A Coordination Protocol for Distributed Context Management Systems

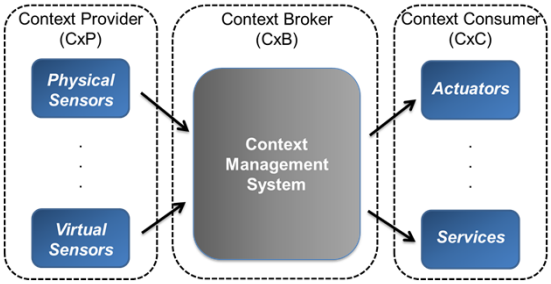
Jörg Schneider, Christian Mannweiler, Andreas Klein,
Hans D. Schotten
23.07.2012

Contents

University of Kaiserslautern
Department of Electrical Engineering
and Information Technology

EuroView 2012

1. Motivation



- A Context Management System (CMS) offers the ability to:
  - reuse available resources (transport medium and context data)
  - manage context data from physical and virtual sources
  - allow support of niche applications

Motivation

- Key requirements for Context Management Systems (CMS) include:
  - Collecting data of any format
  - Enabling dynamic reconfiguration
  - Supporting real-time context data processing
  - Guaranteeing scalability and extendibility
- Examples for applications domains:
  - Transport and logistics
  - Personalized services
  - Smart grid
  - E-tourism
  - Agriculture and vehicular technology
  - Network services
- Reference implementations of context management middleware platforms are Gaia, PACE, Confab, and AURA CIS

Contents

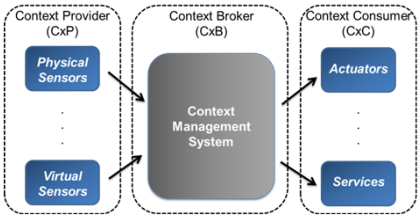
2. Context Management Systems

- Key requirements for context management systems (CMS):
  - Plug-and-Play: automatic discovery and configuration
  - Scalability: meaningful data distribution between nodes
  - Fault detection and recovery in case of node failure
  - Support for mobile subsystems
  - Efficient implementation, i.e. utilization of
    - Standardized data formats
    - Wide-spread transmission protocols
  - Additional requirements: data security, access control, …

Contents

1. Motivation
2. Context Management Systems
3. **Centralized Context Management Systems**
4. Distributed Context Management Architecture
5. Conclusion

3. Centralized Context Management Systems

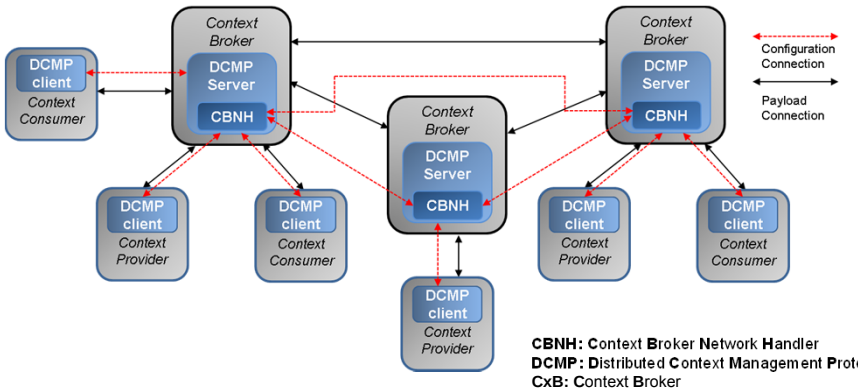


"Over-The-Top" implementation
REST-ful signaling

- Weakness of centralized system architecture:
  - Scalability problems
  - Limited support for mobile subsystems
- Solution:
  - Extension of the centralized context broker architecture to a distributed one
  - Introduction of Distributed Context Management Protocol (DCMP)

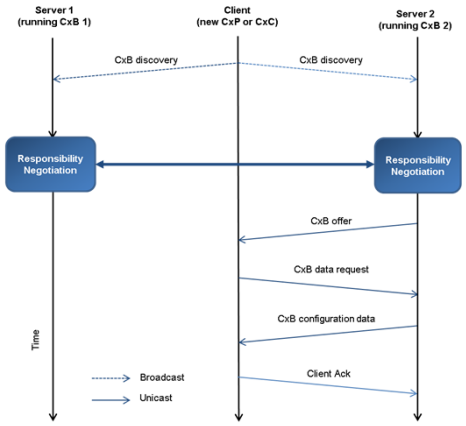
Contents

4. Distributed Context Management Architecture



Distributed Context Management Architecture

Auto discovery function of DCMP:

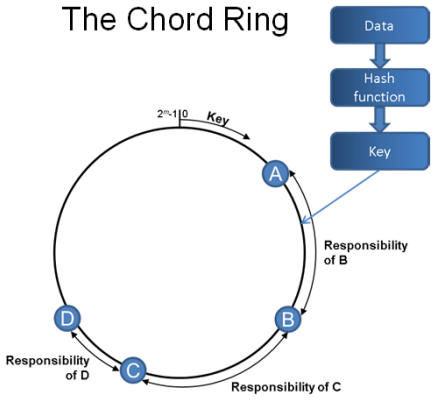


- Each client (CxC or CxP) sends a "***CxB discovery***" request
- Assignment to CxB is reconciled between CBNH entities (DHT based)
- Responsible CxB offers all necessary configuration data to the client
- CxP advertisement is similar to the centralized architecture

Distributed Context Management Architecture

Distributed Hash Tables (DHT)

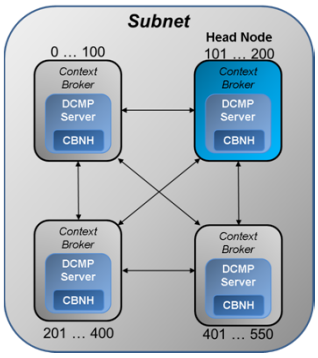
The Chord Ring



- Motivated by peer-to-peer systems and usually realized as "on the top" implementation
- Belongs to the class of decentralized distributed system that provides a lookup service
- Each node chooses a n-bit ID
- Each lookup key is also a n-bit ID
- Each node is responsible for storing keys "near" its ID
- Data keys are generated via hash functions

Distributed Context Management Architecture

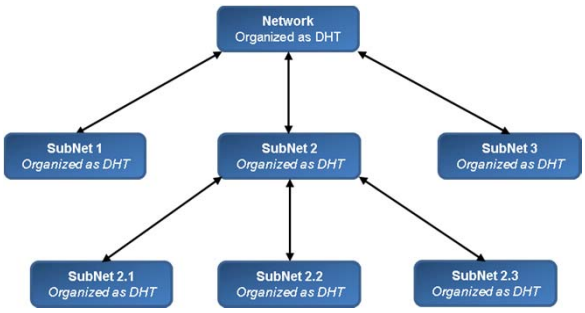
Reconcilement



- System consists of *m* CxBs
- Introduction of a CxB efficiency factor *α* (indicating CxB computing performance)
- Each CxB is responsible for a defined "range" of context data (keys)
- Distribution of clients between CxBs is organized with hash functions
- CBNH is responsible for key calculation

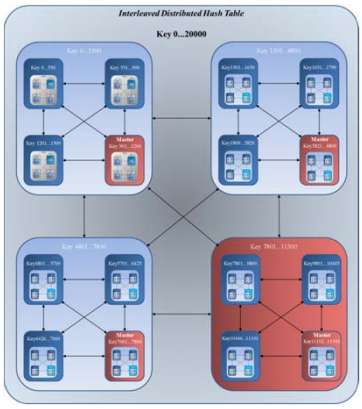
Distributed Context Management Architecture

Hierarchical Subnet Organization



- Each subnet is organized as a DHT
- One CxB in a subnet acts as "*Head Node*"
- Head Node is responsible for communication with upper network
- Key range of the Head Node in the upper network is equivalent to the key range of the whole subnet

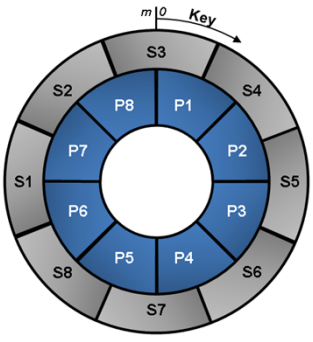
Nested Distributed Hash Table



- First CxB becomes Head Node and is responsible for communicating with nearby sub-network
- Receives configuration data about key range responsibility of the subnet
- Responsible to determine key ranges of each CxB in the subnet with respect to its efficiency factor α, upper key bound key_{up}, and lower key bound key_{lo}

Distributed Context Management Architecture

Primary/Secondary CxB Backup Concept



Px: Primary Broker
Sx: Secondary Broker

- System needs a “backup” solution to be resistant against failures of entities
- Each CxB has a primary and a secondary key range
- Storage of data and key responsibility has to be performed on the primary and secondary CxB
- Redundancy increases system stability

University of Kaiserslautern
Department of Electrical Engineering
and Information Technology
EuroView 2012

Contents

5. Conclusion

- Resolved design challenges
  - Auto-configuration function has been incorporated
  - Broker architecture has been redesigned as distributed system by exploiting DHTs
  - Full compatibility with mobile brokering subsystems has been achieved
  - Primary/Secondary broker concept ensures a fault-tolerant system
- Proof-of-concept implementations have shown that our architecture fulfills the requirements of scalability, extensibility, and mobility

Thank You for your attention!

Questions?