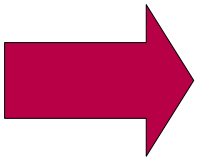
A distributed architecture to support infomobility services

Claudia Canali **Riccardo Lancellotti**

University of Modena
and Reggio Emilia

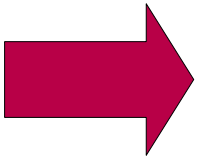
Motivation

- **Web 1.0**
  - Static Web pages
  - Information repository
  - Limited interactivity



- **Web 2.0**
  - Dynamic Web-based services
  - Personalized services
  - Collaborative services
  - High interactivity

- **Infomobility 1.0**
  - Static maps
  - Basic navigation support
  - No interactivity
  - Car-oriented services



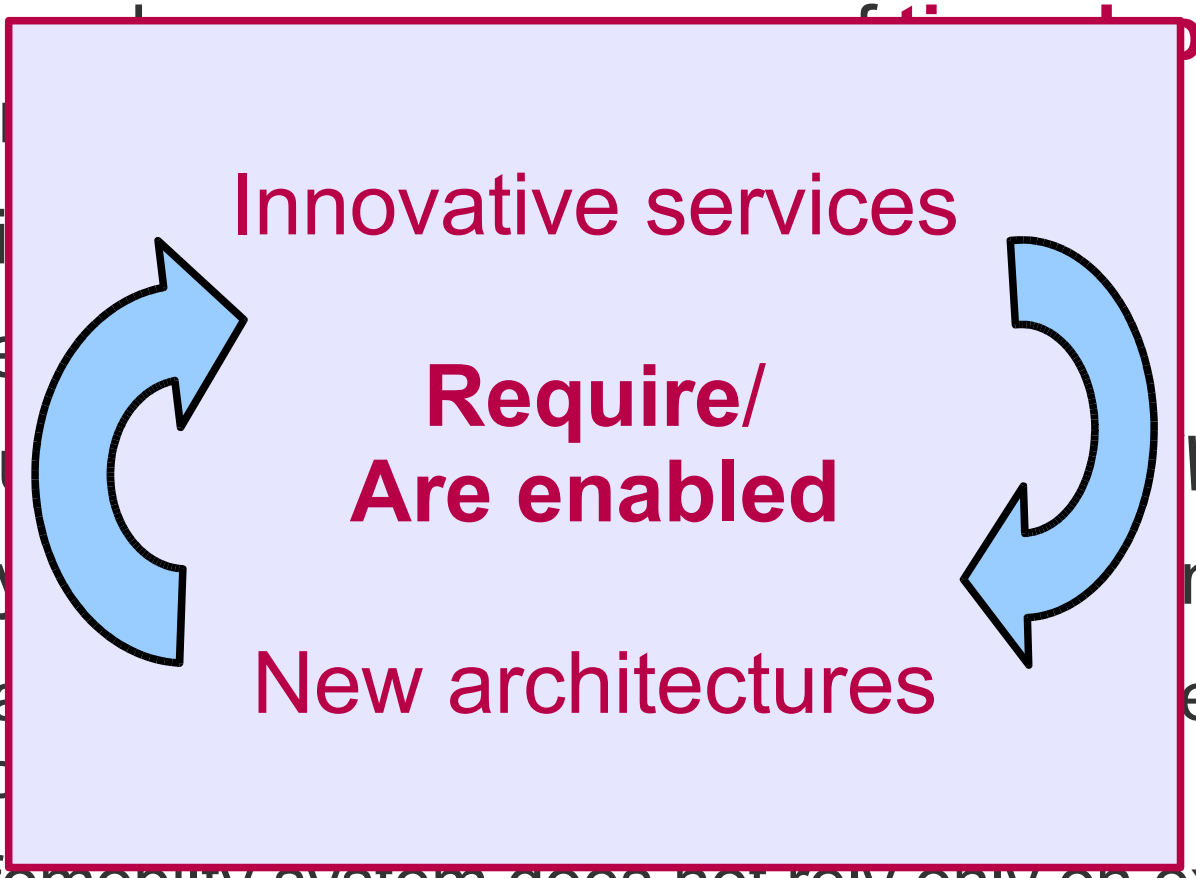
- **Infomobility 2.0**
  - Collaborative infomobility services
  - Personalized services
  - High interactivity (wireless connections)
  - Services oriented to every means of transportation

Examples of Infomobility 2.0 services

- Always up-to-date maps (on-demand map download)
- Dynamic exchange among users of **time dependent** Geo-referenced data
  - Real-time POI sharing
  - Geo-referenced bulletin boards
  - Maps updated by the users for the users *(Wikimaps)*
- On-the-fly user feedback analysis to extract information
  - Automatic detection of delays, traffic jams depending on user position/speed information
  - The infomobilty system does not rely only on external data sources

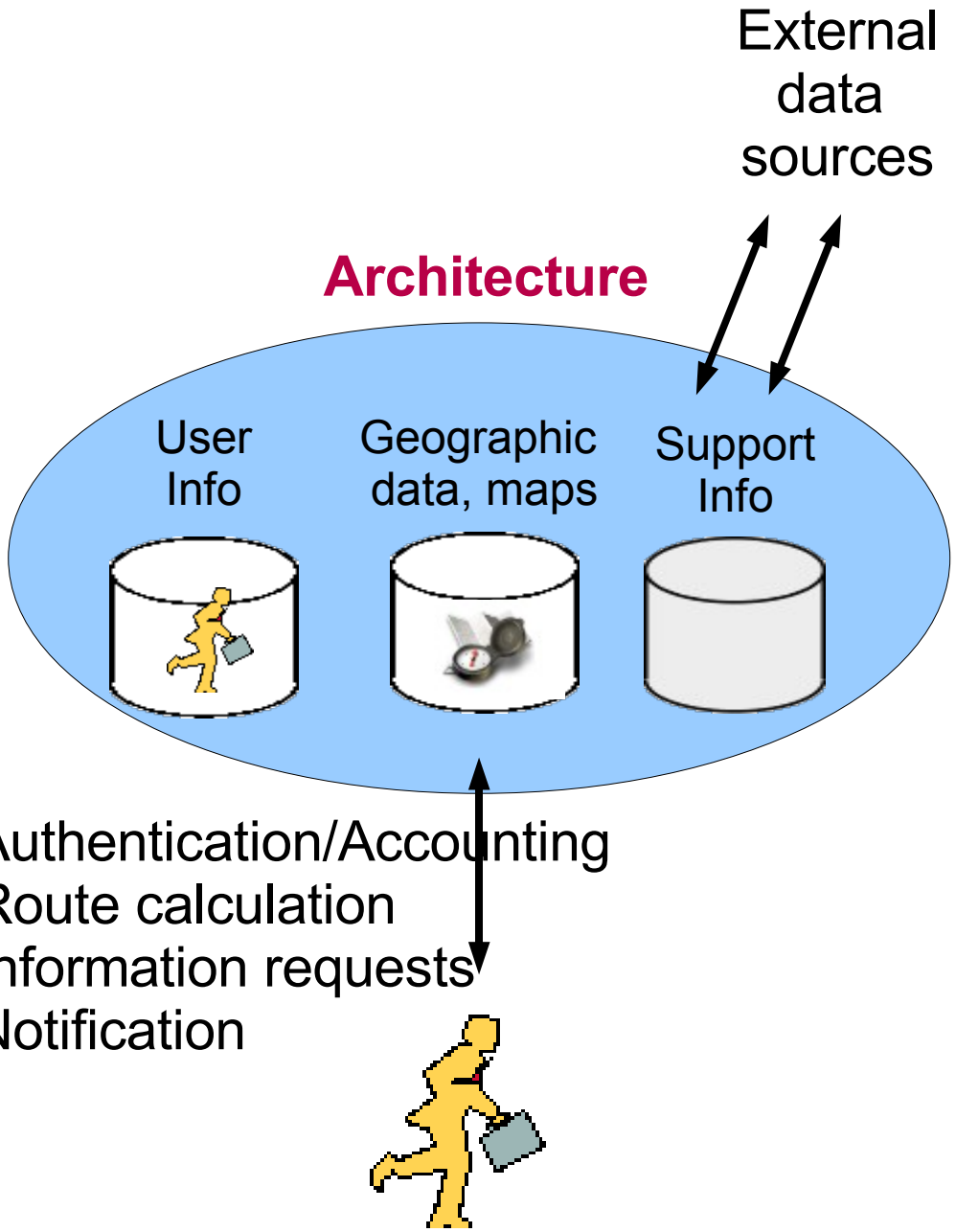
Examples of Infomobility 2.0 services

- Always up-to-date maps (on-demand map download)
- Dynamic
  - Real-ti
  - Geo-re
  - Maps
- On-the-fly
  - Automa
    user pc
  - The infomobility system does not rely only on external data sources



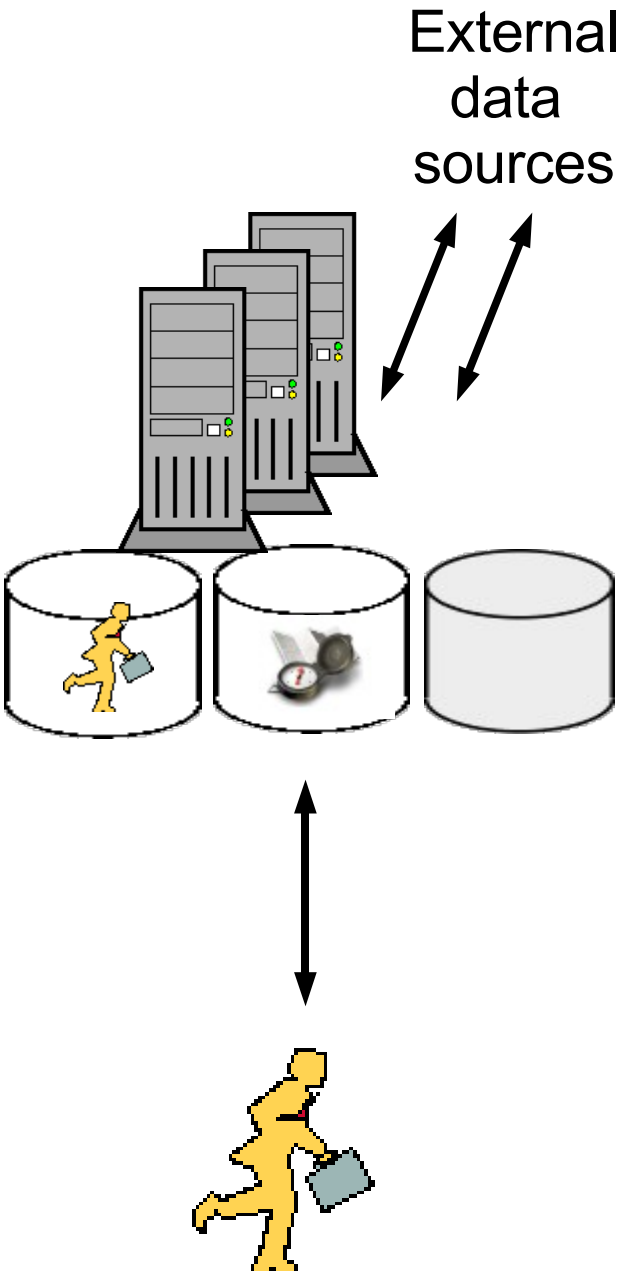
Architecture requirements

- Interactions with the user
- Management of information
- Interaction with external data sources
- Key requirements:
  - User data privacy
  - Data consistency
  - Service performance
  - Service availability



Centralized architecture

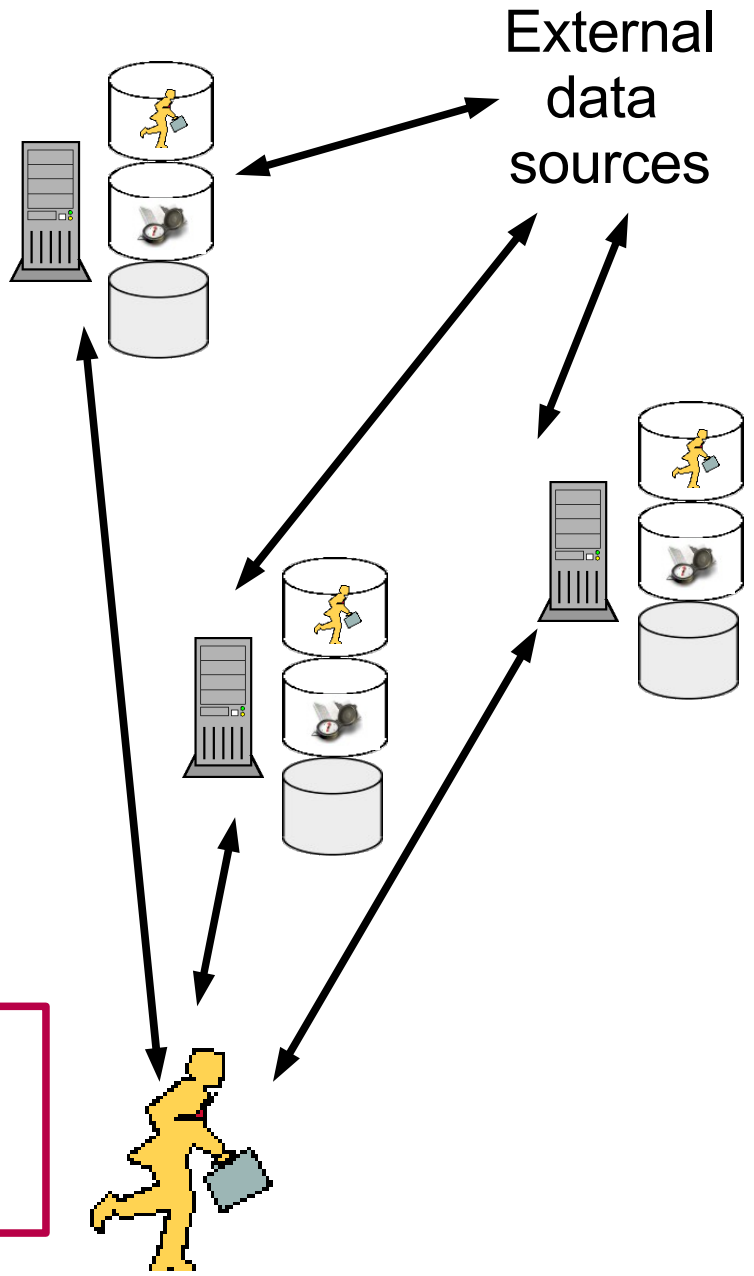
- User data privacy → OK
- Data consistency → OK
- Service performance
  → Possible bottlenecks
  - preliminary experiments with Web services: CPU, network, sockets
- Service availability
  → Single points of failure
  - central node, first mile, DoS attacks



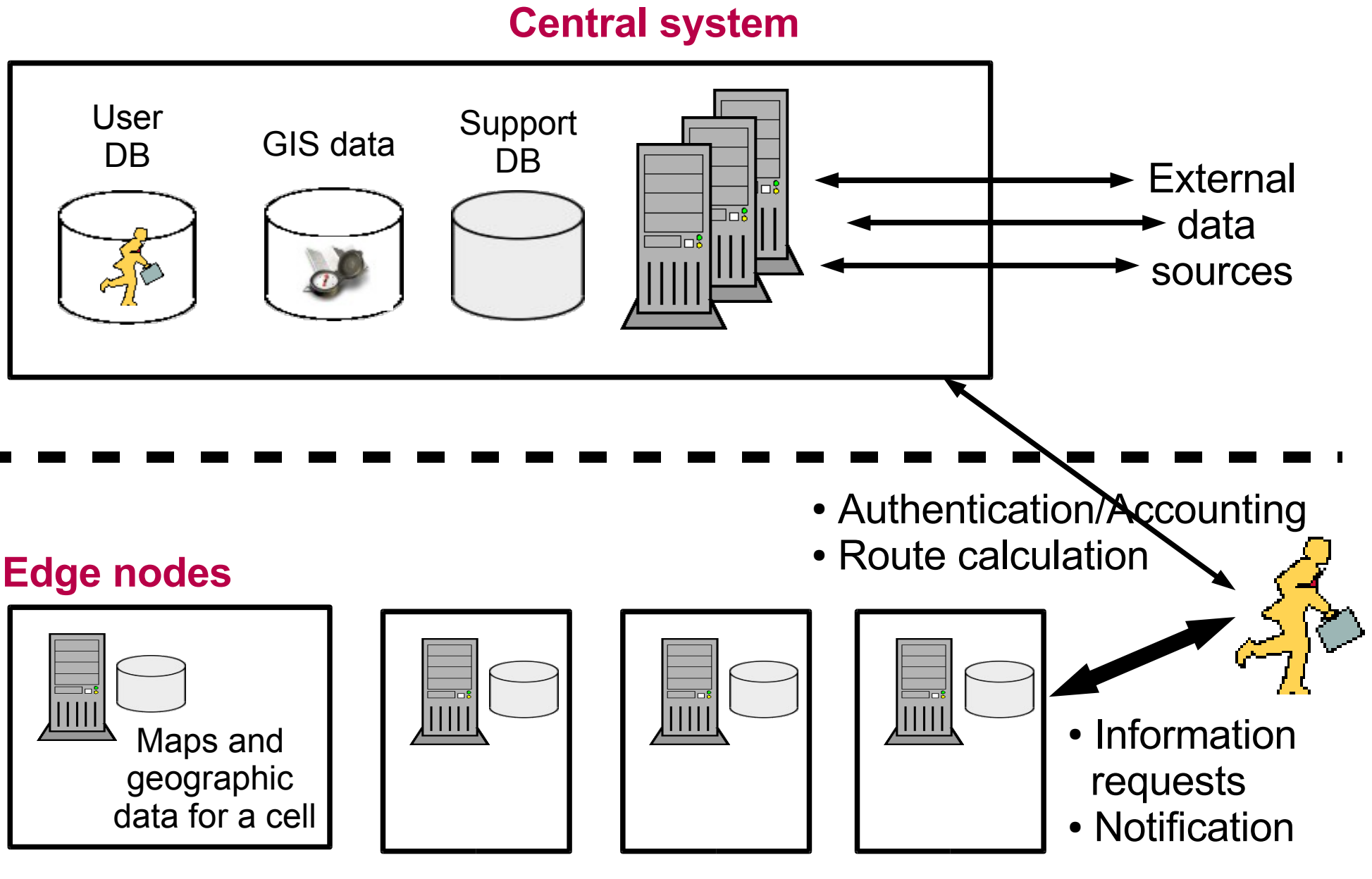
Fully distributed architecture

- User data privacy
  → Expensive to guarantee high security level for every node
- Data consistency
  → Critical when # of nodes is high
- Service performance → OK
- Service availability → OK
- – Function replication

Possible solution
→ hybrid architecture



Hybrid solution: Two-level architecture



Two-level architecture

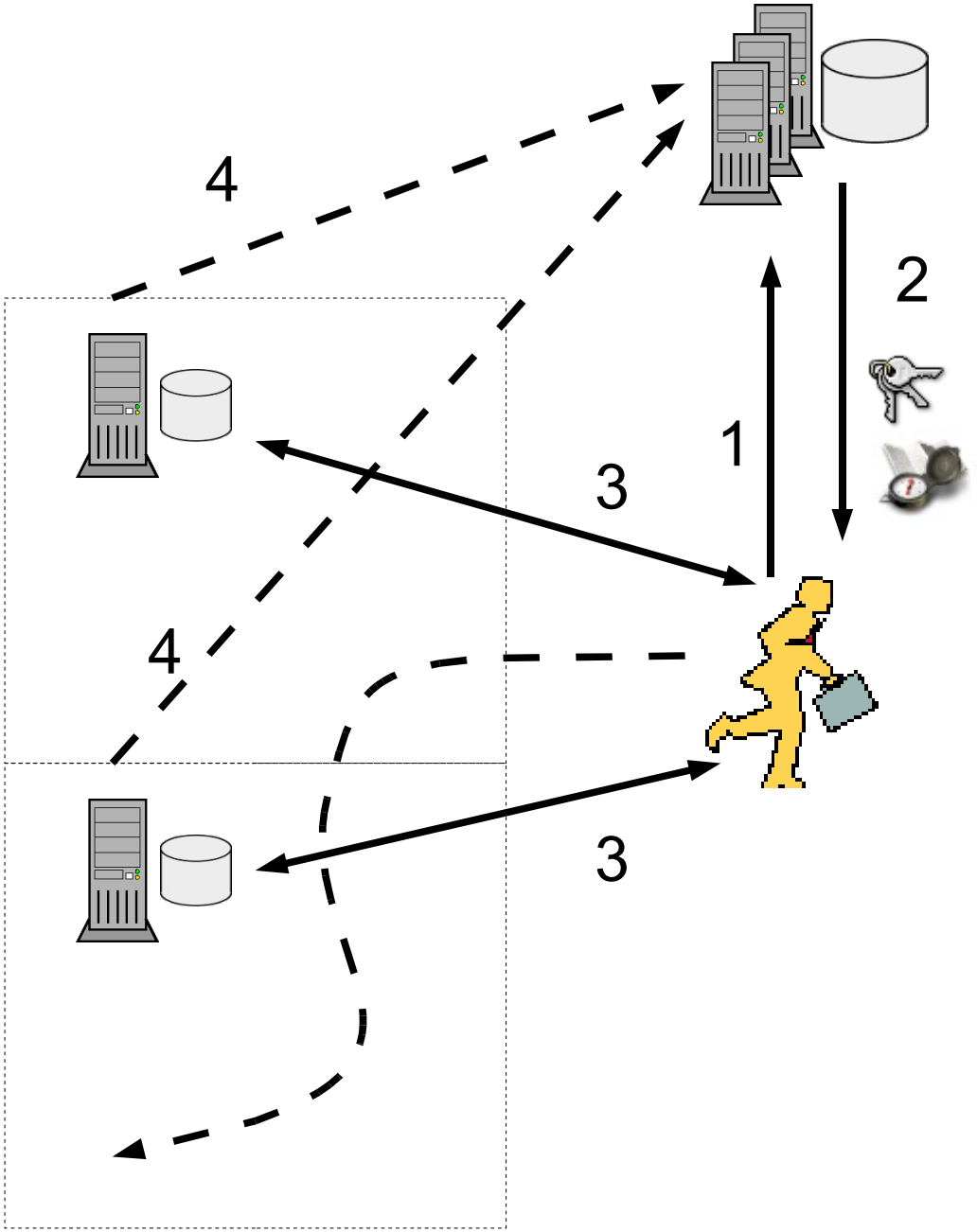
- User data privacy
  - Critical information only on central system
  - Use of temporarily IDs for interaction with edge nodes
- Data consistency
  - Data partition on the edge nodes and on central system
- Service performance
  - Central system → clustering
  - Edge nodes → replicated
- Service availability
  - Most interaction is with edge nodes

Prototype implementation

- System based on Web services
  - Apache httpd + Tomcat
  - Axis 2 as the Web service implementation
  - GRASS GIS
  - Mysql
- Central system: cluster of 5 nodes
  - 1 Apache httpd dispatcher
  - 4 Tomcat + Axis2 + GRASS
- Edge nodes: 10 nodes
  - Tomcat + Axis2 + Mysql
- Support for WAN emulation

Prototype support for user navigation

1. User requests to central system
   - log in
   - route request
2. Central system returns
   - route description
   - auth tokens for cells
3. Interaction with edge nodes
   - Information requests
   - Notification (polling)
4. Feedback to the central system (e.g., delays, accidents, detours)



Management of user information

- User authentication only on the central system
- Central system issues a set of temporarily tokens:
  - Route ID
  - Expiry date
  - Cell for which the token is valid
  - User info: reputation,...
  - Signature of the central system

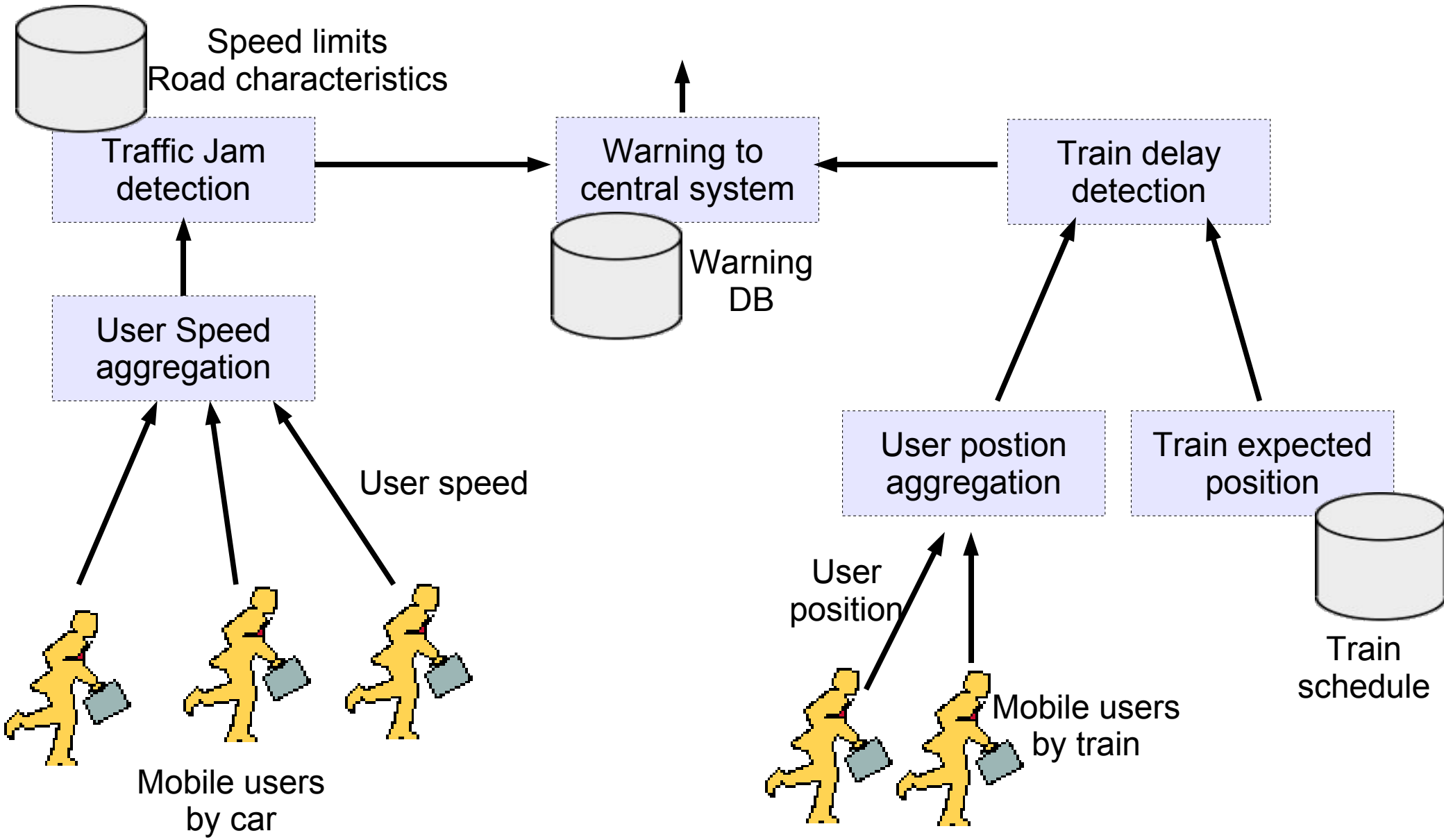
Only the central system can determine the user identity from the token ID

- Edge nodes accept tokens as authorization credentials
- Cryptography in communication with edge nodes may prevent replay attacks (HTTPS)

Interaction of users with edge nodes

- Requests
  - Maps
  - POIs
- Notifications
  - New POIs
  - Information with global relevance (e.g., public transportation delays, traffic jams, accidents)
    - Edge nodes aggregate information with quorum/reputation-based filters
  - User position and speed
    - Automatic information extraction

Automatic extraction of information: edge node prototype implementation



Conclusions

- Infomobility 2.0 → , collaboration, personalization interactivity
- Centralized and fully distributed architectures are not suitable for infomobility services
- Proposal: two-level architecture to support infomobility services
  - Compromise between fully distributed and centralized architectures
- Prototype based on Web services

A distributed architecture to support infomobility services

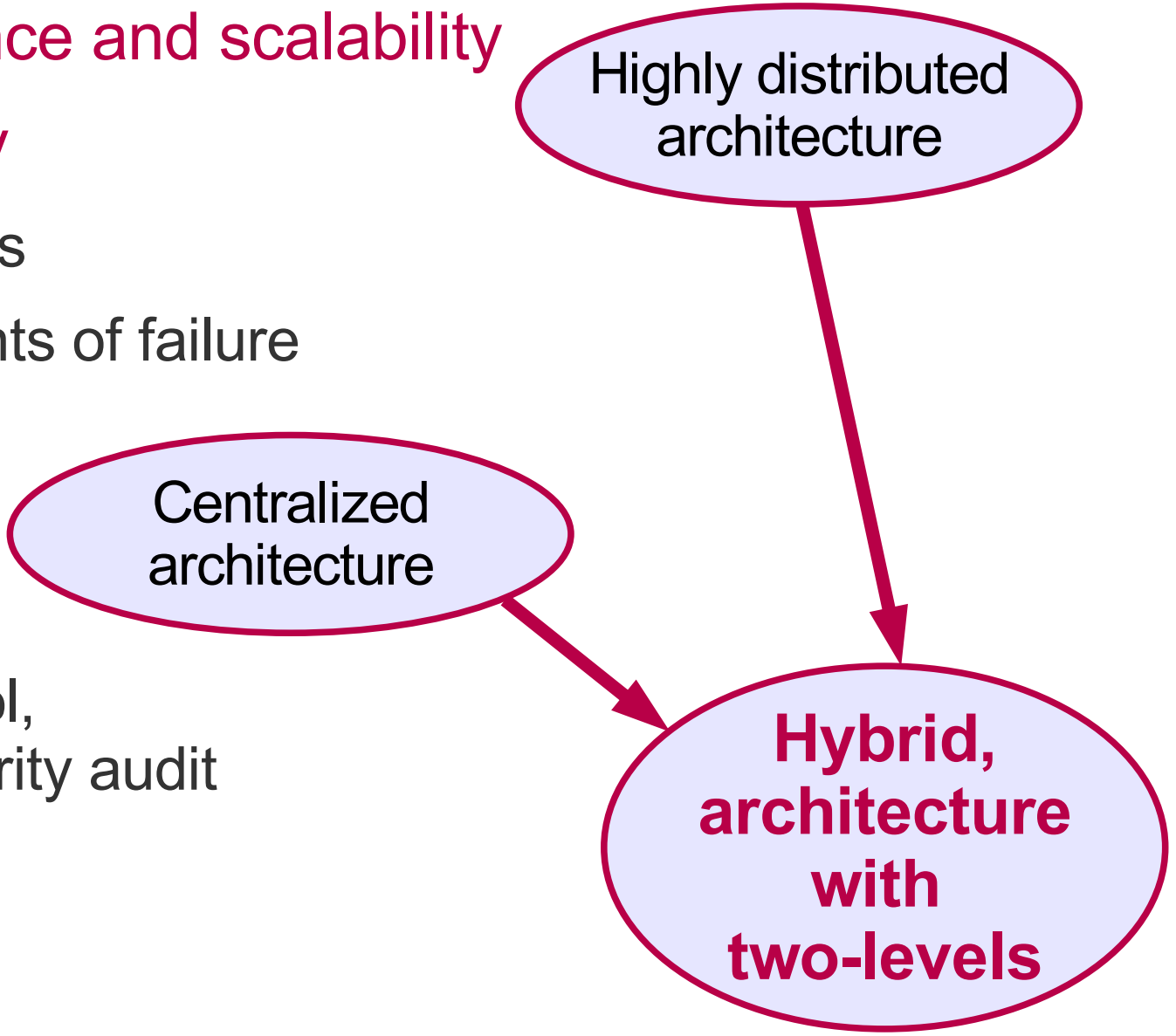
Claudia Canali **Riccardo Lancellotti**

University of Modena
and Reggio Emilia

Frattaglie

Requirements for the architecture

- High performance and scalability
- High availability
  - No bottlenecks
  - No single points of failure
- User privacy
  - High security
  - Access control, frequent security audit



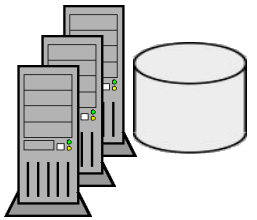
Requirements for the architecture

- High performance and scalability
- High availability
  - No bottlenecks
  - No single points of failure
- User privacy
  - High security
  - Access control, frequent security audit

Architectural alternatives

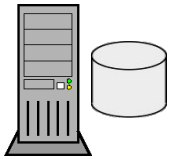
- Centralized architecture
  - Privacy → OK
  - Performance and scalability → Possible bottlenecks
  - Availability → Single Point of failure
- Completely distributed architecture
  - Performance and scalability → OK
  - Availability → OK
  - Privacy → High security in every node is expensive
- **We introduce a new Two-level architecture**

Details on the central system



- Highly controlled environment
- Computationally powerful (Cluster)
- Functions of the central system
  - Calculation of the user route
  - Authentication of the users
  - Accounting (pay-per-user services)
  - Generation of auth tokens for the interaction with edge servers
  - Access to external data sources (e.g., traffic status, transportation booking services)
- Data stored on the central system
  - Geographic data for the computation of user routes (GIS)
  - User preferences
  - Additional databases (e.g. public transportation schedules)

Details on the edge nodes



- Highly distributed
- Functions of the edge nodes
  - Servicing user request for geographic data (e.g., nearby POIs, maps)
  - Updating geographic data based on user-supplied informations (e.g., new POIs, detours, traffic jams, ...)
  - Extraction of information from user behavior
  - Aggregation and notification to central system of information with global relevance
- Data stored on the edge nodes (only related to the cell)
  - Maps, POIs, speed limits and other Geographic data about the cell (and possibly about nearby cells)
  - Additional databases (e.g. public transportation schedules)

The Client device

- Portable device (e.g, handheld device, not limited to car-based travels)
- Wireless connectivity (GPRS, UMTS, WiMax, ...)
- GPS support
- No need for large storage (maps are downloaded as needed)
- Support for interaction based on Web services
- User interface may exploit other Web-based technologies (e.g., Ajax)

Requirements for the architecture

- High performance, scalability
  - High number of edge nodes
  - Central system only for few, critical operations
- High availability
  - High number of edge nodes
  - Data replication allows an edge node to “take over” nearby cells in case of failure
- Privacy
  - Central system is secure
  - User-related information are stored only on the central system
  - Use of temporarily ID for user interaction with edge nodes