# Adaptive Context Data Distribution with Guaranteed Quality for Mobile Environments

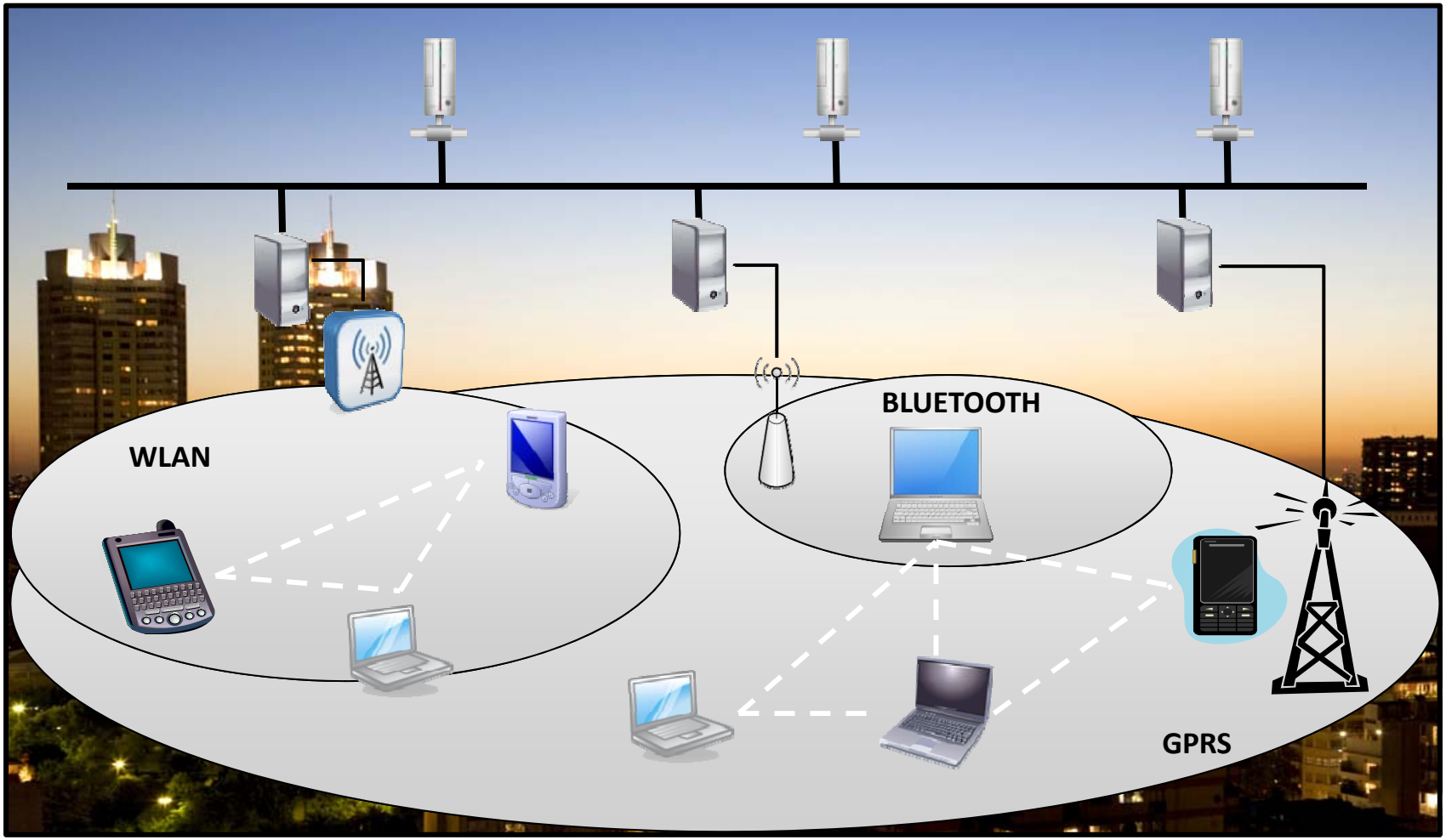
Antonio Corradi, Mario Fanelli, Luca Foschini

*Dipartimento di Elettronica Informatica e Sistemistica - Università di Bologna*

# Agenda

- Context data distribution in mixed infrastructured/ad-Hoc anvironments
- Quality of Context (QoC)-based context data distribution
- Scalable context-Aware middleware for mobiLe EnviromentS (SALES)
  - Data distribution process
  - Mapping QoC objectives to query parameters
- Implementation insights
- Experimental evaluations
- Lessons learned and ongoing work

# Context-aware applications in mobile and densely populated environments



# Need for Context Data Distribution Infrastructures (CDDI)

**Context data distribution** is a complex task that poses several challenging requirements:

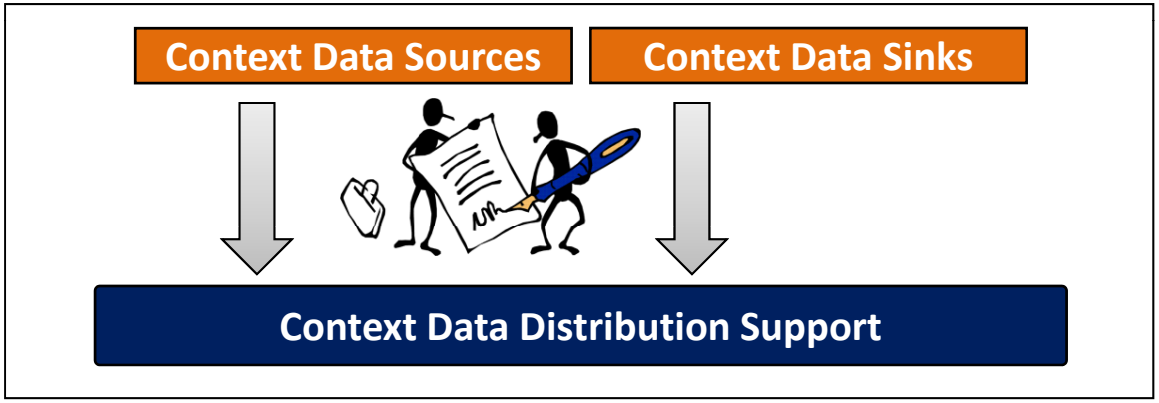
- **Heterogeneity** of the computing environment: devices (smartphones, Personal Digital Assistants, netbooks, …) and communication technologies (WiFi, Bluetooth, cellular 3G) and types (infrastructure and ad-hoc)
- **Device mobility** and **density**: ever-increasing number of mobile devices, already producing huge amounts of context data (environmental sensing, social computing, …)
- **Data delivery with guaranteed quality levels**: depending on specific service, (disaster recovery and emergency scenario, entertainment, …)

We claim the need for novel **Context Data Distribution Infrastructures (CDDIs)** to transparently address and take over context data distribution aspects (integration aspects, ***data distribution differentiation***, ***scalability***, …)

# QoC-based context data distribution

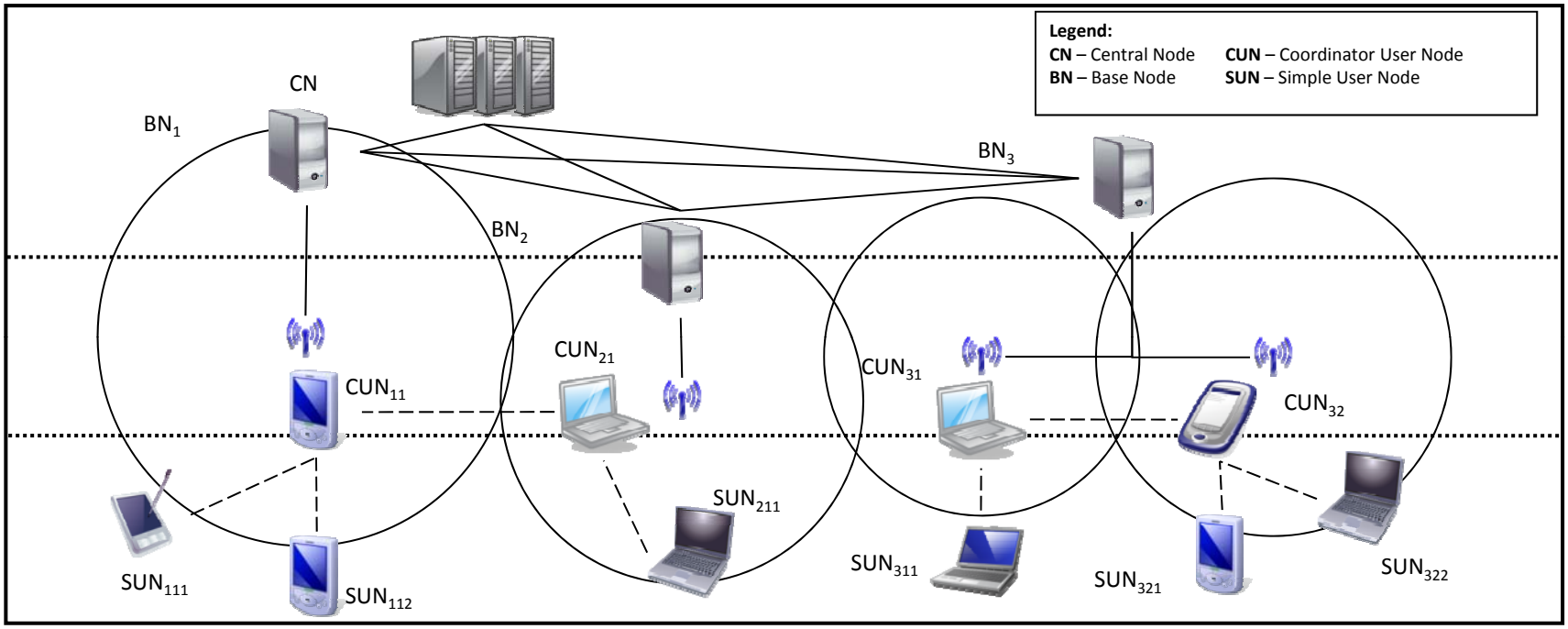
## Quality of Context (QoC)

- A concept usually applied to the quality of the distributed context data
- We extend this concept to tailor context data distribution and delivery so to save previous system-level resources




**Context Data Distribution Level Agreements (CDDLAs)** to detail the quality level the infrastructure has to ensure

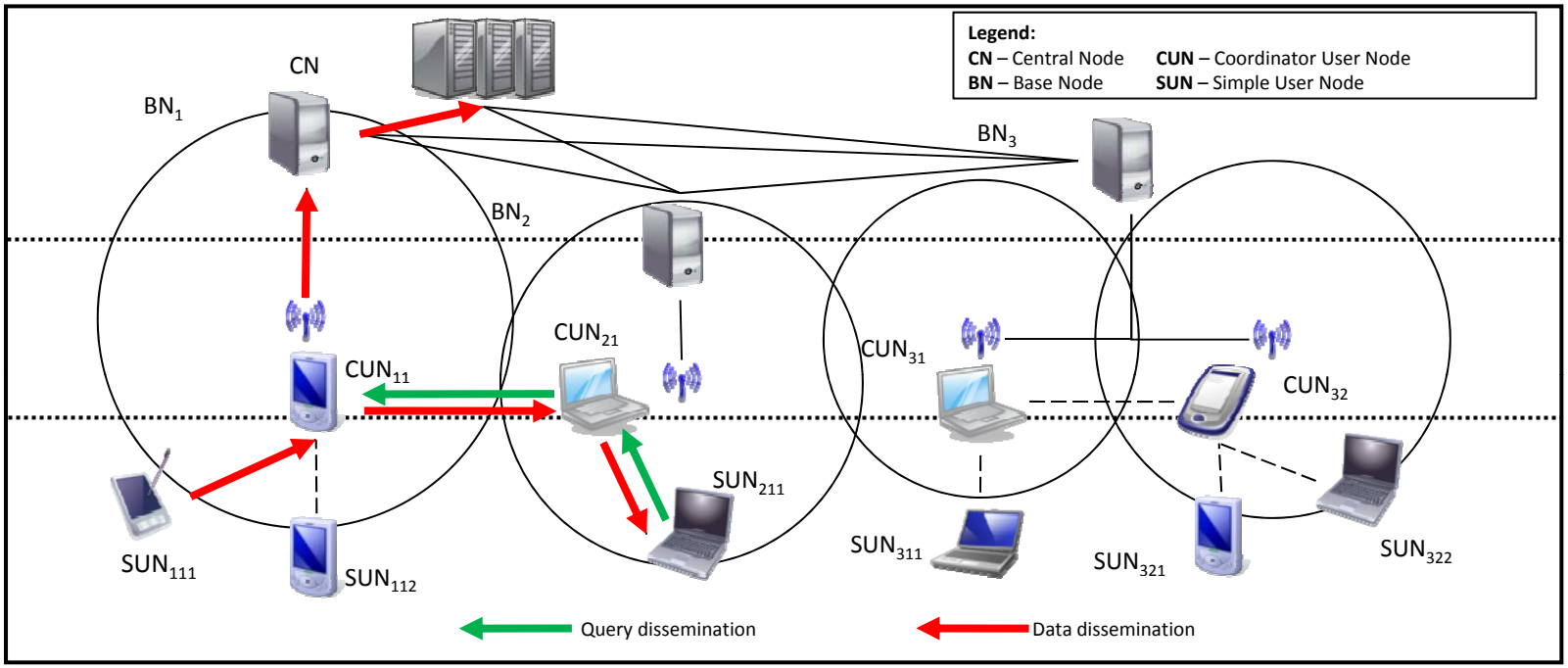
# Scalable context-Aware middleware for mobiLe EnvironmentS (SALES)




Three-level tree-like architecture

- minimizes tree depth to reduce management overhead
- ensures effective and integrated usage of wireless infrastructure and ad-hoc communication modes

# Data Distribution Process




- SALES distributes **context queries** to build dissemination paths
- Queries are disseminated both on the same level (horizontal distribution) and to the level above (vertical distribution)
- Data flow only on the bottom-up path between the data creator node and the CN
- Different dissemination paths are considered only when matching queries exist

# Data Distribution Process

SALES CDDLA comprises three main objectives:

1. ***Freshness*:** up-to-dateness requirement on matching data
2. ***Data retrieval time*:** the time needed by the mobile node to retrieve context data
3. ***Priority*:** traditional priority value used to enable traffic differentiation and to favor the routing of high-priority data under load conditions

To tailor query distribution, each SALES query has four main parameters:

1. ***Horizontal time-to-live (HTTL):*** used to limit the number of nodes traversed at the same hierarchy level
2. ***Routing delay (RD):*** used to delay query distribution to the next hop
3. ***Query total lifetime (QTL):*** used to handle query aging. When zero, the query is expired, and discarded by the system
4. ***Query priority (QP):*** used to enable priority-based query/data forwarding

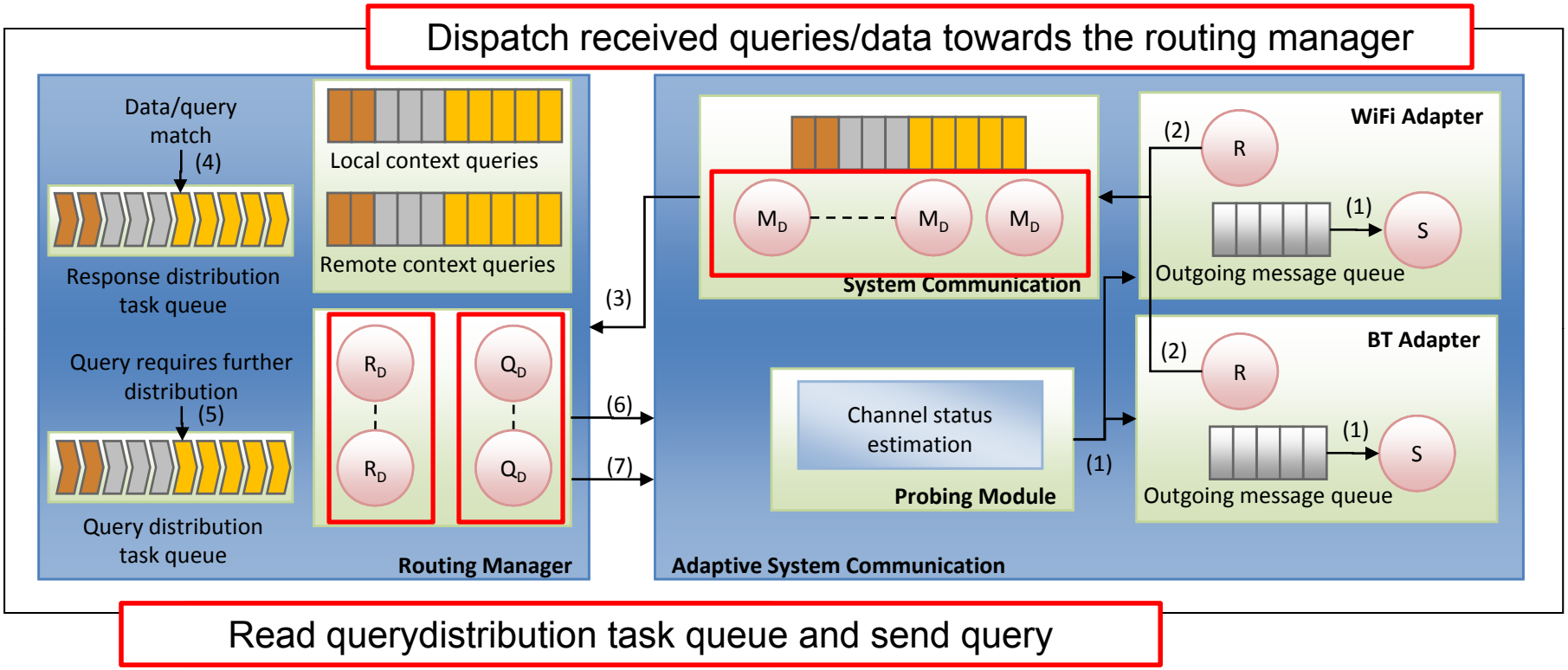
# Mapping QoC objectives to query parameters

For the sake of clarity, we define three principal users' classes:

- **Gold** [latest version of data, retrieval time 2s, priority 0]
- **Silver** [valid data, retrieval time 4s, priority 1]
- **Bronze** [non-valid data expired at most by 2 seconds, retrieval time 6s, priority 2]

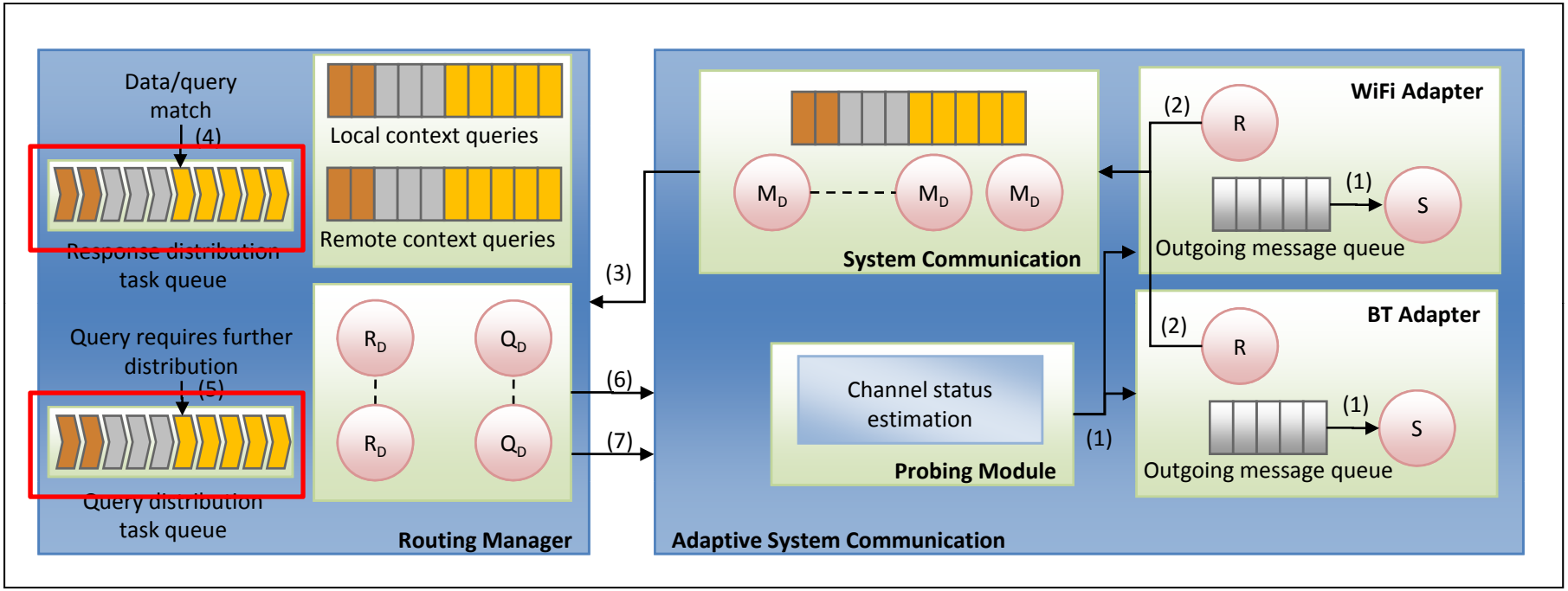
| Query Parameters | Depends on | Details |
|---|---|---|
| Query Total Lifetime (QTL) | CDDLA data retrieval time | Equal to the CDDLA data retrieval time |
| Query Priority (QP) | CDDLA priority | Equal to the CDDLA priority |
| Horizontal Time To Live (HTTL) | CDDLA freshness | HTTL = 0, 1, and 2 for respectively gold, silver, and bronze users |
| Routing Delay (RD) | CDDLA freshness and data retrieval time | RD is calculated considering the current level and HTTL |

# Implementation insights



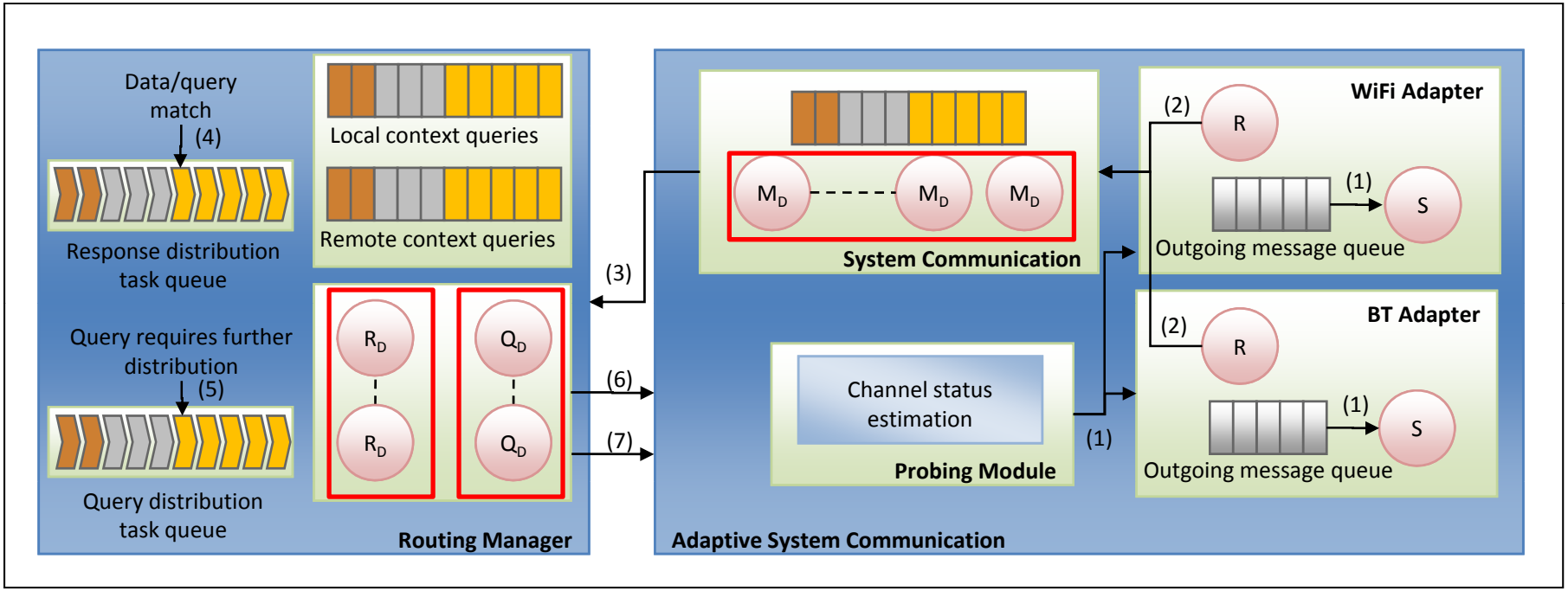

- Asynchronous architecture: operations scheduled by means of temporary descriptors
- Minimum intrusion principle → **Limited queue length, Limited processing**

# Implementation insights (cont'ed)




- **Query distribution task** queue ordered according to QP
- **Data distribution task** queue ordered according to the same priority of the matching queries that triggered data distribution
- Queue are partitioned: **50% of queue to priority 0 (gold users), 30% to priority 1 (silver users), and 20% to priority 2 (bronze users)**

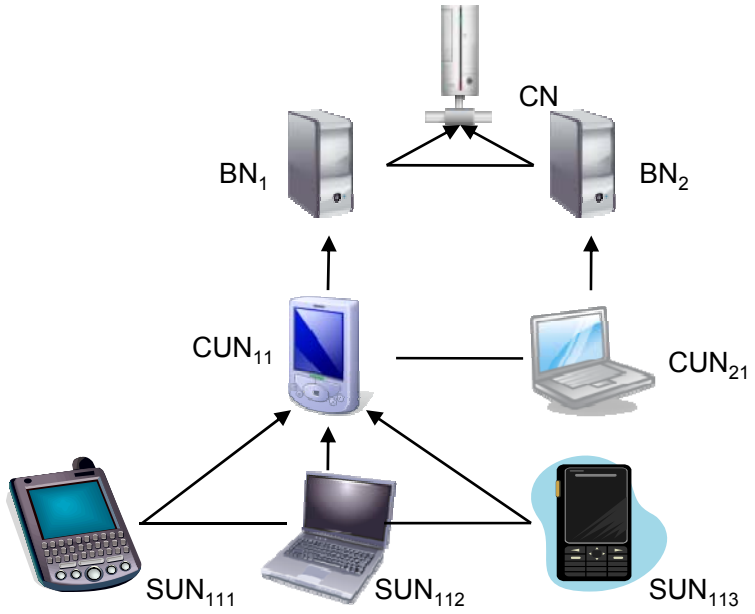
# Implementation insights (cont'ed)




- For each thread pool, we impose number of threads and maximum execution rate for single thread (number of tasks served for second)
- **Proactive approach** to keep queue limited: monitors the time elapsed between the expected (scheduled) task execution time and the effective execution time, and proactively discharge tasks

# Experimental testbed

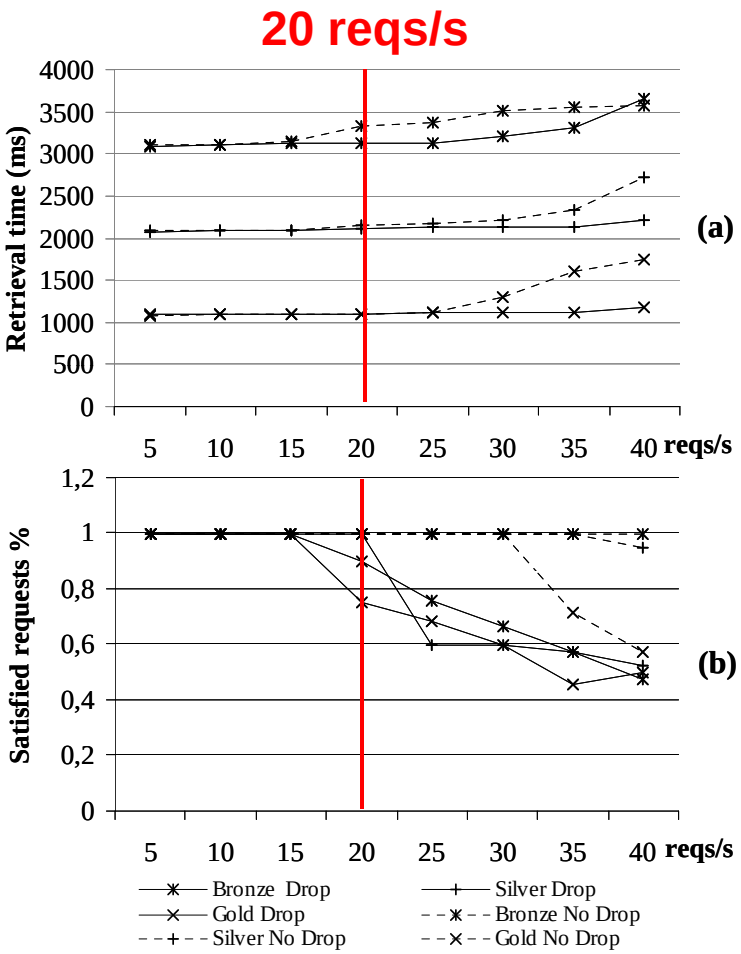
- Implementation insights
  - SALES has been realized on J2SE 1.6
- Experimental testbed
  - CN and BNs execute on 2 CPUs 1,80GHz, 2048MB RAM, Linux Ubuntu
  - Wireless infrastructure composed by Wi-Fi Cisco Aironet 1100 AP
  - Test stations with IEEE 802.11g D-Link WDA-2320 and Linux Ubuntu




- SALES configurations
  - Each SUN belongs to a different user class
  - CUNs/SUNs have 3 $M_D$, 1 $R_D$, and 1 $Q_D$ threads; processing rate 20 reqs/s
  - BNs and CN have 10 $M_D$, 3 $R_D$, and 3 $Q_D$ threads; processing rate 60 reqs/s
  - All queues are limited to 50 elements for CUNs/SUNs, and to 200 elements for CN/BNs

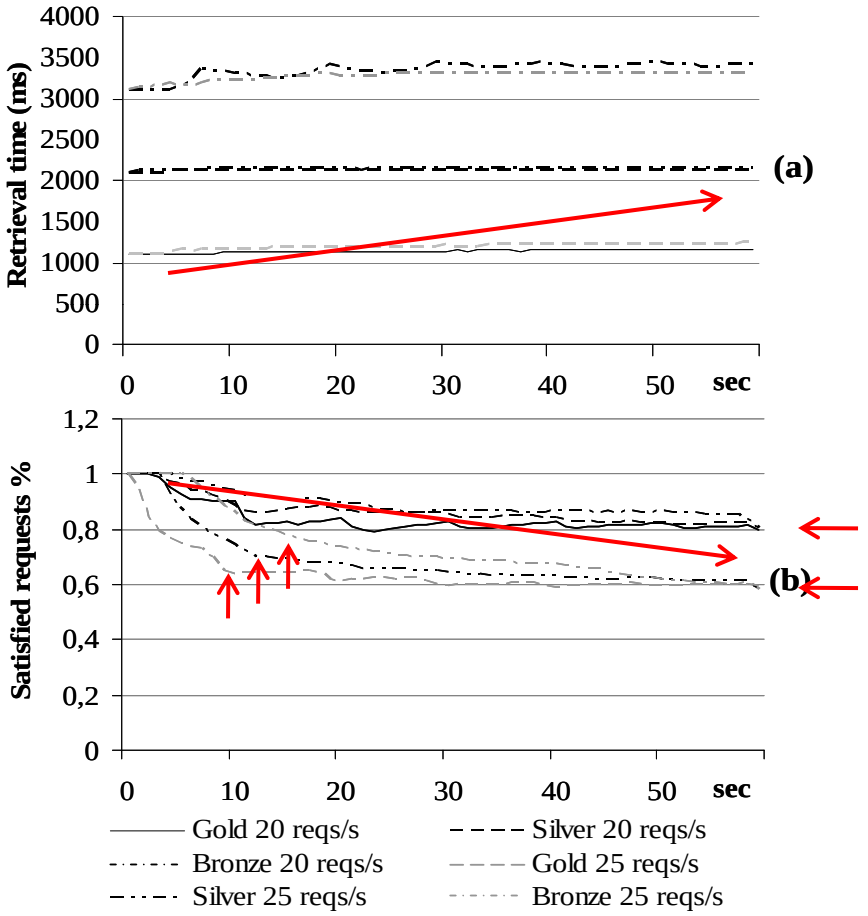
# Experimental results: SALES under overload conditions

- Query **dropping disabled** (dashed lines) → fewer failures, but higher average retrieval times
- Query **dropping enabled** (continuous lines) → close to the local processing capacity of 20 reqs/s, we experience request failures, but we guarantee limited average response times



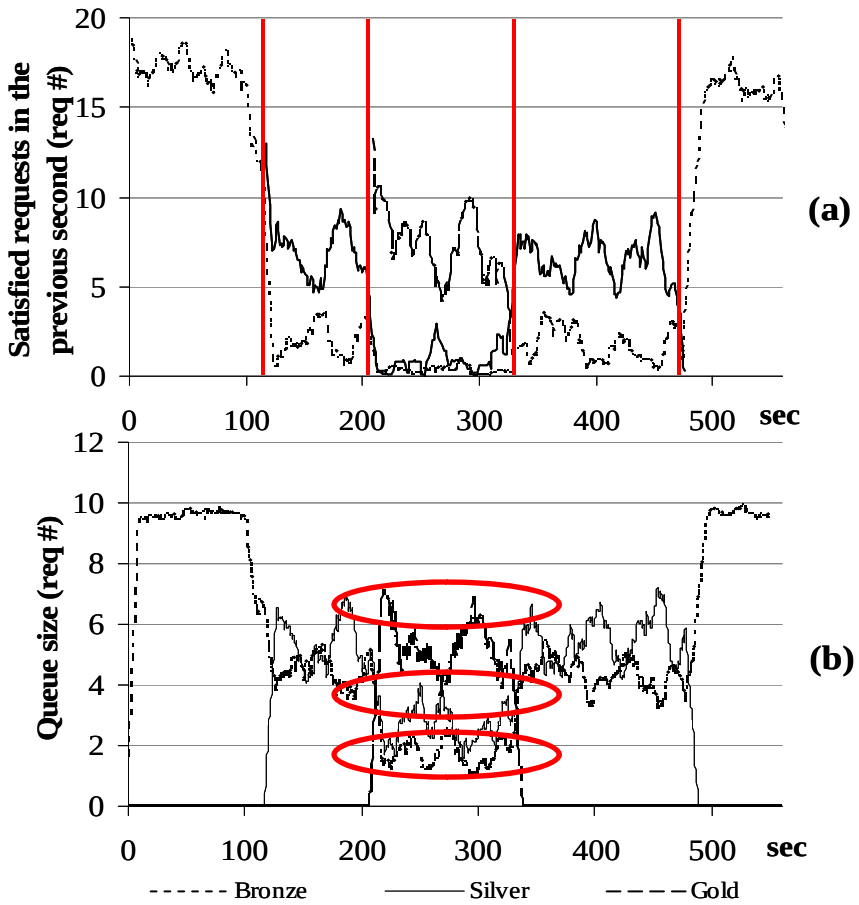
# Experimental results: long lasting runs with high load

- Query dropping enabled
- Worse values during time due to system saturation
- Different clients converge to the same drop rates → SALES exploits all available resources when traffic belongs to only one user class
- Clients belonging to higher classes can experience more failures due to the tight time constraints



# Experimental results: mixed traffic and CDDI differentiation

- 3 SUNs with different class (gold, silver, bronze); SUNs emit 20 reqs/s
- Each time a SUN of a higher class connects, low-priority SUNs experience higher message drops
- Each time a SUN of a higher class disconnects, low-priority SUNs experience lower message drops
- No one reaches its maximum queue limit → the queue dropping policy does not affect the reported results



# Conclusions and ongoing work

## Conclusions

- **CDDLAs** to drive CDDI management and obtain scalability
- **Minimal intrusion** to control CDDI overhead (CPU, memory, …)
- **Differentiation** to grant QoC (especially under congestion ☺)

## Ongoing work

- **Extensive simulations** to validate our approach in wide deployments composed of several nodes (SALES on ns-2 ☺)
- **Extension of the CDDLAs** to introduce data-related quality
- **Different dissemination algorithms**, e.g., flooding- or gossip-based, according to data scope and environmental conditions

# 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!