



MODEL-BASED, EVENT-DRIVEN SCALABLE PROGRAMMING FOR THE MOBILE CLOUD

Gul Agha (PI), Darko Marinov (co-PI), Karl Palmskog (postdoc)
University of Illinois at Urbana-Champaign

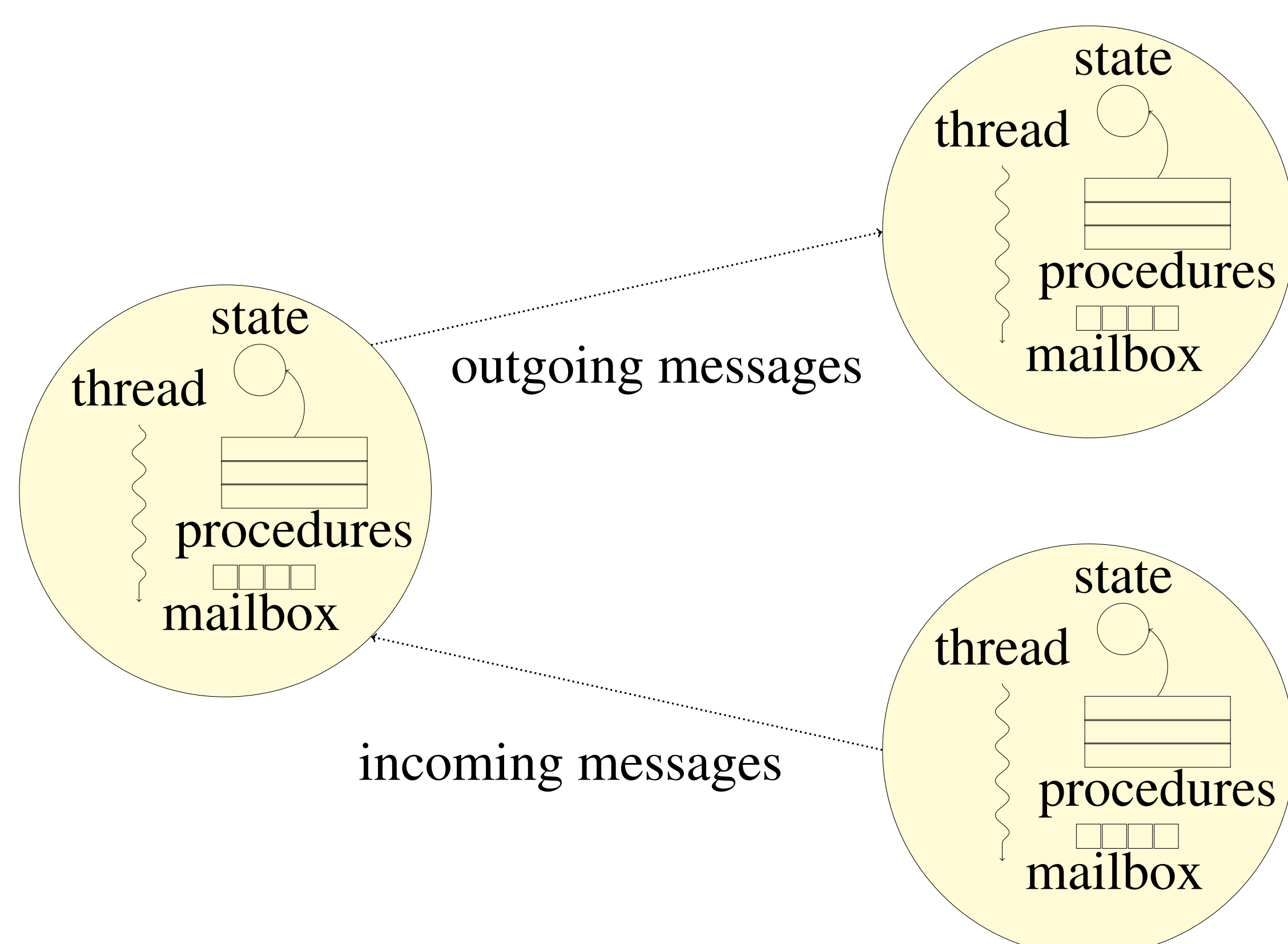
Daniel Jackson (co-PI), Ivan Kuraj (PhD student)
Massachusetts Institute of Technology



Objectives

- Support mobile applications integrated with cloud computing.
- Current mobile cloud application development frameworks complicate user-level code.
- Our Goal: develop a framework to build mobile cloud applications. The framework will:
 - simplify user-level code specification (*Sunny Programming Framework*); and
 - facilitate mobility and scalability (*Actor implementation*).

Actors for Scalability



- Decentralized control, state encapsulation, location transparency, and mobility make actors suitable for implementing scalable systems.
 - Example Actor Systems: LinkedIn, Twitter, Facebook Chat
- “...the actor model has worked really well for us, and we wouldn’t have been able to pull that off in C++ or Java.”
- Facebook Engineering

Cloud-based Web Programming Simplified

Developing web applications such as *chat* using *Sunny* requires only defining a data model (*records*) and client-server interactions (*events*):

```
record Room {
  name: String,
  members: set User,
  msgs: set Msg
}
```

```
record Msg {
  text: String,
  time: Timestamp,
  sender: User
}
```

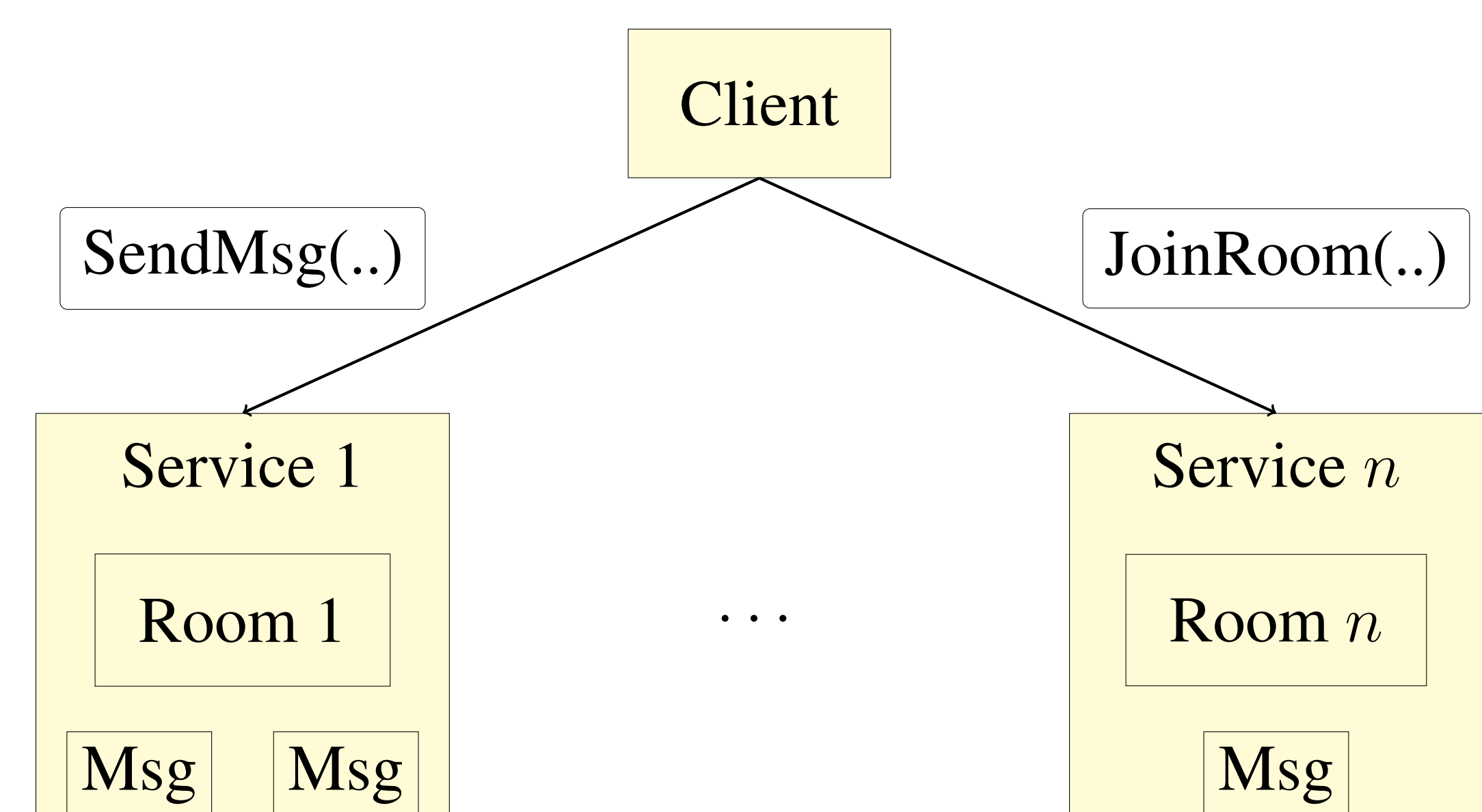
```
event JoinRoom(r: Room, u: User)
on (not u in r.members) {
  r.members += u
}
```

```
event SendMsg(r: Room, m: Msg)
on (m.sender in r.members) {
  r.msgs += m
}
```

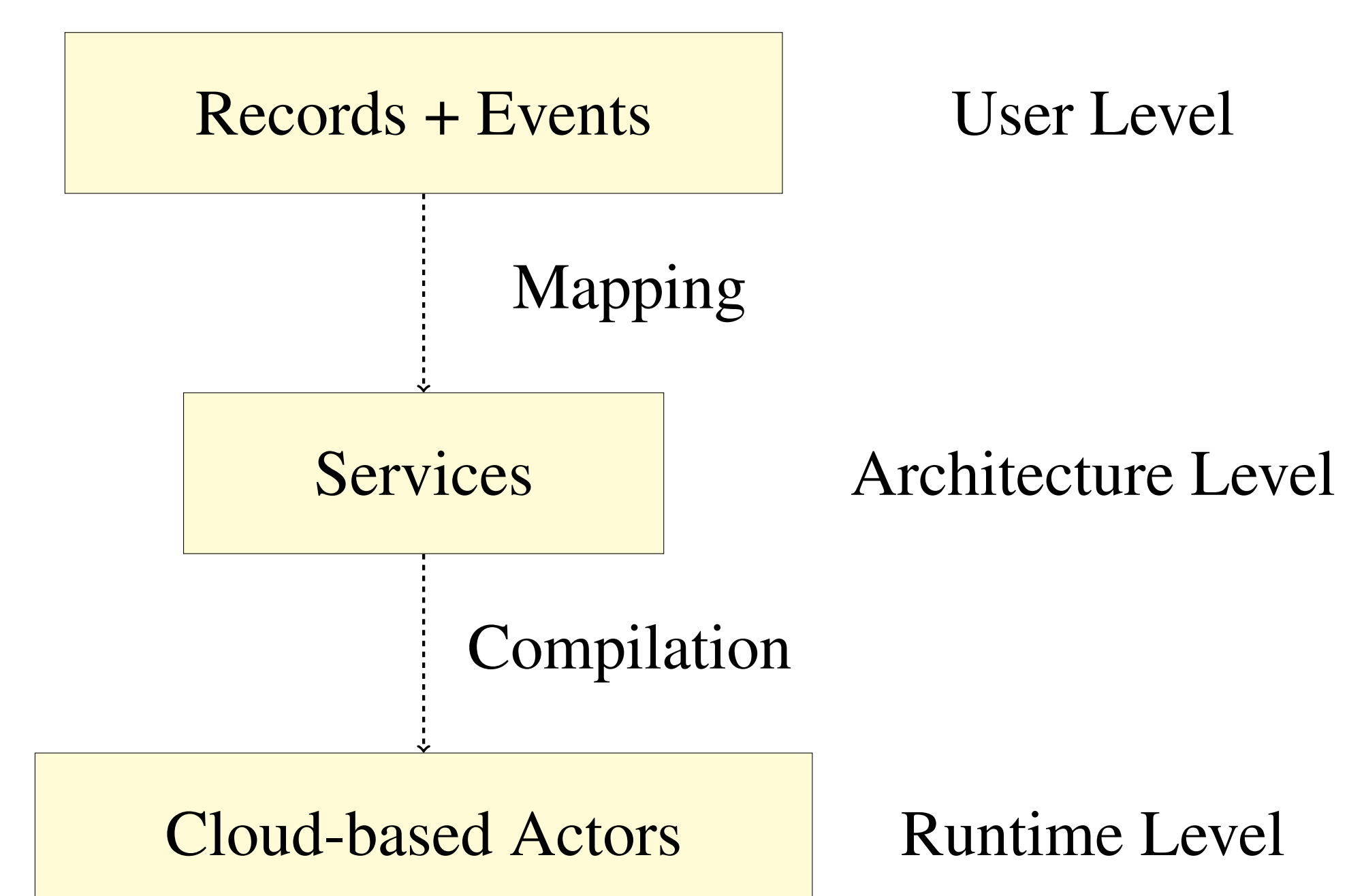
Events can be augmented by *security policies* to prevent unauthorized data access, represented at runtime with low overhead.

Approach

The application data model is decomposed, and its set of events split in a controllable way among services that are units of concurrency. Example chat application:



Development and deployment concerns are separated into levels:



Application Scalability

- Data model decomposition allows for scalable data storage.
- Events represented as client/server message exchanges at runtime.
- Concurrency and communication abstracted from *app* programmer.
- Distributing event processing among services represented as mobile actors allows scaling event throughput horizontally by adding more cloud servers.
- Mapping to services and compilation to actors enables trading availability for consistency.
- Strategies for actor placement on cloud servers to minimize communication can be inferred by observing communication patterns.

Current and Future Work

- Formalization of mapping to actors.
- Framework implementation.
- Evaluation of scalability for representative web applications.
- Tool support for modelling, testing and verification.