Ad hoc Service Grid – A Self-Organizing Infrastructure for Mobile Commerce

Klaus Herrmann, Kurt Geihs, Gero Mühl

Berlin University of Technology
Email: klaus.herrmann@acm.org
Web: http://www.ivs.tu-berlin.de/Herrmann/

Oslo, Norway, September 17th 2004
Outline

> Ad hoc Service Grid
  > General vision, advantages, and challenges

> Research Focus

> Self-organizing Service Distribution

> Complementing Concepts

> Summary and Conclusions
Wireless Services at medium-sized Locations

> Locations:
  > Construction sites, hospitals, shopping malls etc.

> Services (e.g. at a shopping mall)
  > Local, facility-specific services for local users
  > Examples: navigation, product finder, reservation (e.g. restaurant)

> Using cellular phone networks
  > Non-local communication, expensive, low-bandwidth

> Using WLAN access point technology
  > Wiring is extremely expensive(!), inflexible, centralized server
Ad hoc Service Grid

> Basic idea: Use an ad hoc network
  > Distribution of PC-like computers (*Service Cubes*) at the location
  > Wireless network interface, power connector, no peripherals
  > Direct communication between neighboring *Service Cubes*
  > Multi-hop communication between Cubes that are further apart
  > Users access services via nearest *Service Cube*

> Advantages
  > Communication is free of charge, modest expenses for setup
  > No high initial expenses for monolithic central server
  > Flexibly scalable: adding or removing Cubes during runtime is easy
Example Setup: Shopping Mall
General Challenges

> Decentralization and Self-Organization
  > Distributed resources → Control and organization is difficult

> Service infrastructure should be invisible
  > Minimal manual interventions
  > Self-organize and adapt to changing conditions

> Personalization vs. privacy and security
  > Offer personalized services while providing privacy
  > Interactions must be secure

> Business Models
  > Indirect revenue
Current Research Focus

> Self-organizing dynamic service distribution
  > Dynamic replication and node selection to meet current demand
  > Maximize QoS: response times perceived by users
  > Minimize network load, balance processing load

> Service lookup and discovery
  > Enable users to discover services and find best service replica

> Data consistency
  > Achieve data consistency among replicated stateful services

> What does an overall ASG Middleware/Serviceware look like?
Self-organizing Service Distribution

- Installation: one service replica positioned arbitrarily
- Clients start accessing the service
  - Assumption: Spatial distribution of requests is non-uniform
  - General Approach: Use request patterns to guide distribution
  - Clients always choose closest service
  - Request tree $T$ is recorded at each service replica’s Cube
  - Service is replicated or migrated to request hot spots
Distribution algorithm

> Runs periodically at the replica’s Cube
  > Compute weighting function $M_n$ for each node $n$ in the tree
  > Find nodes $i$ and $j$ in request tree $T$ such that
    > $i$ and $j$ are not in the same subtree
    > $M_i > M_j > M_k$ for all $k$ with $k \neq i \neq j$
  > Migrate service to node $i$ if it is dominating ($M_i >> M_k$ for $i \neq k$)
  > Replicate service to $i$ and $j$ if both are dominating and the service-specific replica limit has not been reached
  > Dissolve replica if idle for too long
Weightning Function $M_n$

\[ M_n = (D_n + 1) \sum_{i=t-k}^{t} R_n(i) \]

> Informally: Number of transmissions caused by $n$

> Inputs

  > $D_n$: Hop Distance of node $n$ from service’s node

  > $R_n(i)$: Number of requests transmitted by node $n$ at time index $i$

  > $t$: the current time index

  > $k$: length of relevant request history time window
Simple Example

![Diagram showing network nodes and request production.]

- **Requests produced:**
  - First scenario: 10, 10, 10, 10, 10
  - Second scenario: 10, 10, 10, 10, 10

- **$R_n(i)$:**
  - First scenario: 50, 40, 30, 20, 10
  - Second scenario: 10, 20, 50, 20, 10
  - Total: \( \sum = 100 \) (transmitted)

- **$D_n+1$:**
  - First scenario: 1, 2, 3, 4, 5
  - Second scenario: 3, 2, 1, 2, 3

- **$M_n$:**
  - First scenario: 50, 80, 90, 80, 50
  - Second scenario: 30, 40, 50, 40, 30
  - Total: \( \sum = 60 \) (transmitted)
Oscillation Avoidance

> Maintain a history of adaptations performed locally at each node

> Adaptation = (Destination, Request Tree)

> Check for past adaptations with similar Request Trees before performing an adaptation

> Similarity of two trees $T_1$ and $T_2$ is given by

$$s(T_1, T_2) = 1 - \frac{\sum_{i \in N_1 \cup N_2} \left| M_i^1 - M_i^2 \right|}{\sum_{i \in N_1 \setminus \{r_1\}} M_i^1 + \sum_{i \in N_2 \setminus \{r_2\}} M_i^2}$$

with

- $M_i^j = 0$ iff $i \notin N_k$
- $N_k$ : Set of node IDs from $T_k$
- $r_k$ : ID of root node from $T_k$
Emergent Effects

- Replicas find positions where traffic is balanced
  - None of the nodes involved in the request flow stands out in terms of network load produced (no *dominating* nodes)
  - Tunable parameter: *Domination Factor*

- Preset limit on per-service number of replica controls the average distance between service and clients
  - Tunable Parameter: *Replica Limit*

- Oscillation avoidance reduces unnecessary adaptations while still keeping the system reactive
  - Tunable Parameter: *Similarity Threshold*

- Processing load is balanced
  - Replication and choice of nearest service by clients
Result – Adaptive Reduction in overall Traffic
Work not Covered in the Talk

> Distributed *lookup service* for mobile services
  > Forwarding of client requests to current service location
  > Lazy propagation of location changes by snooping meta information piggybacked in service replies → Self-repairing

> Data consistency in stateful services
  > Weak, optimistic consistency model (inspired by Bayou)
  > Current work!

> Architectural implications on overall middleware
  > Putting it all together…
  > Past, current, and future work!
Summary and Conclusions

> *Ad hoc Service Grid*: Basic vision for a service provisioning platform for medium-sized locations

> Conceptual groundwork (algorithms and protocols)

> Self-organizing service distribution

  > Simple, usage-driven algorithm

  > Transmission hot spots attract services until network load is balanced

  > Oscillation is damped while the system remains reactive to changes

  > Network load is reduced
Thank you.

Question and comments are welcome.

Klaus Herrmann
klaus.herrmann@acm.org
Intelligent Networks and Management of Distributed Systems
Berlin University of Technology
www.ivs.tu-berlin.de