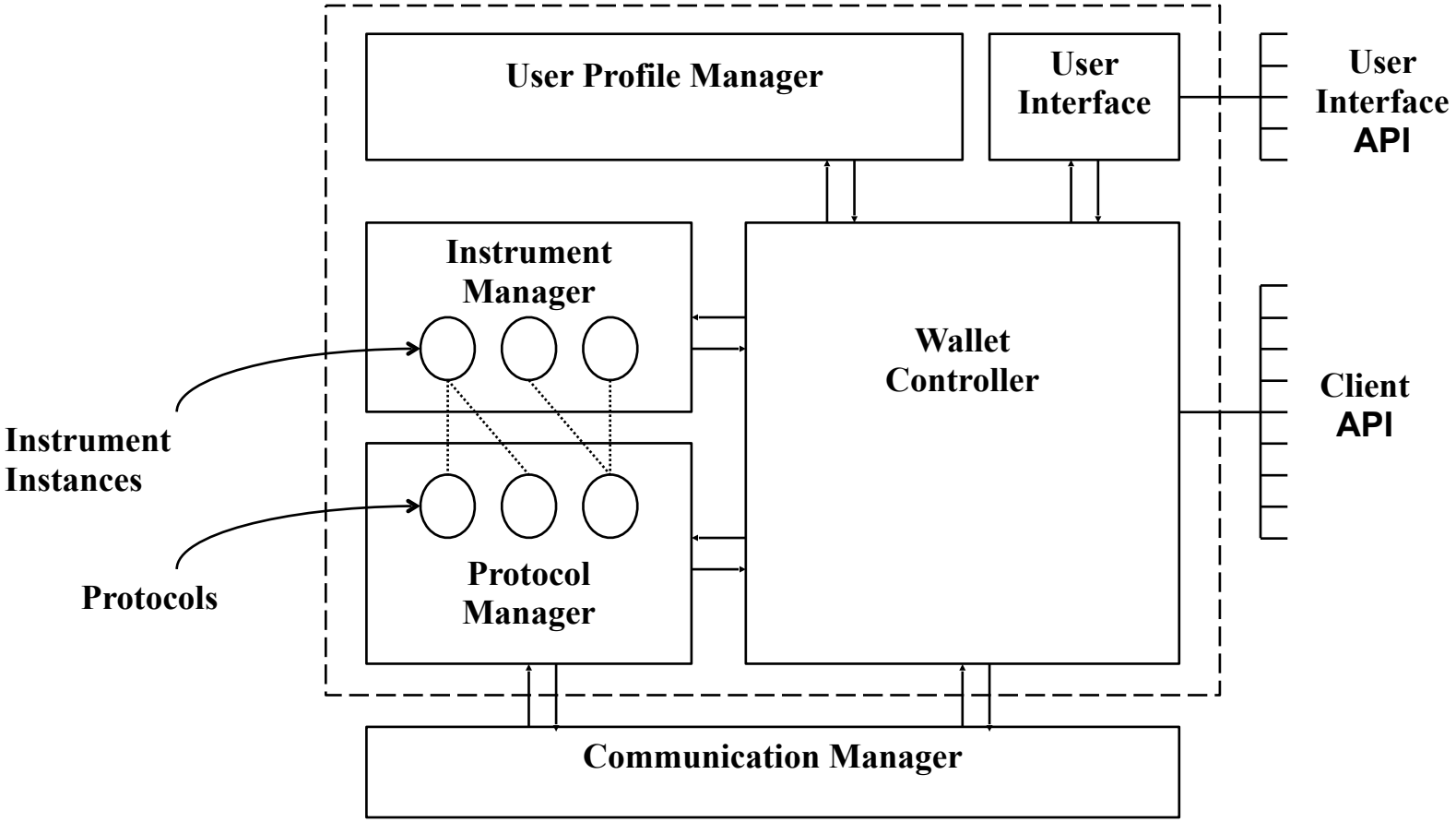
An Example

- Protocol Negotiation for MasterCard
  - Dilbert: SET (2KP)
  - Amazon.Com: SET (2KP), SET (3KP), or Unencrypted
  - ==> SET (2KP)
- Protocol Selection: SET (2KP)
- Available Operations: PAY, CREDIT

An Example

- Instrument Instance Selection:
  Dilbert's Citibank MasterCard
- Transaction Execution
  - SET (2KP) PAY
- Close Session

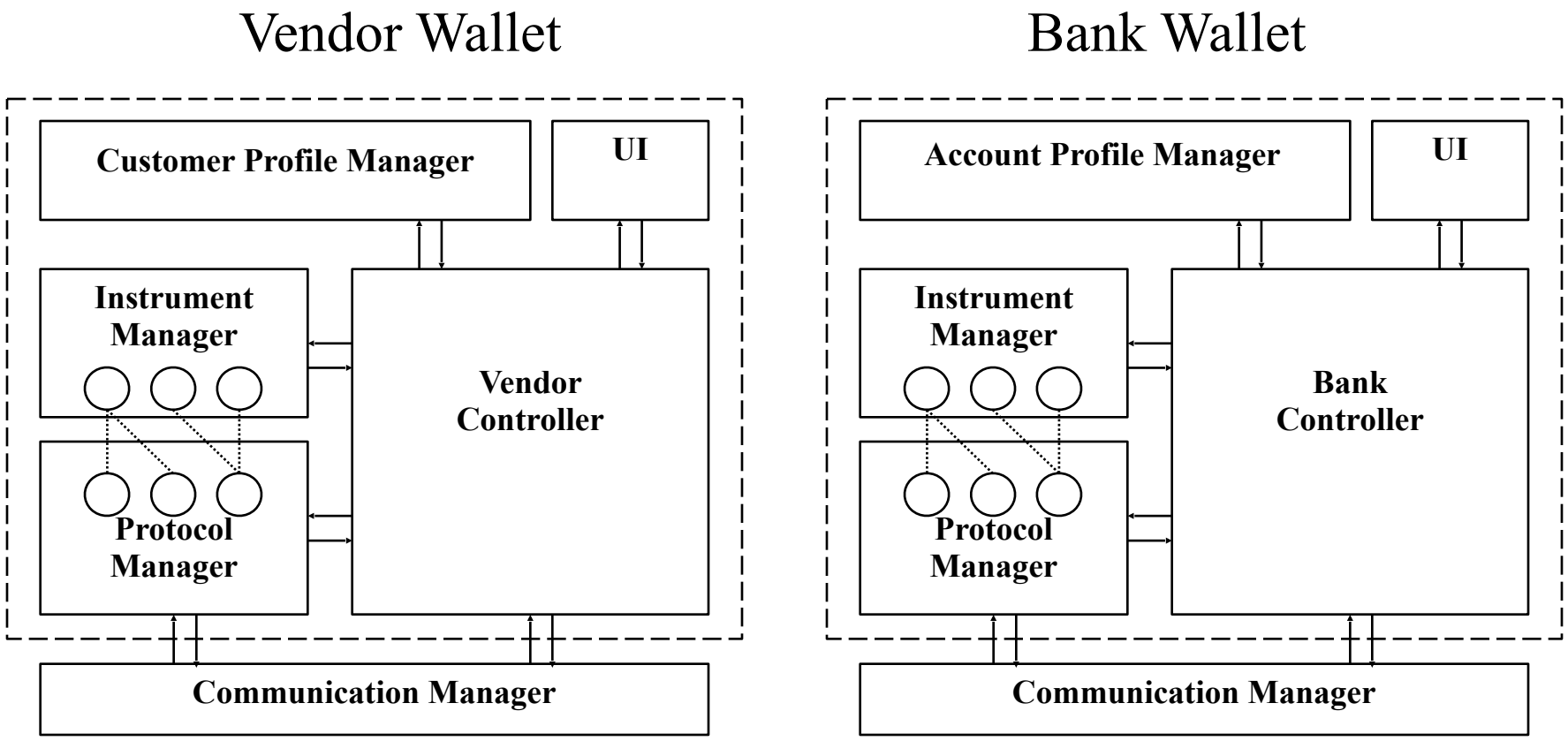
SWAPEROO Architecture



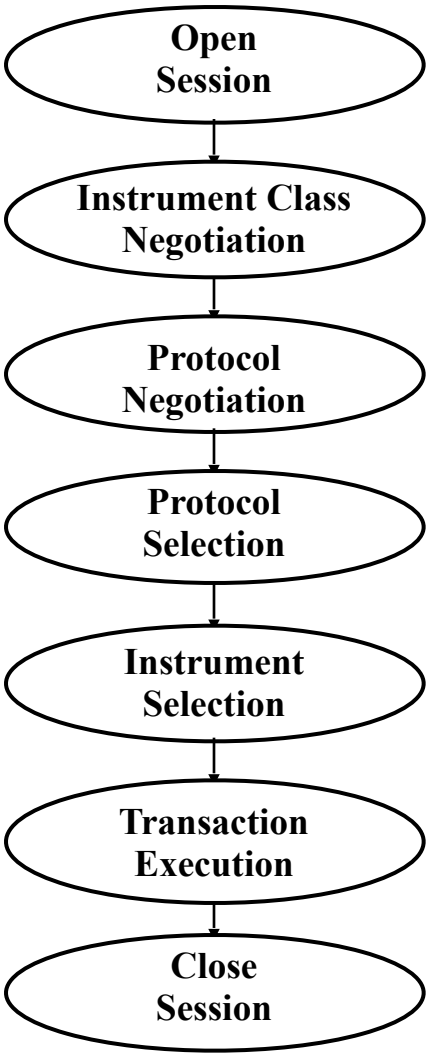
Function Descriptions

- Instrument Manager: encryption of instruments
- Protocol Manager: protocol invocation
- Communication Manager: low-level, synchronous messaging
- User Profile Manager: stores access control information
- Wallet Controller: coordinates wallet operations & enforces access control

Symmetric Vendors/Banks

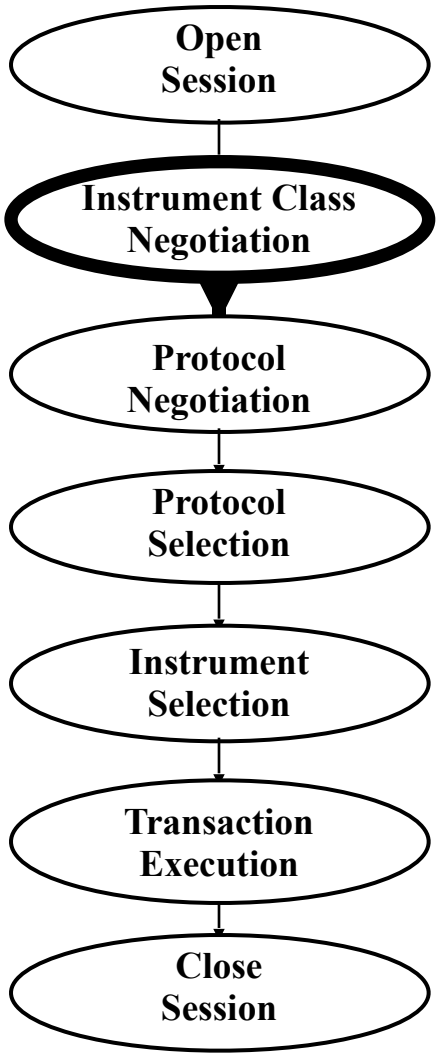


Wallet Interaction Model



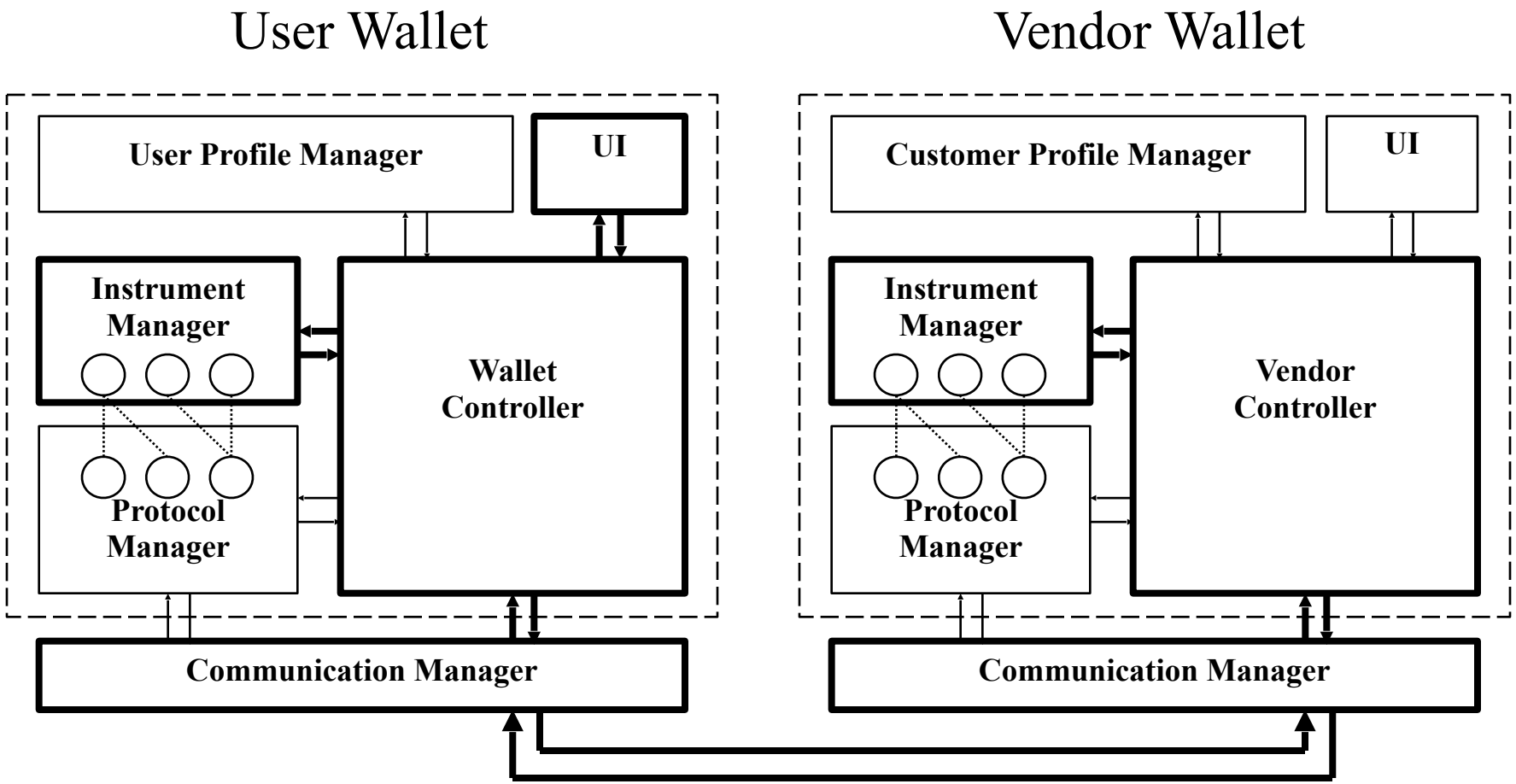
- Open Session
- Instrument Class Negotiation
- Protocol Negotiation
- Protocol Selection
- Instrument Selection
- Transaction Execution
- Close Session

Wallet Interaction Model

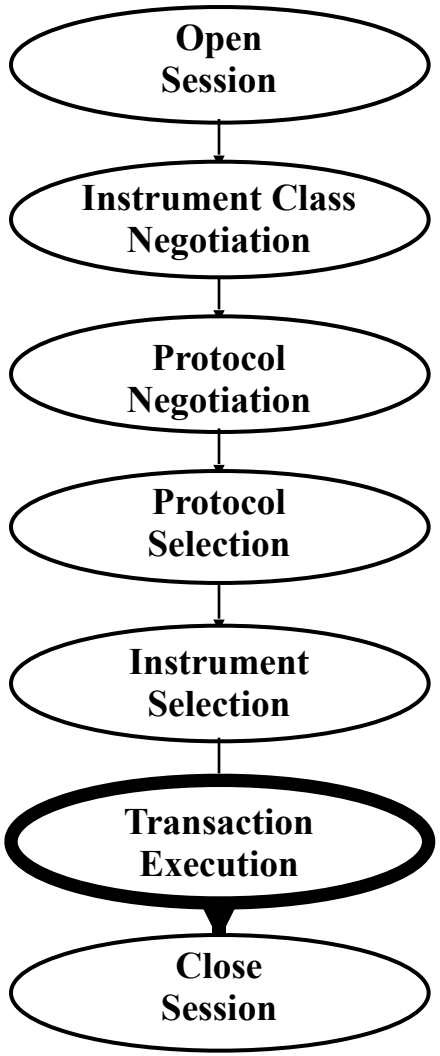


- Open Session
- **Instrument Class Negotiation**
- Protocol Negotiation
- Protocol Selection
- Instrument Selection
- Transaction Execution
- Close Session

Instrument Class Negotiation

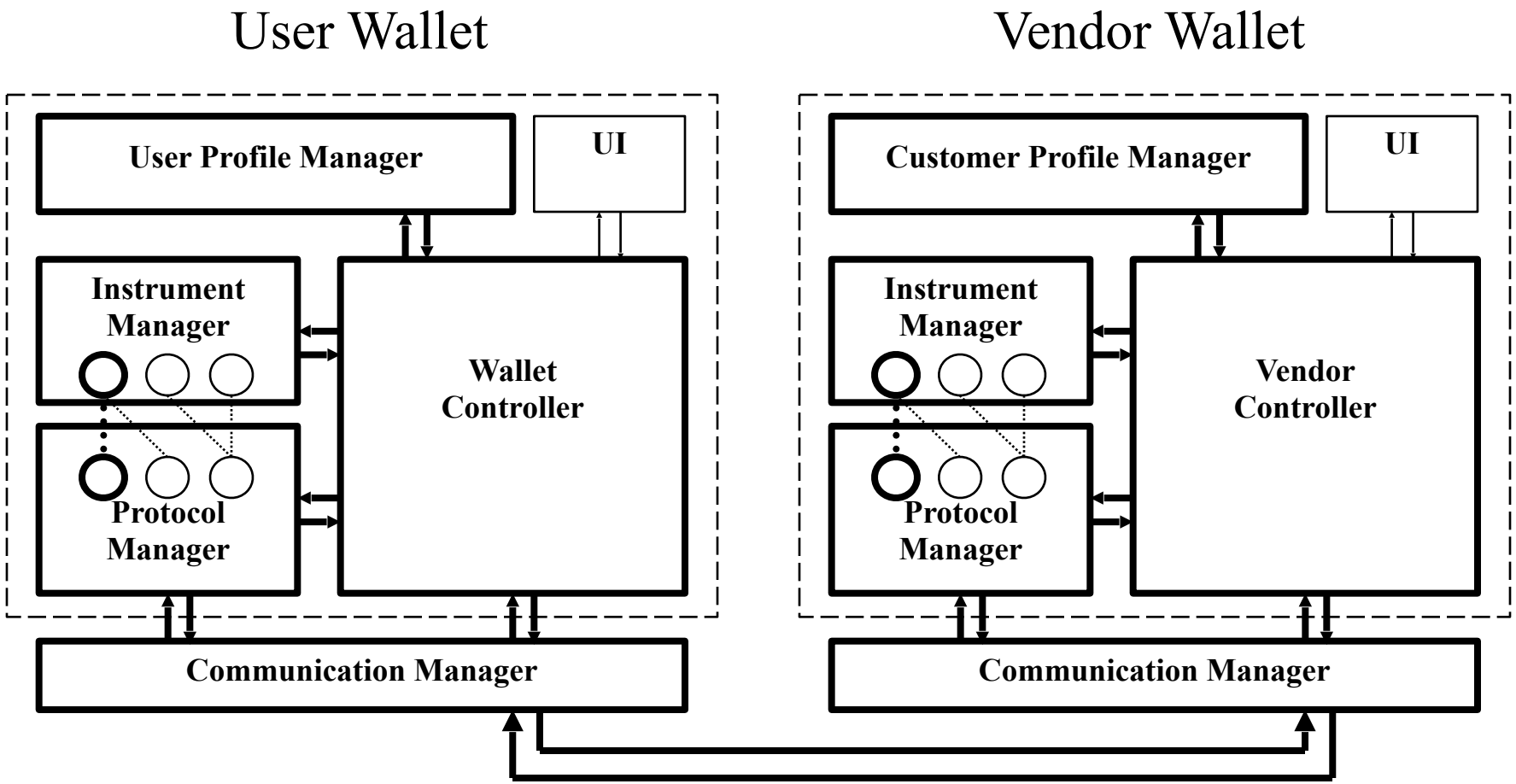


Wallet Interaction Model

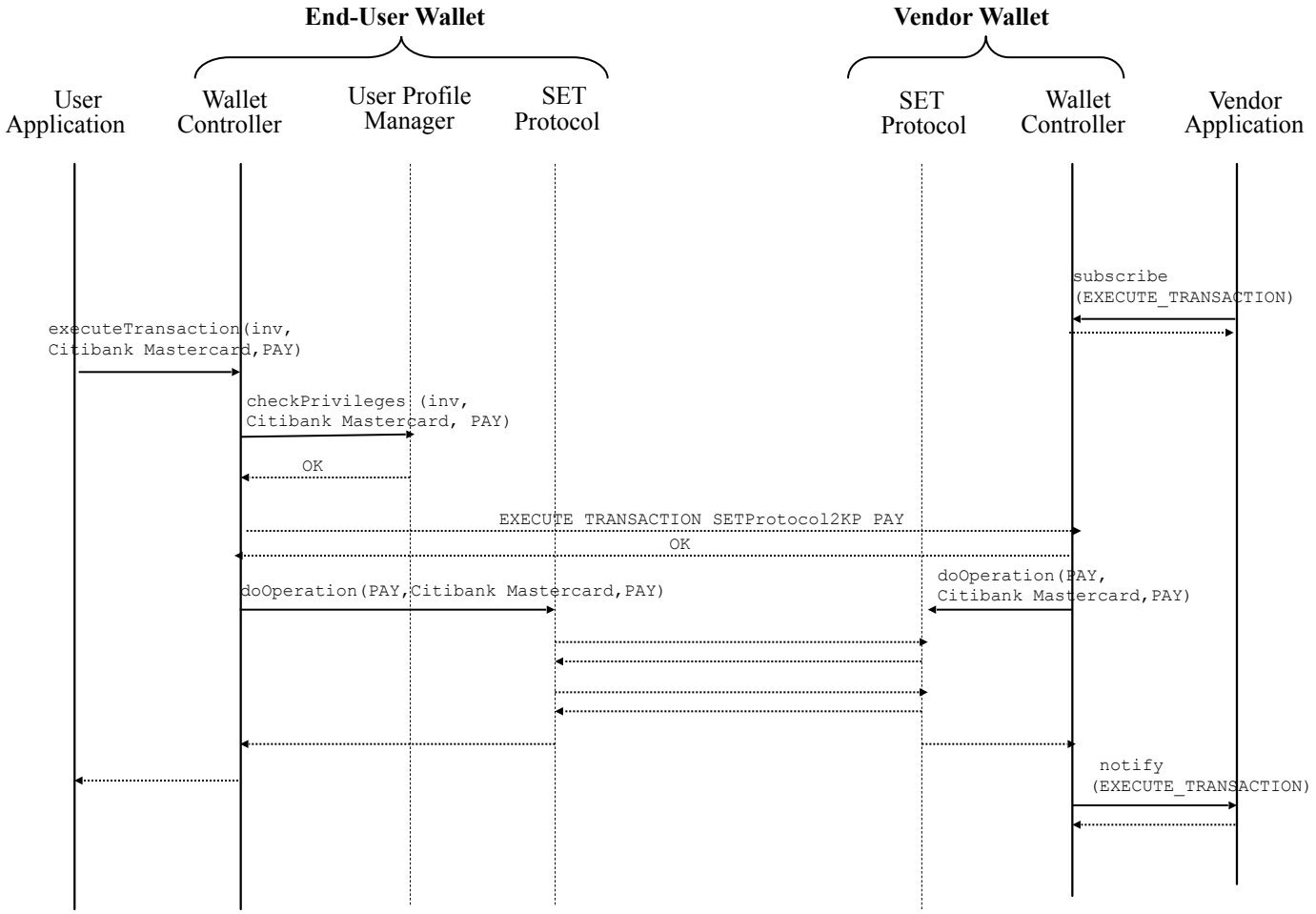


- Open Session
- Instrument Class Negotiation
- Protocol Negotiation
- Protocol Selection
- Instrument Selection
- **Transaction Execution**
- Close Session

Transaction Execution



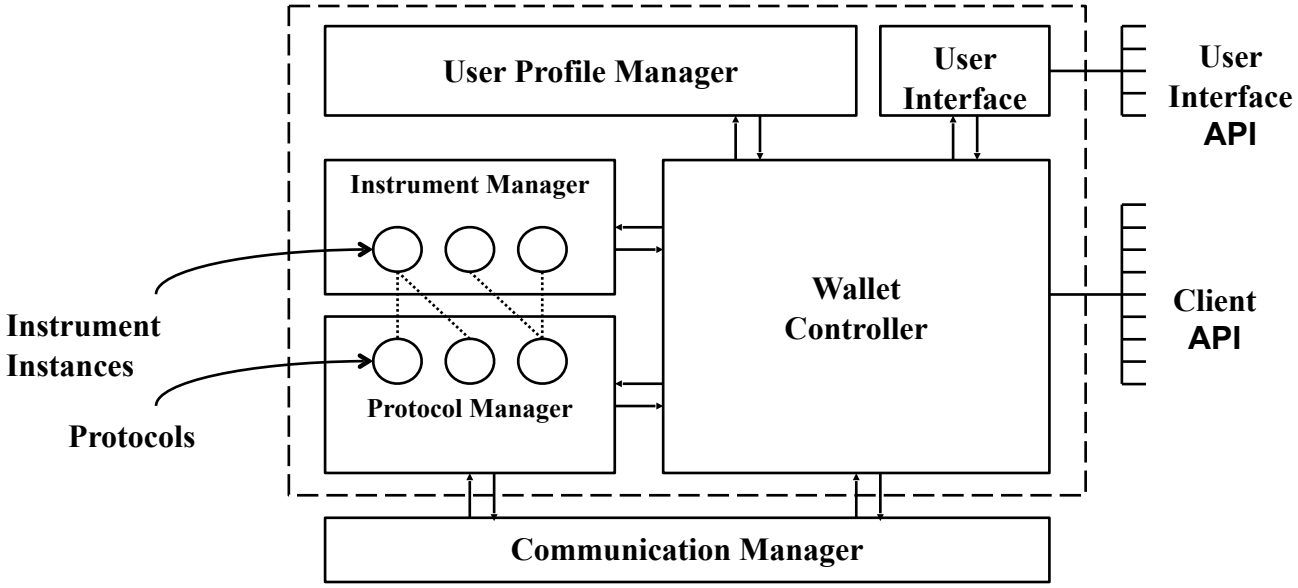
Transaction Execution



Transaction Execution

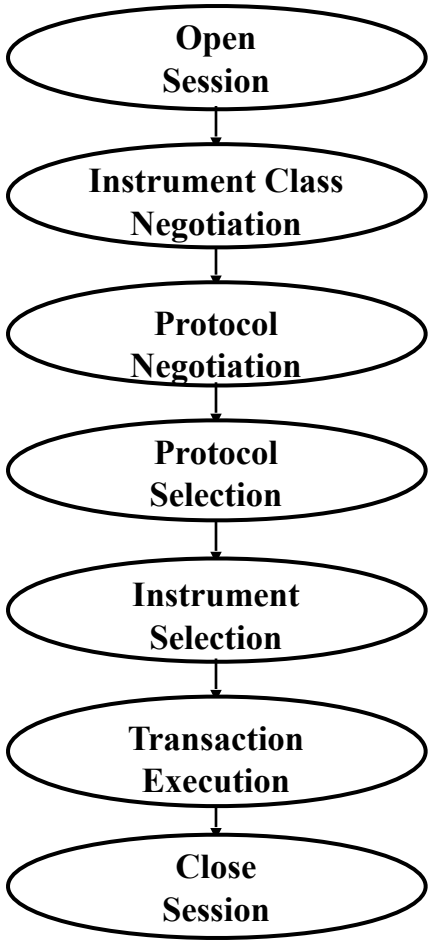
Trade-offs / Issues

- Where to put what functionality?



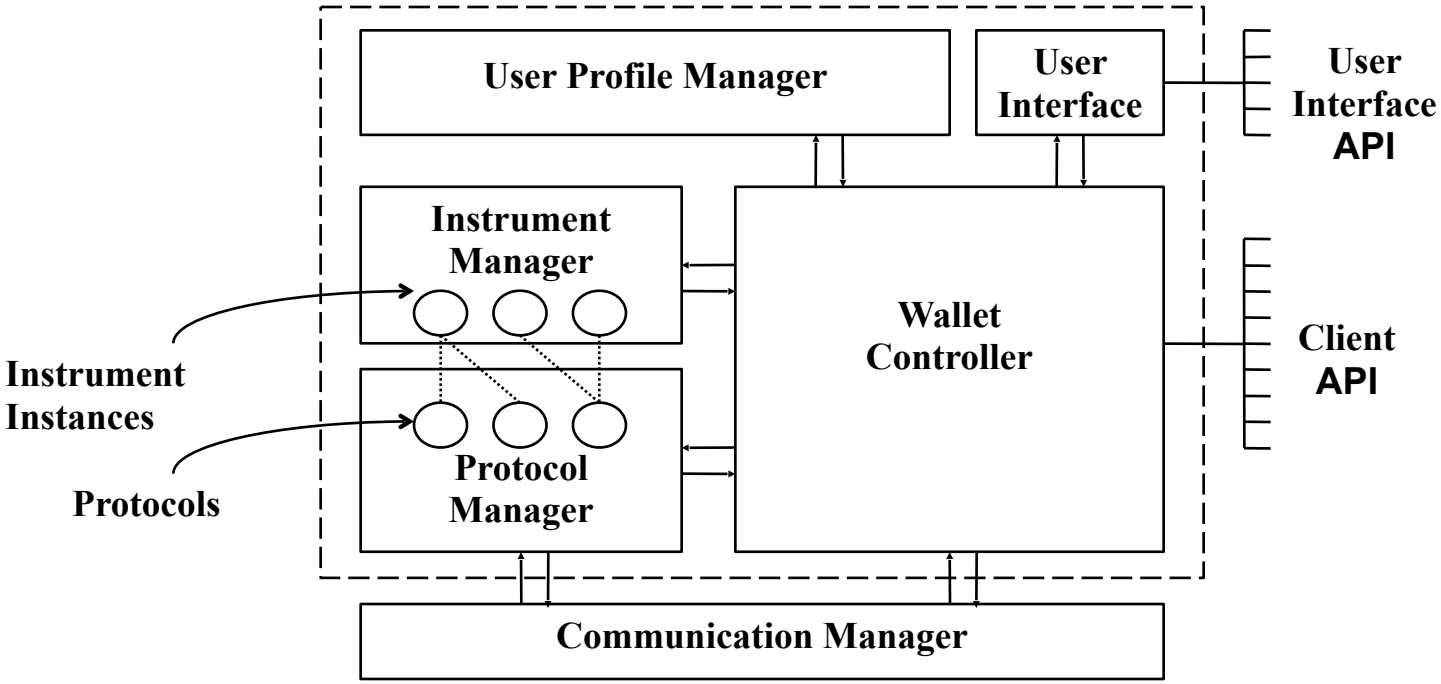
Trade-offs / Issues

- User Interaction
  - number of steps vs. likelihood of an error



Trade-offs / Issues

- Security vs. Customization
  - i.e., User Interface & UI API



Implementation & Future Work

- Implementation
  - C++ (PalmOS)
  - Java (Windows)
  - PonyCash
- Future Work
  - Populate the wallet
  - Experiment with other devices/environments (i.e. smart cards, mobile phones, web, etc.)
  - Abstract Data Manager

Summary / Contributions

- Desirable wallet properties: extensibile, symmetric, non-web-centric, client-driven
- Defined wallet interaction model
- Clean APIs for wallet and its components
- Prototype Implementation in Java & C++
  (available at http://www-db.stanford.edu/~daswani/wallets/)