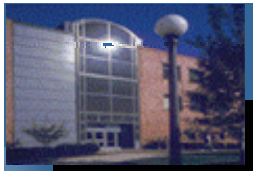
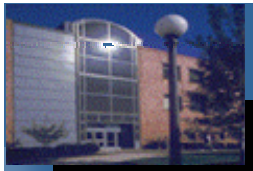
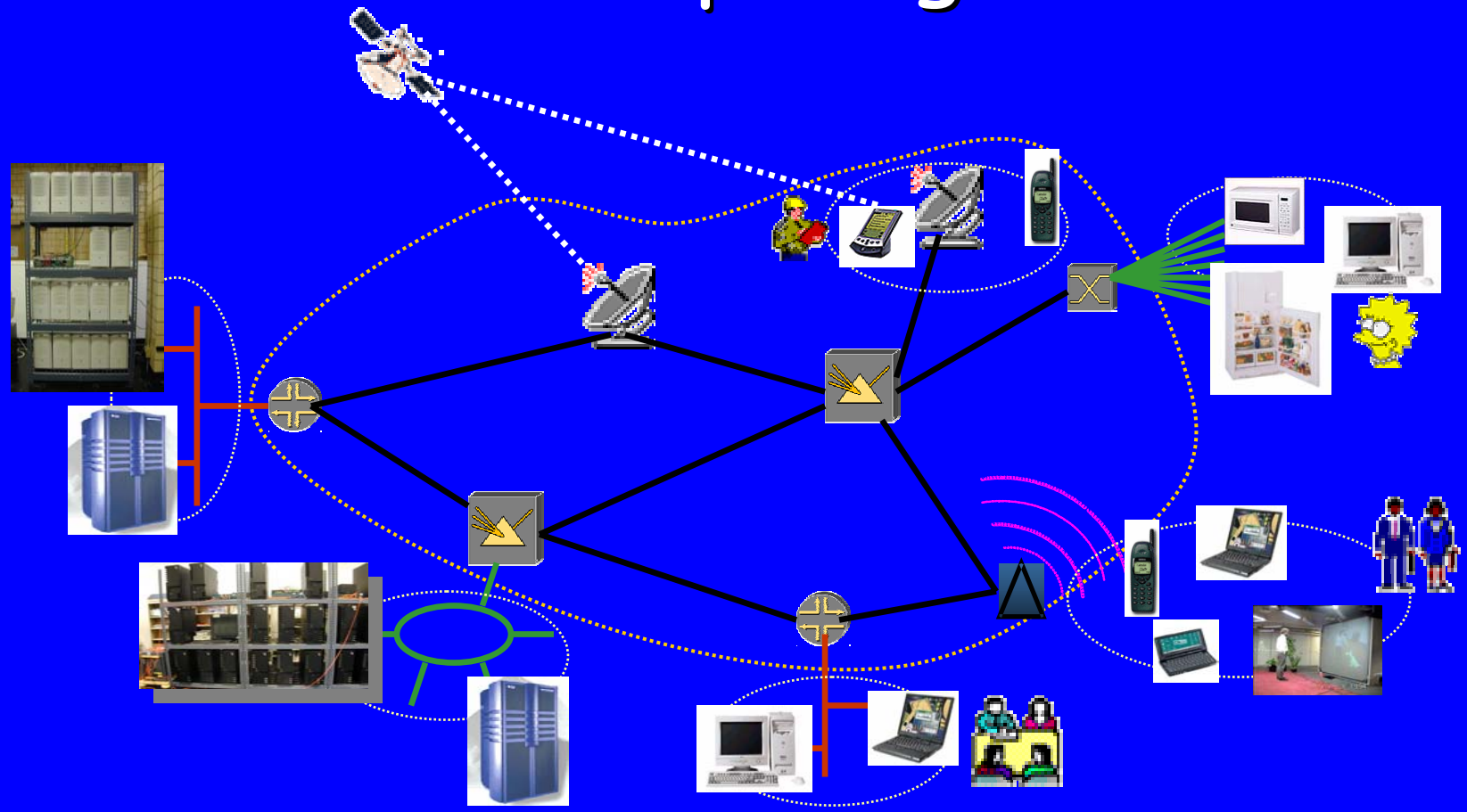


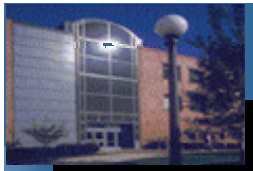
QoS-Aware Service Composition

Klara Nahrstedt
Joined work with Xiaohui Gu and Jingwen Jin
klara,xgu,jjin1@cs.uiuc.edu
University of Illinois at Urbana-Champaign

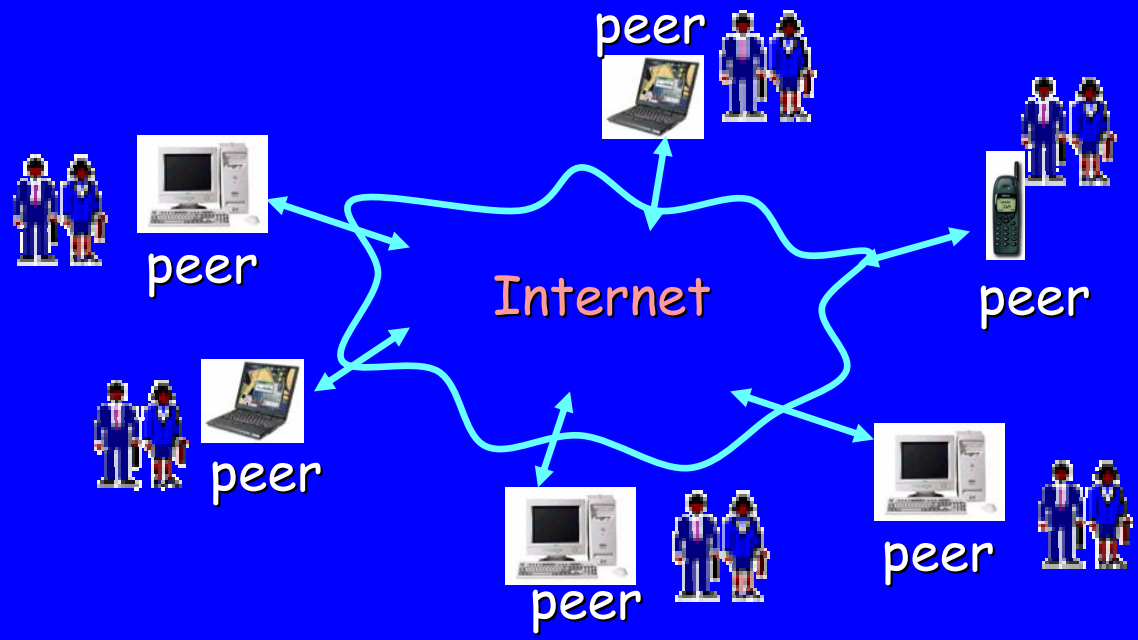


Research Context: Ubiquitous Computing

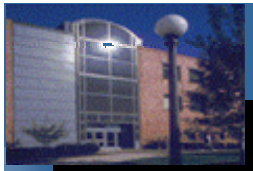




Research Context: Peer-to-Peer Computing Grids

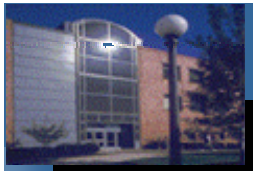


Peer to peer (P2P) computing grids consist of many peer nodes that communicate directly among themselves through Internet and can act as both clients and servers. They are distributed systems without any centralized infrastructure or hierarchical organization.



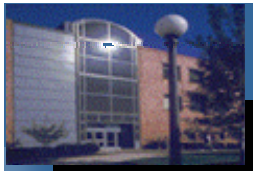
Why Dynamic QoS-aware Service Composition and Distribution ?

- Flexible Service composition
  - compose a QoS-aware distributed application using service components, discovered in the current environment and possibly provided by different service providers
  - accommodate runtime changes and heterogamous clients
- Resource-informed service distribution
  - overcome resource limitations of mobile devices
  - provide load balancing, selection of service proxies
  - maintain quality levels in case of runtime changes (e.g., old devices crash or new devices are added)



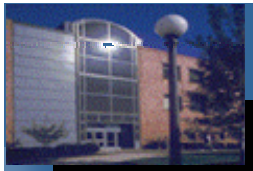
Presentation Outline

- Existing Approaches
- QoS-aware Composable Service Model
- Dynamic QoS-aware Service Composition
- Resource-based Service Distribution and Service Routing
- Experimental results
- Conclusions



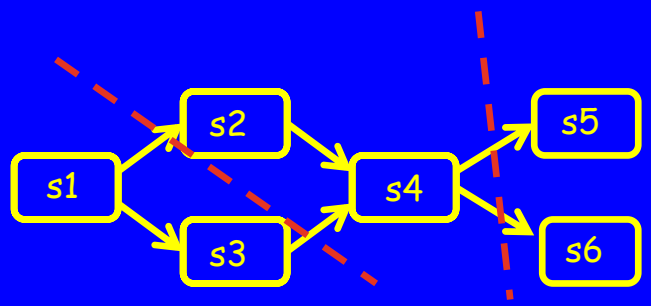
Existing Approaches

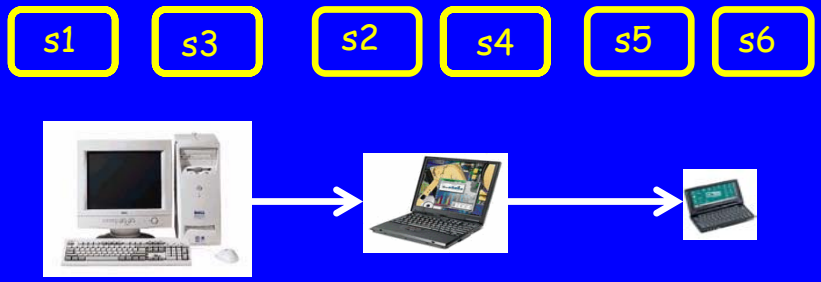
- Service composition
  - PATHS (Stanford): linear service path, media type mediations
  - SAHARA(Berkeley): robust, fault-tolerant service composition
  - 2KQ (UIUC): Representing a distributed application using multiple pre-defined service paths
- Service distribution
  - Load balancing among peer servers
  - Coign: automatically distribute binary application at setup time.
  - AOS: partition application at finer granularity, e.g., Java Class
- None of them addressed both aspects in a coordinated, unified framework.
- Other relevant research in P2P Systems (Gnutella, Napster, CAN, Cord), Web server selection, etc.



Dynamic, QoS-Aware Service Composition for Ubiquitous Environment

- An integrated framework to address BOTH aspects
  - Service composition tier: choose and compose current available service components and coordinate arbitrary interactions between them;
  - Service distribution tier: distribute service components to different available devices



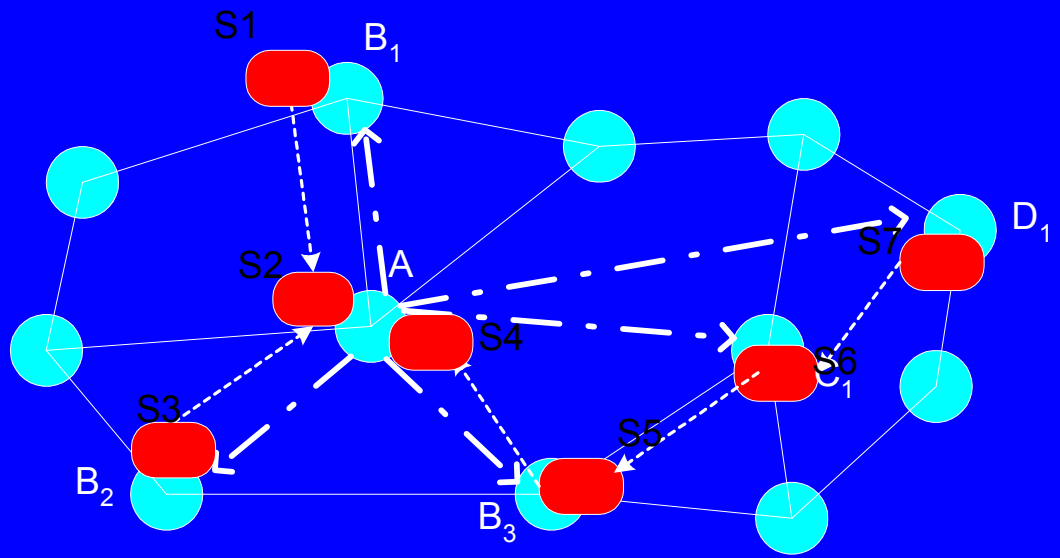


QoS-aware Service Composition for P2P

App1: S1 -> S2;

App2: S3 -> S2;

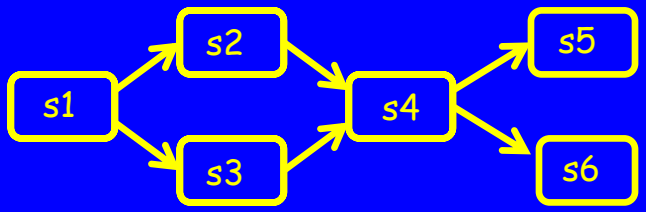
App3: S7 -> S6 -> S5 -> S4;



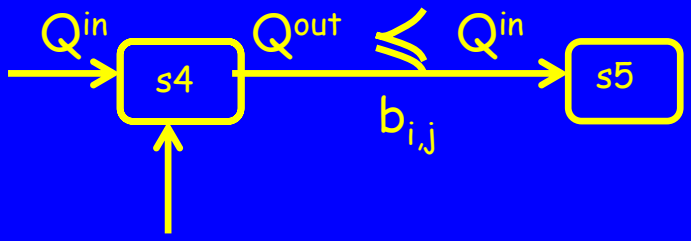
Exploit P2P Redundancy

- each service can have multiple instances with different QoS
- each service instance can be located on multiple peers

QoS-aware Composable Service Model



Service Graph (DAG): nodes represent service components and arcs represent data flows

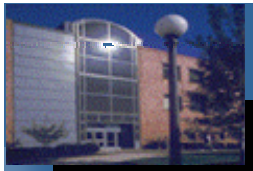


R = f(Q^{in}, Q^{out})

$Q^{in} = [q_1^{in}, q_2^{in}, \ldots, q_n^{in}]$, $Q^{out} = [q_1^{out}, q_2^{out}, \ldots, q_n^{out}]$, $R = [r_1, r_2, \ldots, r_m]$;

DEFINITION: QoS Consistency relation called satisfy "$\preccurlyeq$":

$$\forall i, 1 \le i \le Dim(Q_B^{in}),\ \exists j, 1 \le j \le Dim(Q_A^{out}),$$
$$q_{Aj}^{out} = q_{Bi}^{in},\ \ if\ q_{Bi}^{in}\ is\ a\ single\ value;$$
$$q_{Aj}^{out} \subseteq q_{Bi}^{in},\ \ if\ q_{Bi}^{in}\ is\ a\ range\ value.$$



Service Composition Tier

Acquire the Abstract Service Graph and Translate User's Request (e.g., QoS Compiler).

↓

Discover service instances (e.g., Chord, CAN).

↓

Check QoS consistencies, coordinate ad-hoc interactions, and compose QoS consistent path.

↓

Generate the QoS consistent service graph and deliver it to the service distribution tier.

1. Failed discovery of service instances;

Solution: recursively apply the service composition algorithm

2. Fast and efficient QoS consistency check;

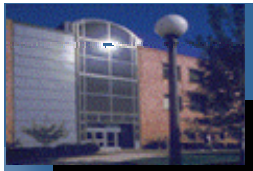
Solution: Ordered Coordination algorithm

3. Automatic correction of inconsistent interactions.

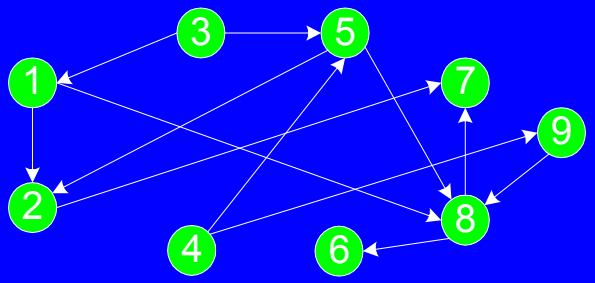
Solution: Ordered Coordination algorithm

4. Selection among multiple QoS paths

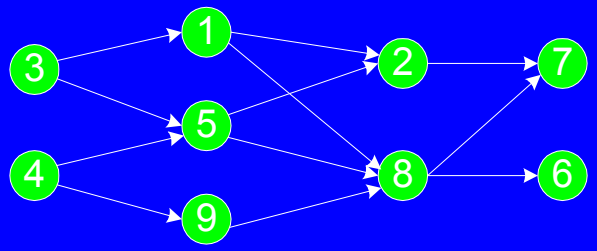
Solution: Shortest path



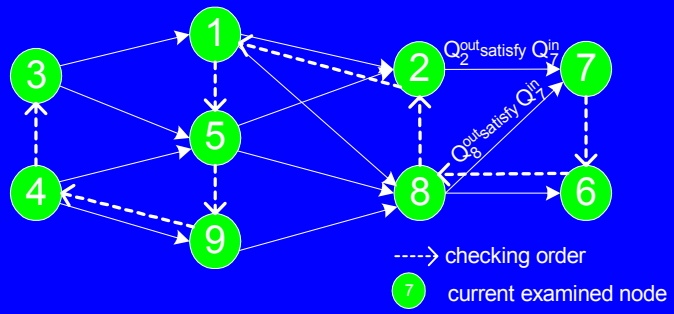
Ordered Coordination Algorithm



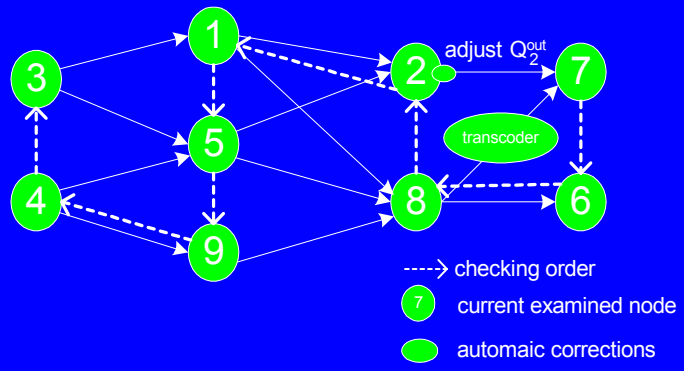
(a) the Original Service Graph



(b) the Topological Sorting of the Original Service Graph

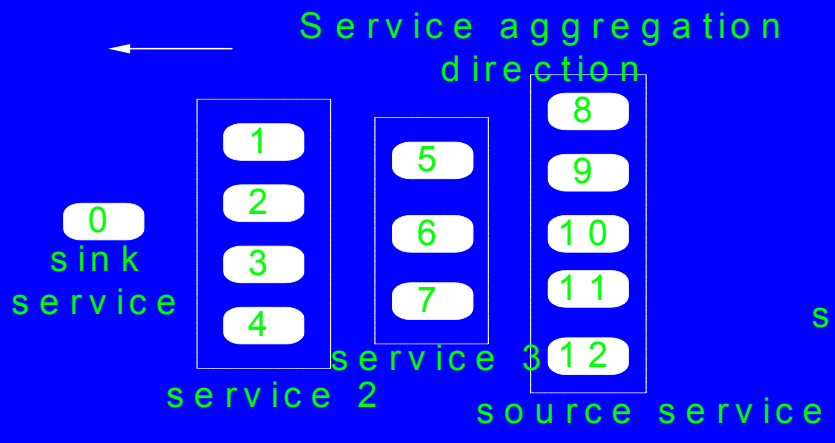


(c) Check QoS Consistency in the Reverse Order of its Topological Sorting

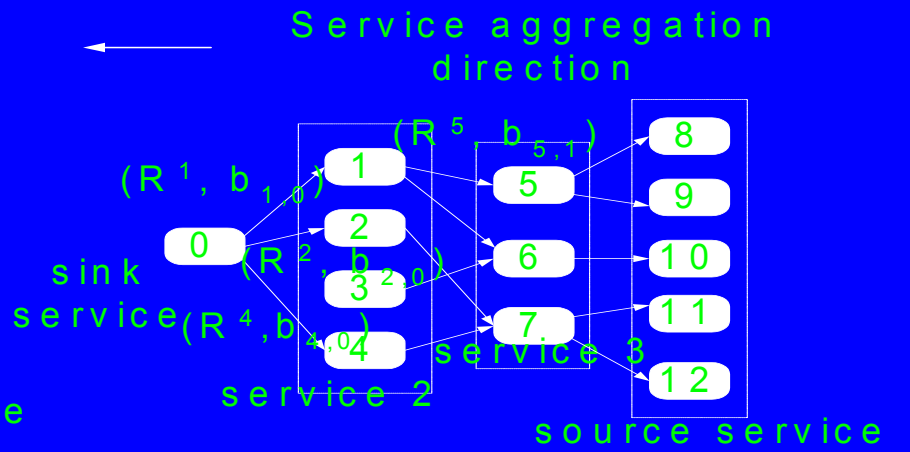


(d) Automatic Corrections on QoS Inconsistencies

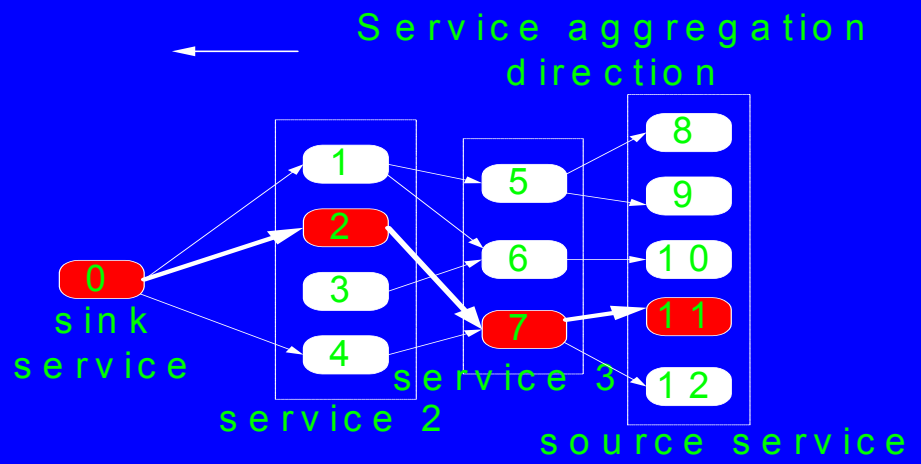
QoS Consistent Shortest Service Path



(a) Service instances for all required services

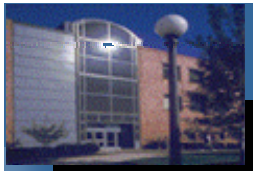


(b) Add edges between two QoS-consistent service instances



(c) Find the shortest service path

QoS-consistent, resource-shortest serivce path

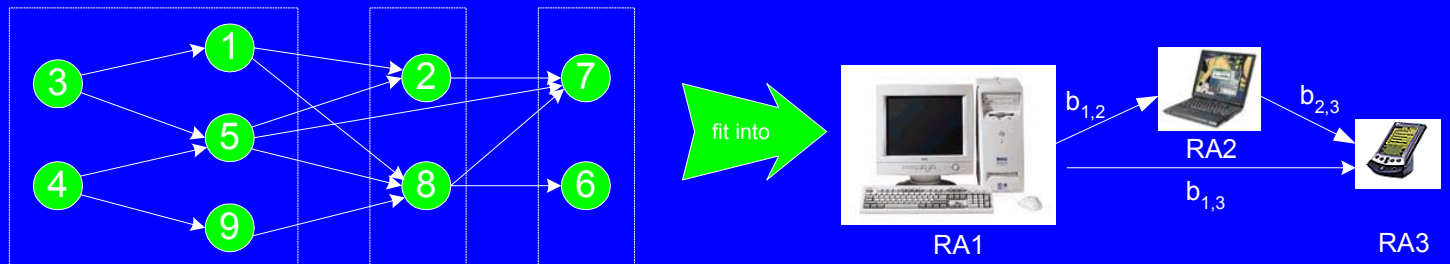


Service Distribution Tier

Mapping from Abstract Composition to Physical Infrastructure

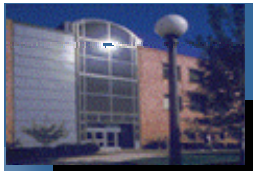
- Service Distribution For Ubiquitous Environment – K-cut
- Service Distribution For P2P – Distributed Peer Selection
- Service Distribution For P2P – Service Multicast

Service Distribution Tier



Optimal Service Distribution(OSD) problem is defined as finding the k-cut (k represents the number of available devices) that makes the service graph fit into k devices and also minimizes the cost aggregation.

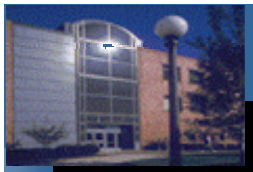
The cost aggregation is defined as the weighted sum of the ratios between the resource requirements and the resource availabilities.



Service Distribution Tier

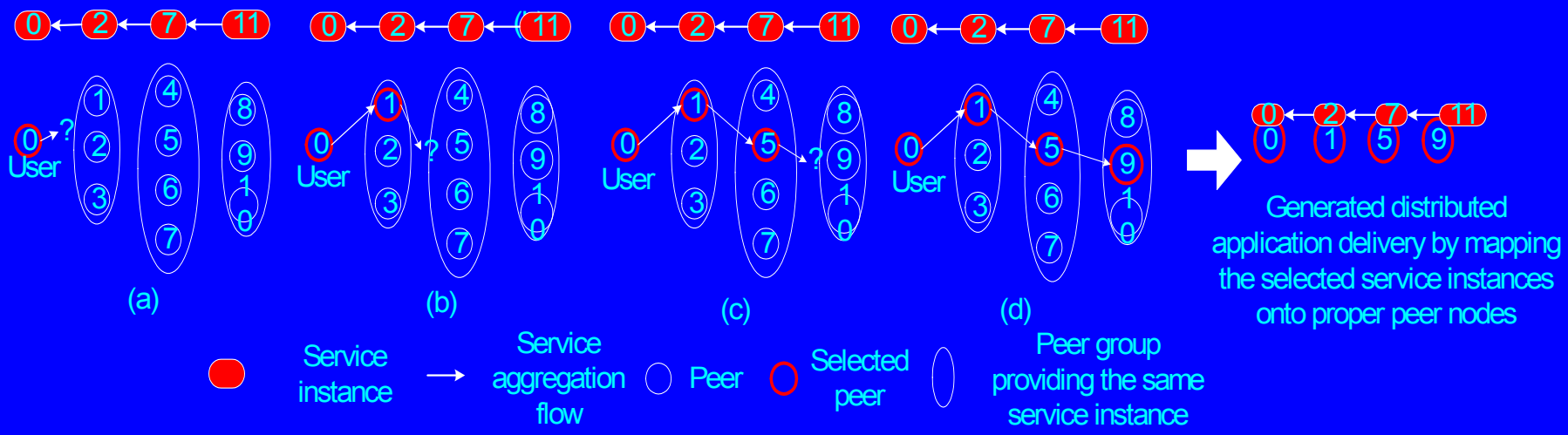
- However, the Optimal Service Distribution problem is NP-hard…
- Polynomial heuristic algorithm for the OSD problem

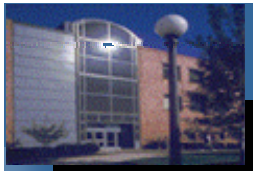
```
Insert those special service components into proper devices;
While  there exists uninserted service components Do
    sort the available devices in decreasing order of resource availability;
    pick the head of the device list , device D;
     if D contains a node A
    then insert A's largest neighbor into D and merge it with A;
    else insert the service component which has the largest resource
          requirements into A;
```



Dynamic Peer Selection Tier

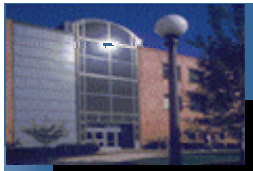
- Distributed and hop-by-hop peer selection





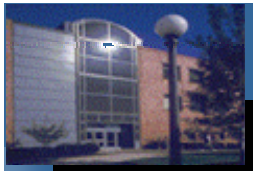
Dynamic Peer Selection Tier

- An integrated and configurable metric to select a peer by comprehensively considering
  - candidate peer's uptime >= app's duration
  - candidate peer's resources >= service instance's requirements
  - if multiple peers qualify, choose the peer with the most abundant resources
  - the importance of different resource types can be configured according to the app's semantics or user preference.
- Dynamic neighbor resolution
  - neighbor list at each peer is maintained as soft states
  - depend on results of the service composer



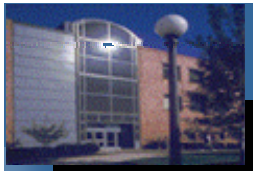
Performance Evaluation

- Using both prototype implementation and simulations
- Experimental results for ubiquitous environments show that our service configuration model achieves the following advantages:
  - flexibly accommodate common runtime changes;
  - handle complex service graph, not just service path;
  - the overhead of dynamic service configuration is small;
  - the heuristic service distribution algorithm is efficient compared to the NP-hard optimal solution;
  - the heuristic service distribution algorithm achieves best user request success rate by aggregating resources efficiently.

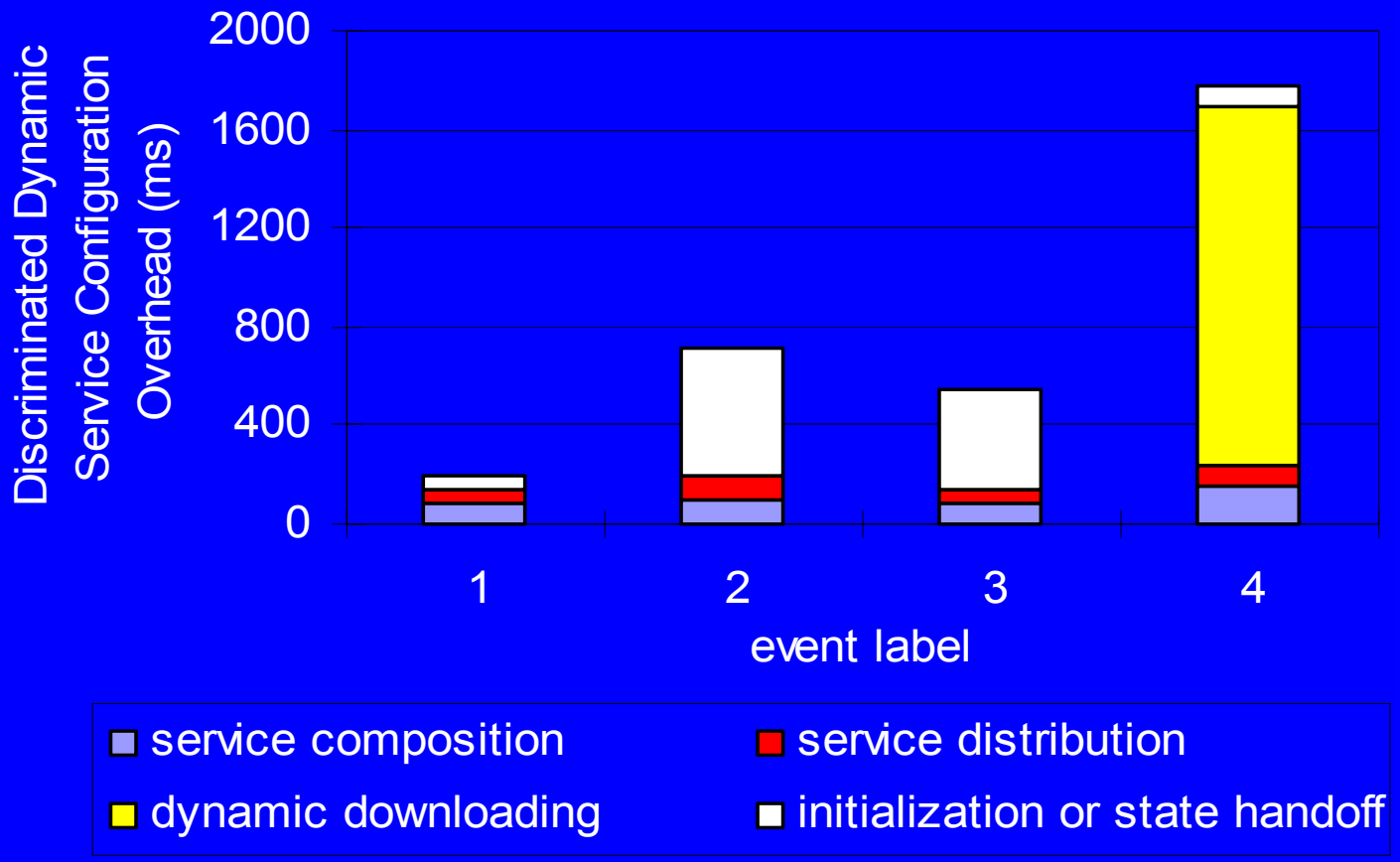


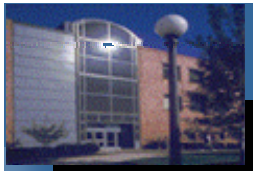
Experimental Results

Event Label	Event Content	Service Configuration Results	Measured QoS
1	Start "*mobile audio-on-demand*" on the desktop1. User QoS request: CD quality music	Desktop 1 Audio Server → Desktop 2 Audio Player	40fps
2	Switch from desktop to PDA with a wireless link. Music continues from the interruption point.	Desktop 1 Audio Server → Desktop 2 MPEG2wav Transcoder → Jornada Audio Player	40fps
3	Switch back from PDA to another desktop3	Desktop 1 Audio Server → Desktop 3 Audio Player	40fps
4	Start *video conferencing* on the workstations. User QoS request: video(25fps), audio(6fps)	Work Station 1 video recorder, audio recorder → WorkStation 2 gateway → WorkStation3 lipsync → video player, Audio player	25fps, 6fps



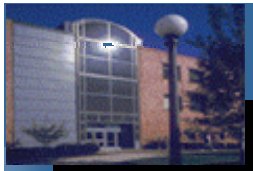
Experimental Results



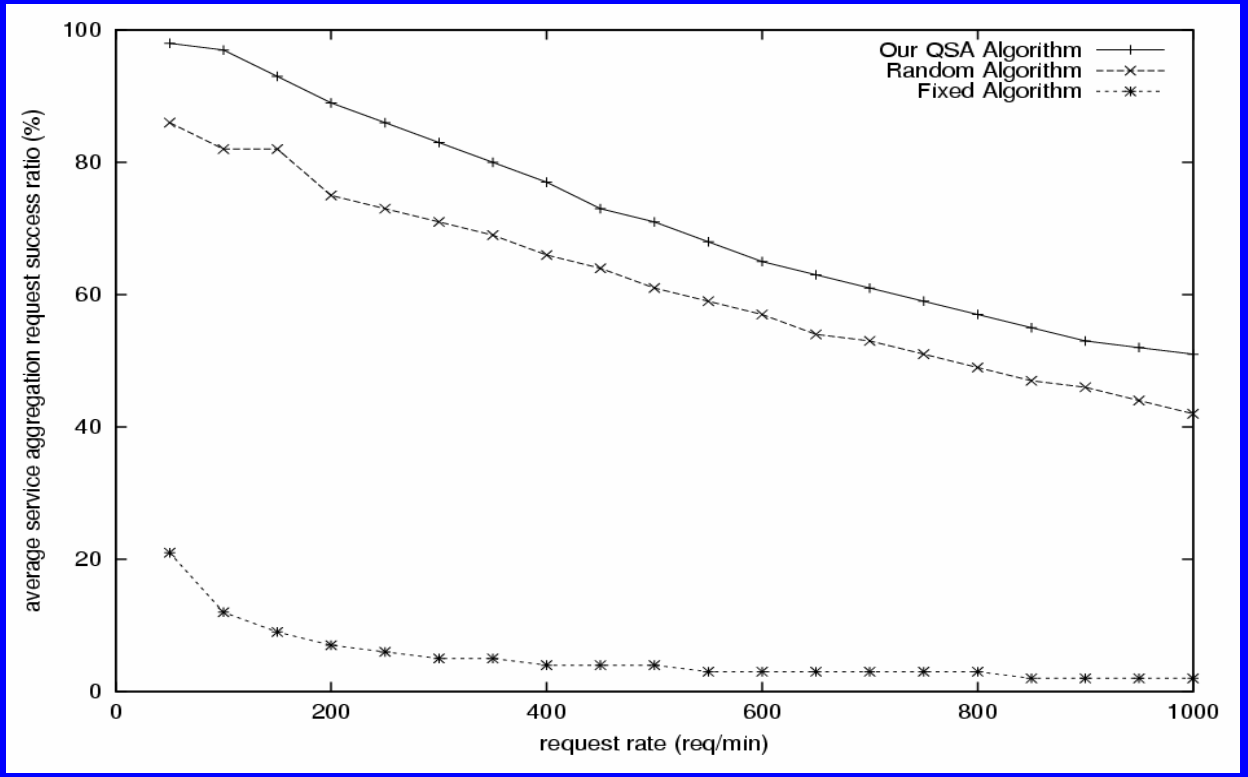


Evaluation Methodology for P2P

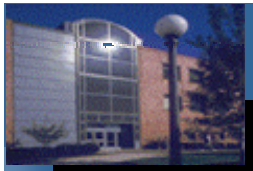
- simulate a large-scale P2P system with 10000 peers.
- system parameters such as network bandwidth between peers are randomly set
- the QoS-aware service composition (QSA) algorithms are locally executed on each peer
- probing overhead is controlled within 1%
- evaluation metric: service composition success ratio
- random and fixed (client-server) algorithms as comparison
- shown experiment: without topological variation of P2P systems (further experiments in papers)



Results and Analysis

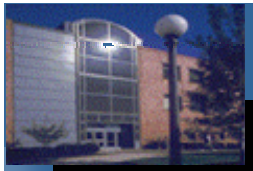


Average service composition success ratio under different service composition request rates without topological variation.



Conclusions

- Identifying two key problems: "*service composition*" and "*service distribution*"
- Introducing an integrated model to solve the above problems in a unified framework
- Providing flexible service composition solution with QoS consistency check
- Defining the optimal service distribution problem which is shown to be NP-Hard in different environments and then providing polynomial approximation algorithm



Acknowledgements

- This research is supported by the NASA grant and National Science Foundation (NSF) grant.

- Questions and Discussions …