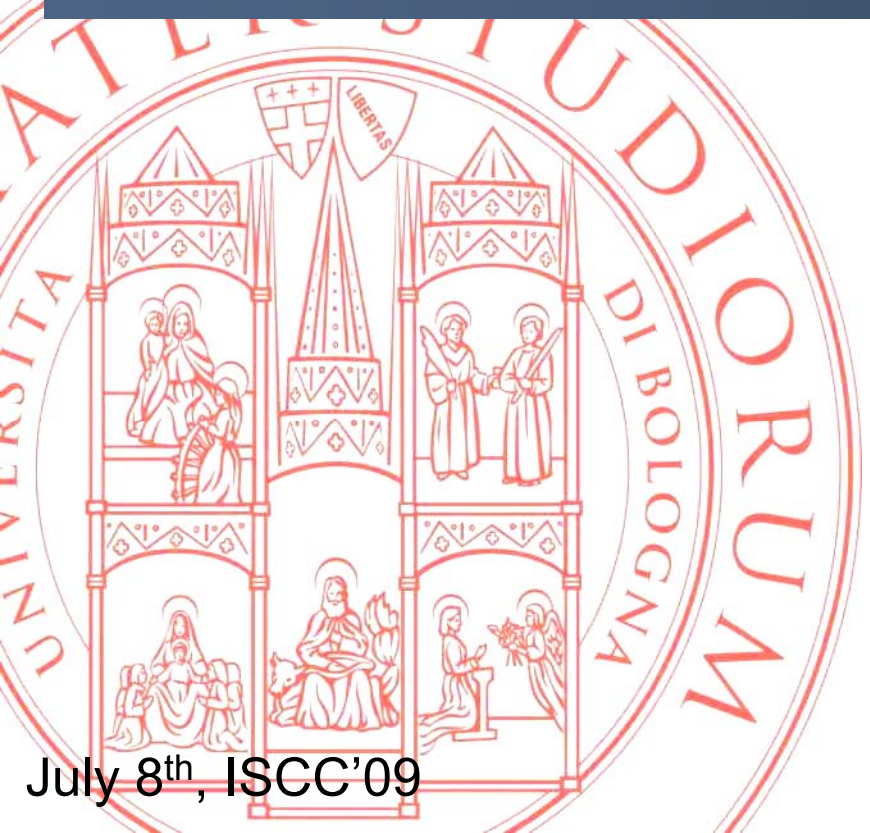
Implementing a Scalable Context-Aware Middleware



Antonio Corradi,
Mario Fanelli, Luca Foschini

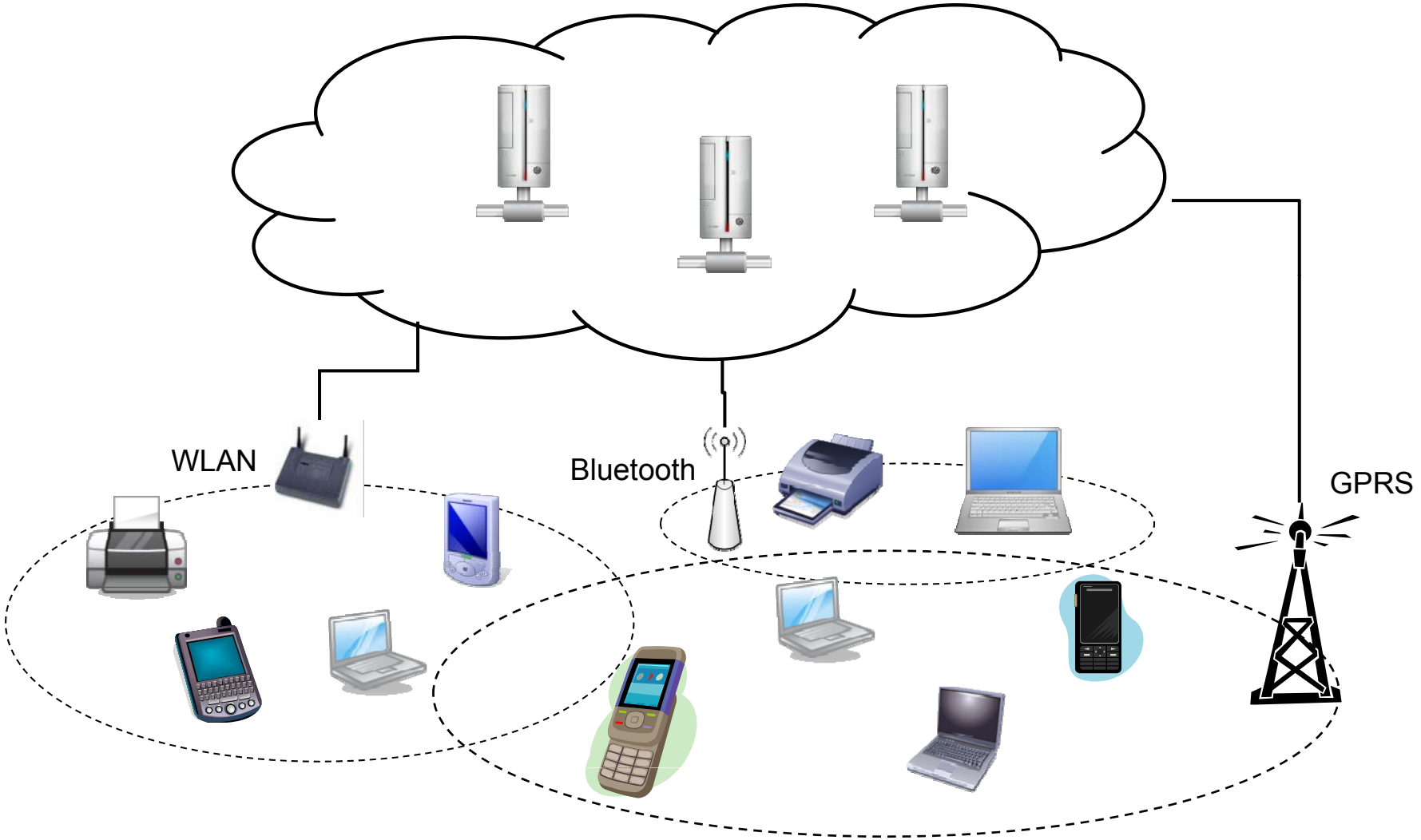
DEIS, University of Bologna, ITALY
{antonio.corradi, mario.fanelli,
luca.foschini}@unibo.it

July 8th, ISCC'09

Agenda

1. **Context-Aware** applications for **mobile environments**
2. Context data dissemination in mobile, heterogeneous, and densely populated environments
   - **S**calable context-**A**ware middleware for mobi**L**e **E**nviroment**S (SALES)**
   - Context data dissemination
3. Implementation insights
4. Experimental evaluations
5. Conclusions and ongoing work

Application scenario: Context-Aware applications for mobile environments



Context data dissemination issues

To enable the realization of context-aware applications, we have to distribute context data to devices taking care of:

- **Mobility**
  - Mobility introduces a heavy management overhead
  - Handoff and reconfiguration mine dissemination dependability
- **Heterogeneity**
  - Many devices (PDA, laptop, etc.) have different computational capabilities
  - Different wireless standards guarantee very different available bandwidths
- **Scalability**
  - Context data can have long payload and high production rates
  - The data dissemination can overwhelm the system

To suitably support the diffusion of context-aware applications, we need proper middlewares that transparently address the above issues

Scalable context data dissemination: Design guidelines

1. **Constraint data dissemination**
   - Physical-locality principle: location-dependent context data must be kept near associated sources
   - Logical-locality principle: context data must be disseminated only to interested node

2. **Use a cluster-based architecture**
   - Each cluster obeys physical locality principle
   - Clusters can be used to constraint the dissemination process

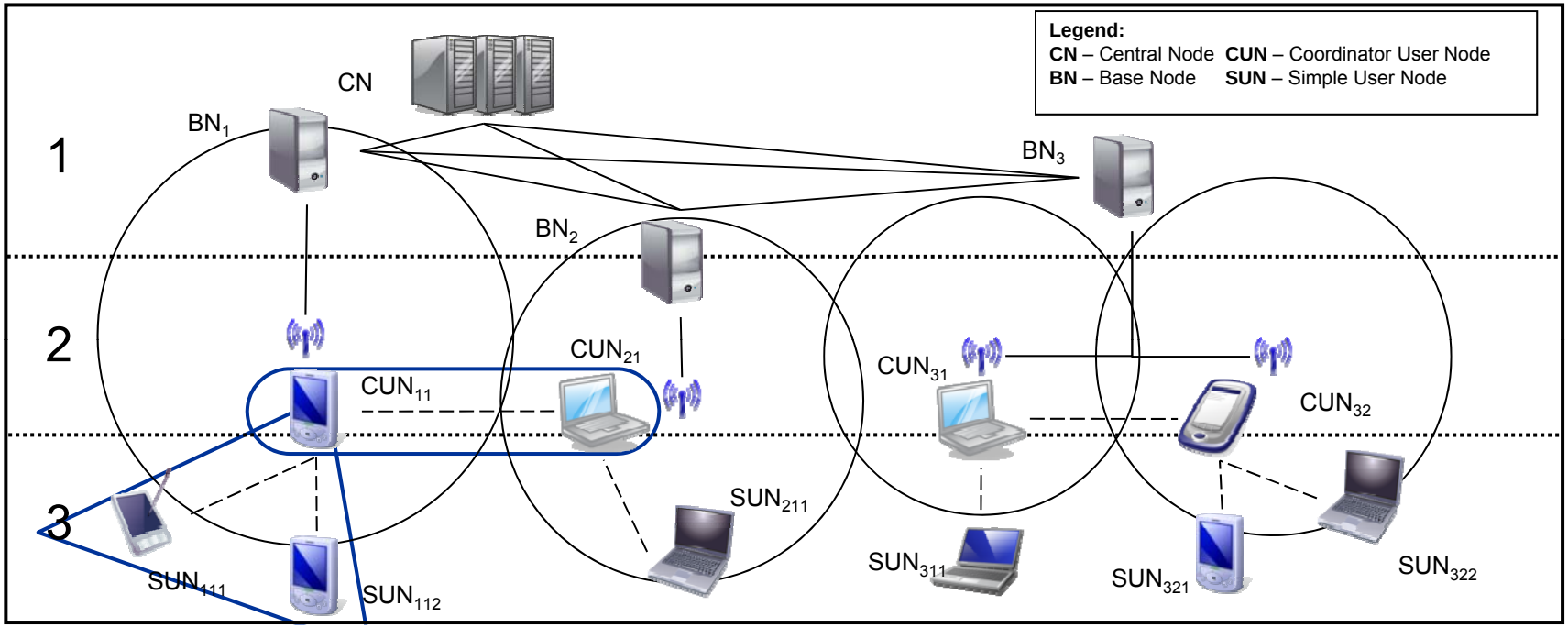
3. **Exploit adaptability to foster scalability**
   - Dissemination task depends on available bandwidth and data scope
   - Mobility management protocols are adapted according to nodes joint mobility and wireless transmission range

Scalable context data dissemination: Design guidelines

4. **Use heterogeneous wireless communications**
   - Supporting different wireless standards increases system coverage and total available bandwidth

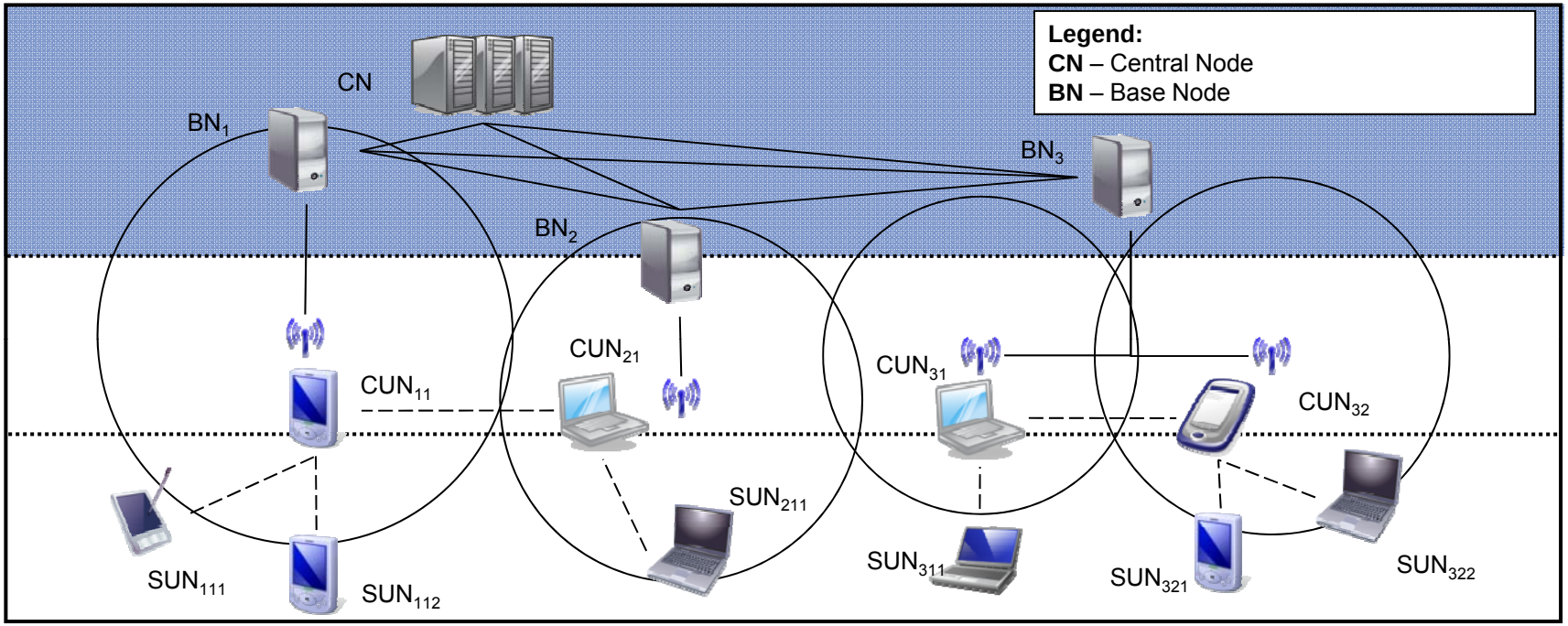
5. **Exploit different wireless modes**
   - Fixed infrastructures guarantee data availability
   - Ad-hoc links enable cheap data dissemination

Scalable context-Aware middleware for mobiLe EnviromentS



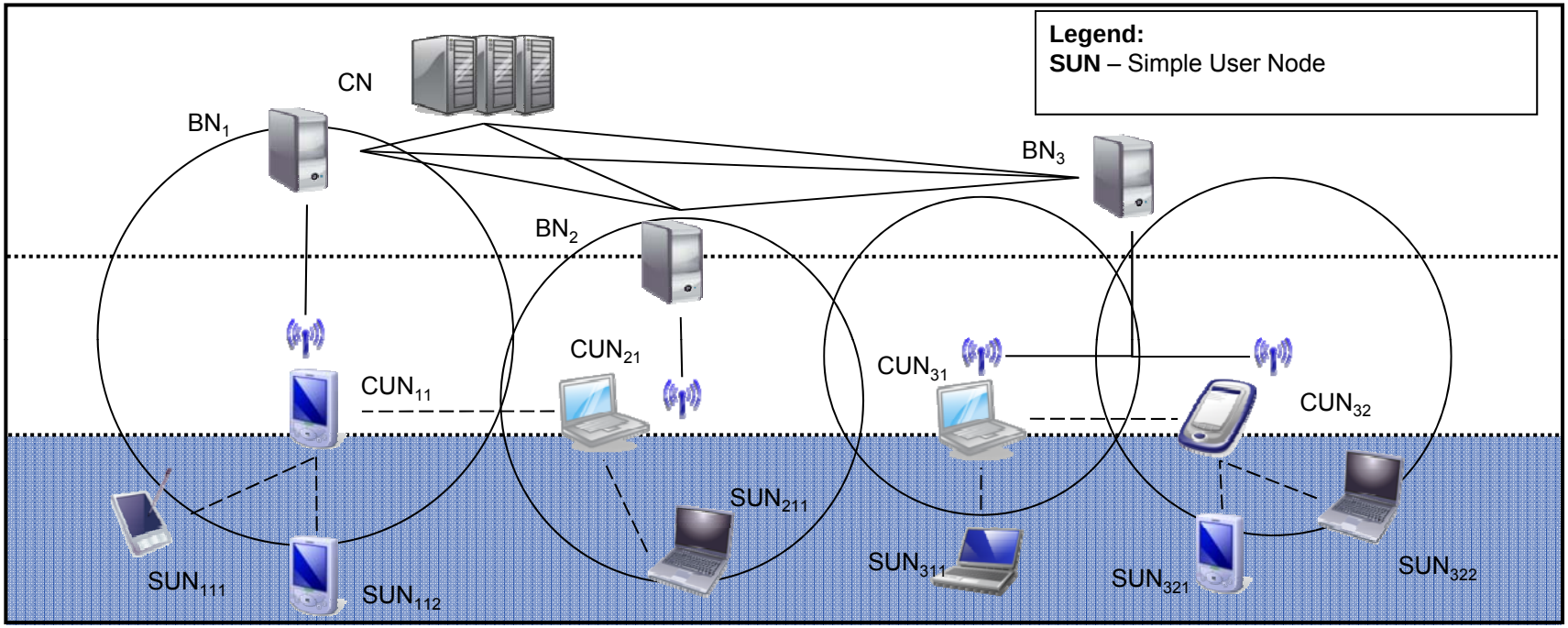
- **Tree-like three-level** architecture
- The distributed architecture reflects **physical locality principle** → Each father node groups near child nodes
- Nodes belonging to the same level form a **collaborative network** in which data can be disseminated in a peer-to-peer (P2P) manner

Scalable context-Aware middleware for mobiLe EnviromentS



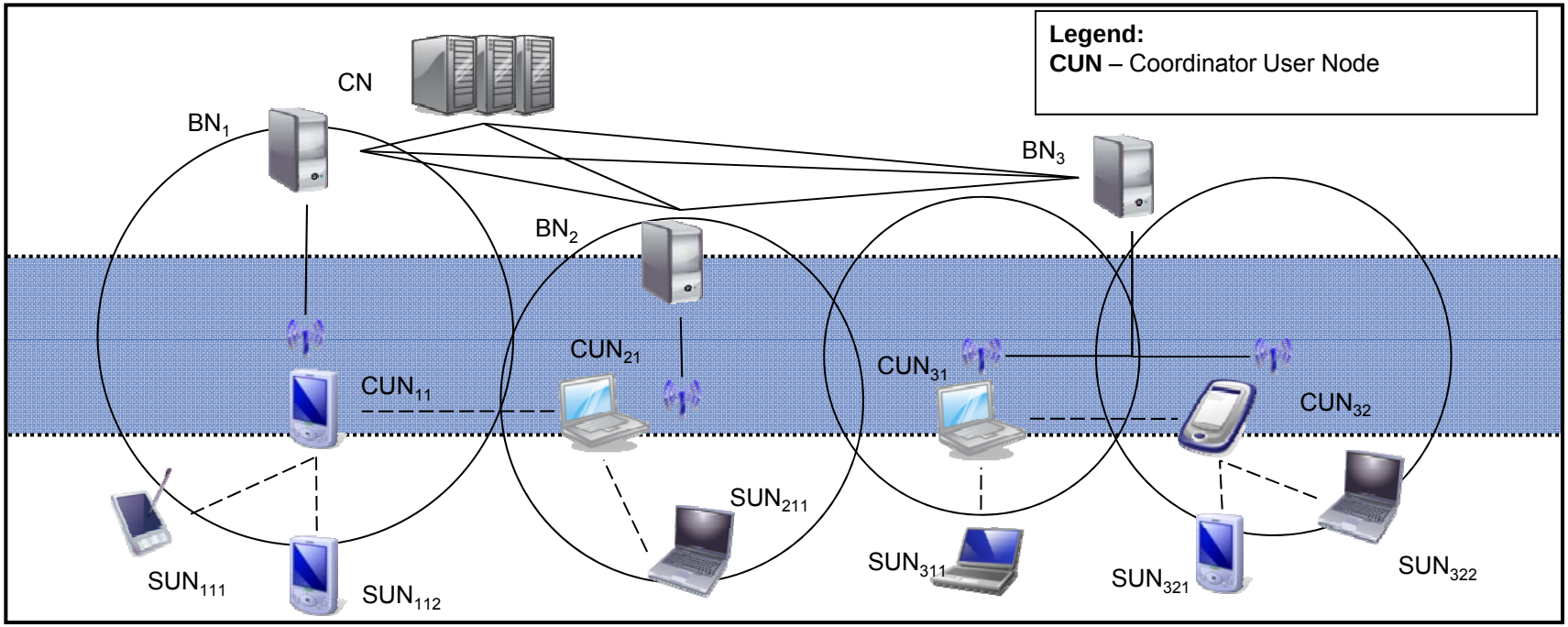
- **Fixed infrastructure**
  - Central Node ensures data history and access
  - Base Nodes are the SALES fixed infrastructure entry points
  - Base Nodes memorize context data to reduce the requests routed up to the Central Node

Scalable context-Aware middleware for mobiLe EnviromentS



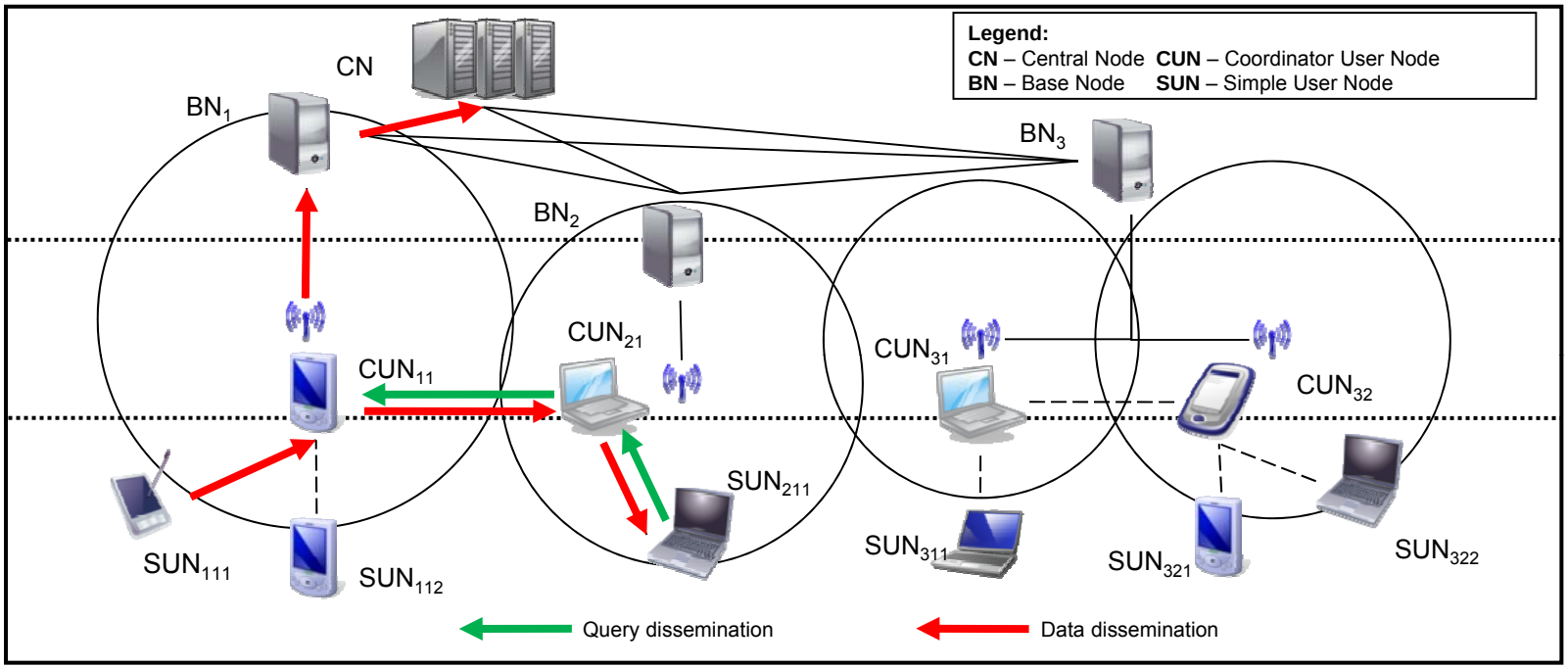
- **Mobile infrastructure**
  - Communications between user nodes exploit only ad-hoc links
  - Simple User Nodes share local context data repositories with peers
  - Coordinator User Nodes share local context data repositories with peers and served nodes

Scalable context-Aware middleware for mobiLe EnviromentS



- **CUNs bridge together the fixed and the mobile infrastructure**
  - CUNs should be multi-homed nodes
  - CUNs enact as routers even between different technology-specific networks
  - Mobility management protocols between a CUN and its served SUNs are completely based on ad-hoc links

Context data dissemination



- To build dissemination paths, SALES adopts **context queries.** A context query captures context needs by imposing constraints on data values
- The data dissemination takes place as follows:
  1. At default, data flow only on the bottom-up path between the data creator node and the Central Node
  2. Different dissemination paths are considered only if matching queries exist

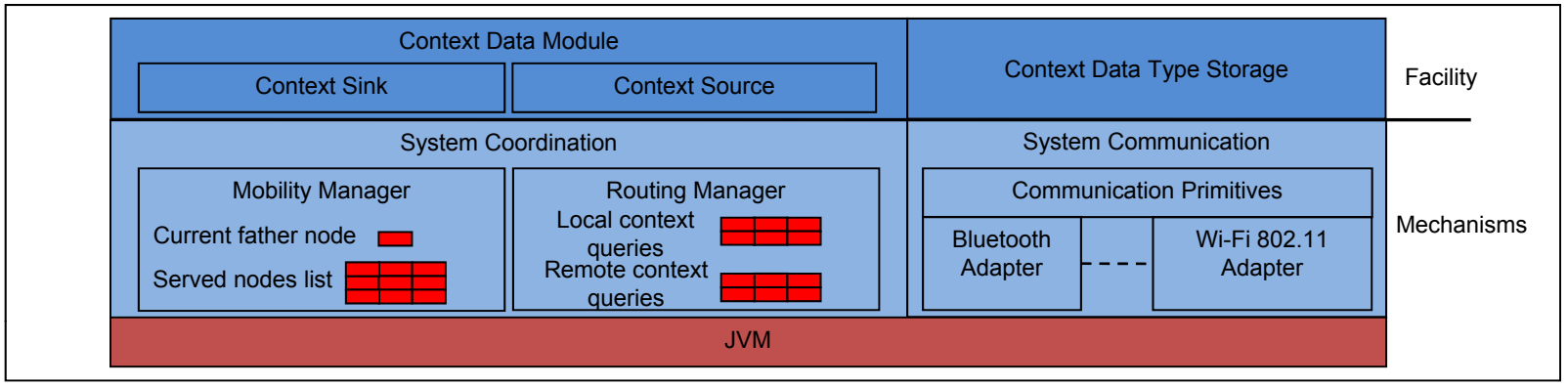
Context data dissemination details

- To build different dissemination paths, each query is disseminated inside the SALES distributed architecture:
  1. First, the query is disseminated on the same level **(horizontal propagation)**
  2. Then, the query is disseminated on the upper level **(vertical propagation)**
  3. Repeat from 1. until the query is valid and current node is not the CN

- SALES offers different parameters to tailor both data and query dissemination process. As regards query parameters:
  1. **Horizontal Time To Live (HTTL)**: The maximum number of nodes traversed at the same hierarchy level. It is used to constraint horizontal query scope
  2. **Query LifeTime (QLT):** Absolute deadline used to limit query lifetime
  3. **…**

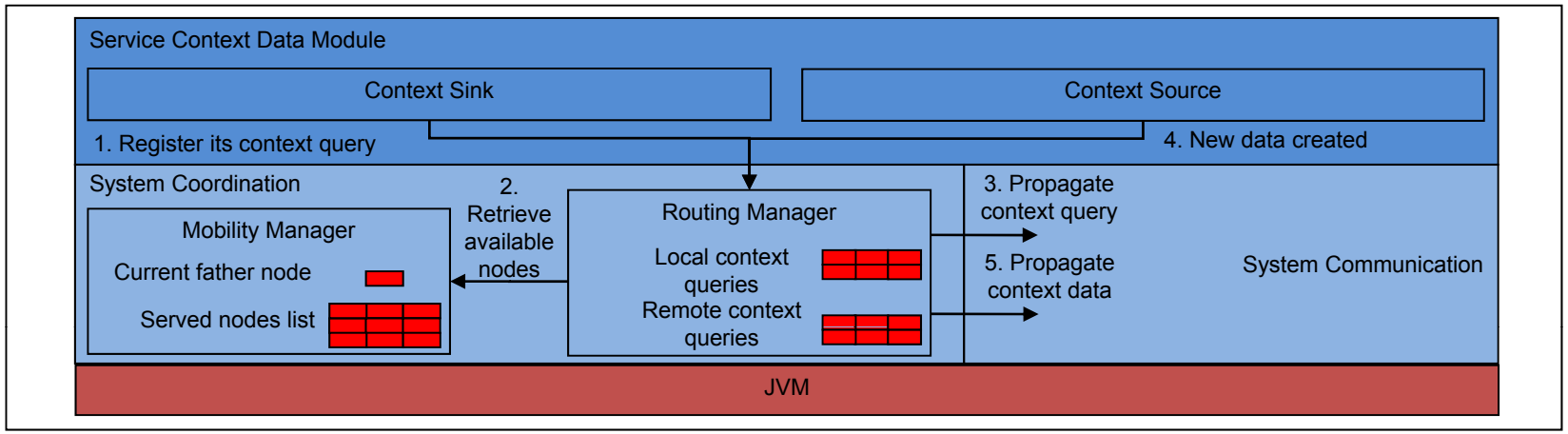
- SALES uses a space-efficient data structure, i.e., **Bloom filter**, to represent context queries

SALES middleware architecture



- **Facility Layer**
  - Each context data type is associated with a *Context Data Module*. The source enables the data injection, while the sink addresses the data retrieval
  - *Context Data Type Storage* maintains context data type definition
- **Mechanisms Layer**
  - *System Communication* offers communication primitives
  - *System Coordination* addresses mobility and data dissemination

Implementation insights



- *Routing Manager* has two different query tables:
  - **Local context queries** stores queries emitted by local sinks
  - **Remote context queries** stores queries received by other SALES nodes
- Local context sinks push local queries to the *Routing Manager* (step 1)
- When a query dissemination is needed, *Routing Manager* retrieves destination nodes, either peer nodes or father node, (step 2) and propagates the query (step 3)
- When new data are produced (step 4), they are matched against local and remote queries and propagated consequently (step 5)

Experimental evaluations

Implementation insights

- SALES Middleware has been completely realized on J2SE 1.6

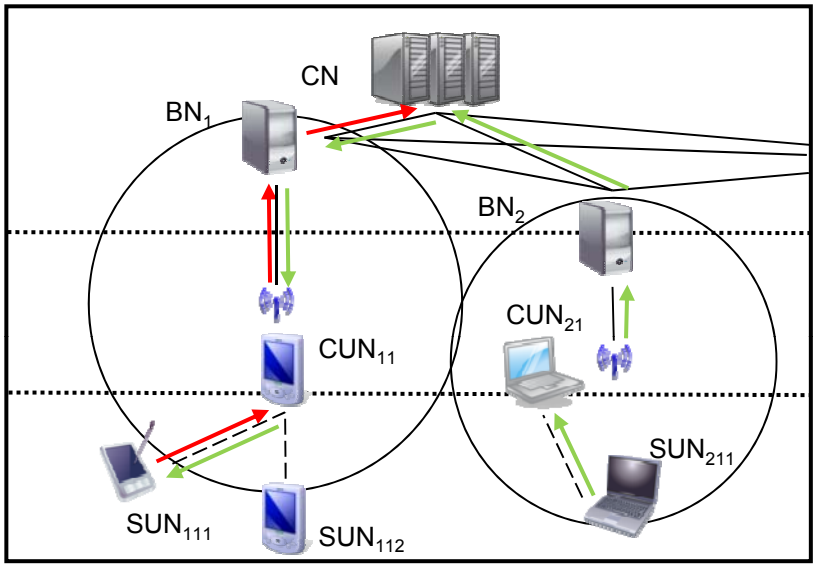
Experimental testbed

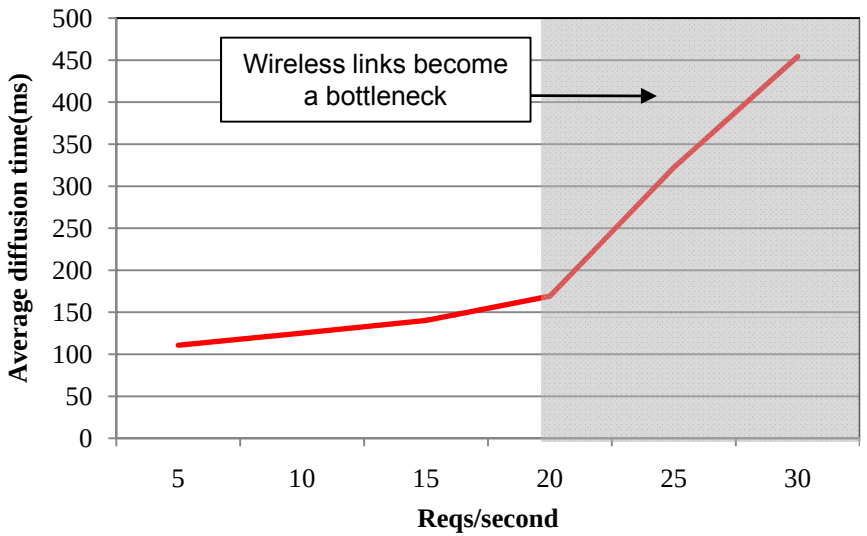
- BNs and CN execute on 2 CPUs 1,80GHz, 2048MB RAM, Linux Ubuntu
- Wireless infrastructure composed by Wi-Fi Cisco Aironet 1100 AP
- Test stations with IEEE 802.11g D-Link WDA-2320 and Linux Ubuntu
- Test code with 20 clients that send contemporary a variable number of context data requests

For testing purpose, we realized two different services:

- **Daily Advertisements Service:** disseminate context data related to lessons, seminaries, or general academic events
- **Service Discovery Facility:** handle discovery of our university services, like printers, projector, etc.

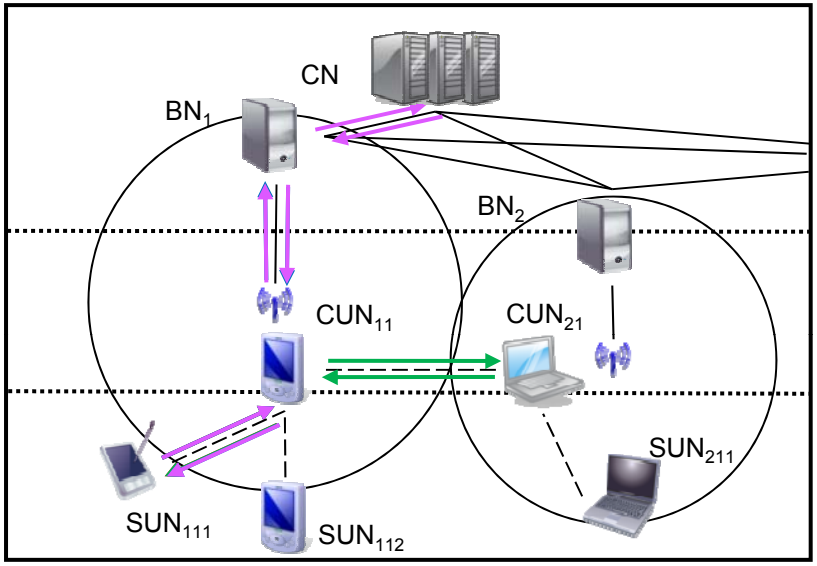
Daily Advertisements Service

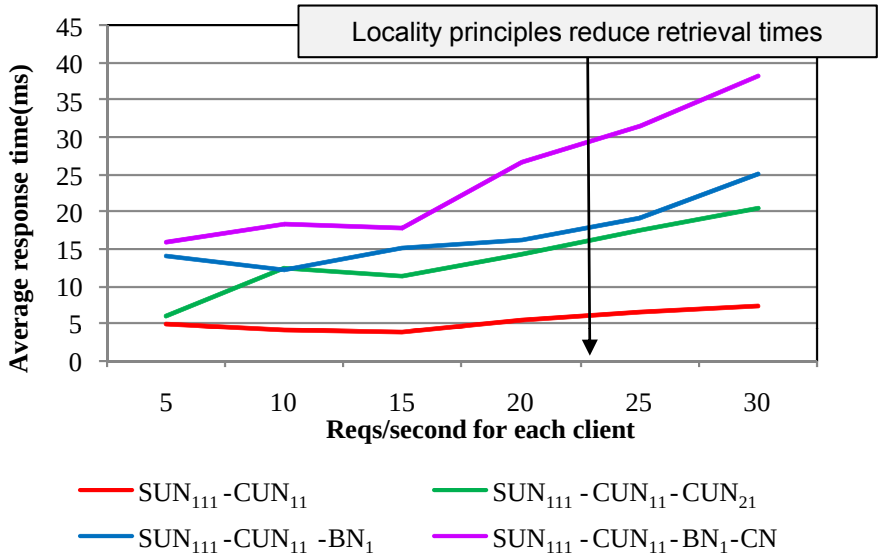




- **Daily advertisements** do not respect **physical locality principle**, and have traditionally a **short lifetime**
- Only CN stores these data

Service Discovery Facility





- **Service discovery advertisements** respect **physical locality principle**, and have traditionally a **long lifetime**
- All SALES nodes store these data and supply them when needed

Conclusions and ongoing work

Conclusions:

Data dissemination must be carefully addressed to obtain **scalability**

- **Locality** enhances scalability by constraining data dissemination scope
- **Mixing ad-hoc with infrastructure-based communications** reduces overhead
- **Different wireless communication standards** increase total available bandwidth

Ongoing work:

- **Multiple dissemination paths** to increase dependability
- **Different dissemination algorithms**, e.g., flooding- or gossip-based, according to data scope and environmental conditions
- Additional **context-aware applications** for our university campus

SALES project web site and contacts

- Prototype code and information:
  http://lia.deis.unibo.it/Research/SALES

- Contacts: Mario Fanelli (mario.fanelli@unibo.it)

Thanks for your attention!