Adaptive Resource Management in Peer-to-Peer Middleware

Thomas Repantis, Yannis Drougas, Vana Kalogeraki

Department of Computer Science and Engineering

University of California, Riverside

{trep,drougas,vana}@cs.ucr.edu

http://www.cs.ucr.edu/~{trep,drougas,vana}

Outline

1. Introduction
2. System Architecture
   - Application Service Graph
   - Resource Graph
   - Fair Task Allocation Algorithm
3. System Operation
4. Performance Evaluation
5. Conclusion

Distributed Real-Time Applications

- Require a sequence of invocations of objects/services across different nodes of a distributed system.
- Have multiple end-to-end QoS requirements:
  - Bounded latency and jitter
  - Guaranteed minimum throughput
  - Reliability and fault-tolerance

Motivation

Challenges in managing large-scale distributed real-time systems:

- No central manager can have an accurate global view of the system.
- The resources are heterogeneous and shared among many applications.
- The applications have unpredictable arrival times.
- Multiple QoS requirements need to be satisfied simultaneously and traded-off.

Peer-to-Peer Middleware

Peer-to-peer (P2P) systems:

- Enable the sharing of computer resources in large-scale distributed environments.
- Consist of nodes acting as both clients and servers.
- Do not require a central coordinator (scalable).
- Are resilient to node failures (fault-tolerant).
- Are self-organizing (adapt to different loads).
- Have been used successfully for:
  - Distributed Computing (Seti@, Folding@)
  - File Sharing (Gnutella, Oceanstore)
  - Online Gaming (Playstation)
  - Spam Detection (SpamNet)

QoS for P2P Applications

Providing QoS guarantees for applications running over P2P remains a challenge.

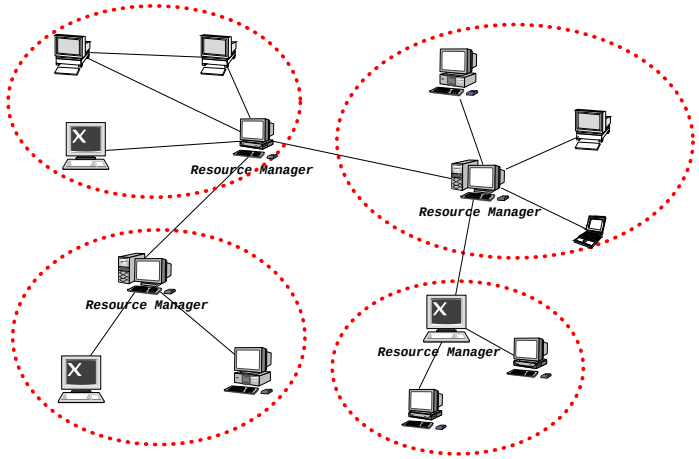
- Peers are heterogeneous in:
  - processor capabilities
  - network bandwidth
- The environment is dynamic:
  - Peers join, leave, fail without a priori notification.
  - Tasks arrive independently and have different QoS demands.

Contributions

The proposed peer-to-peer adaptive resource management system relies on:

- A resource graph that facilitates decentralized scheduling decisions.
- Application service graphs that capture the current execution status.
- A dynamic and adaptive task allocation and load balancing algorithm based on the notion of fairness.

System Architecture



- Peers build an overlay over the actual network.
- Peers are grouped in domains, according to their topological proximity.
- The Resource Managers are selected among regular peers to act as leaders of the domain.
- Each Resource Manager is connected to all the Processors of its domain directly.

Resource Manager's Tasks

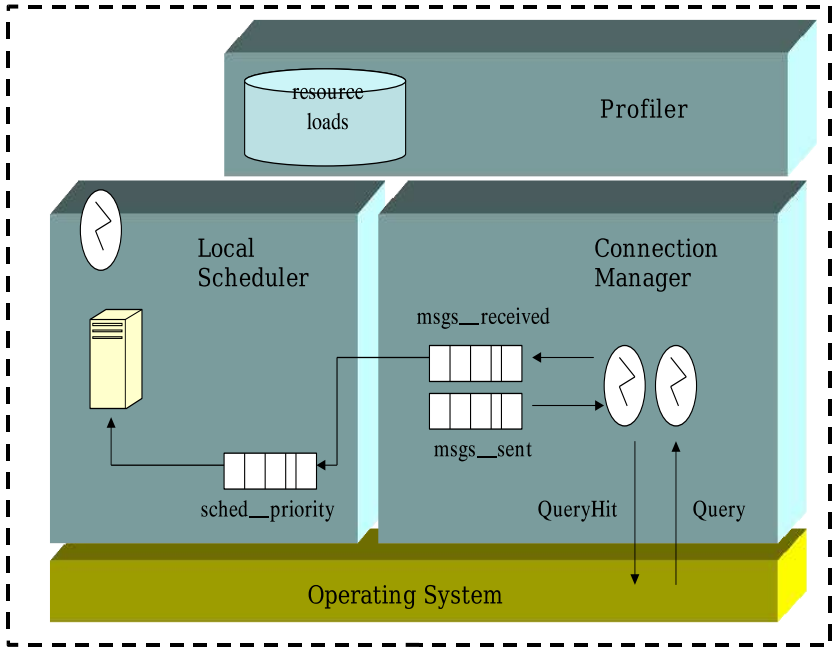
The Resource Manager:

- Has a global view of the domain in terms of the applications in the domain and the utilization of the system resources.
- Distributes the application objects on the processors to meet the application QoS requirements.
- Collects load information of all the peers of the domain, used to allocate objects to the less loaded peers in the domain.

Processor's Components

Each of the Processors in the domain consists of the following main components:

- Connection Manager
- Profiler of resource usage and application behavior.
- Scheduler based on Least Laxity Scheduling (LLS).



Information Base

Our system maintains current information at three different levels:

1. On a node-level, nodes maintain local resource loads and local application execution and communication times.
2. On a domain-level, Resource Managers maintain current resource utilization about all the nodes in their domain and end-to-end application information, collected at run-time.
3. On a system-level, Resource Managers maintain summarized information about the other domains, updated lazily using a gossiping protocol.

Intra-Domain Information

Every Resource Manager maintains the following information for its domain:

1. The identifiers of the processors P_i in the domain.
2. The current processor load l_i for all processors i, expressed as the product of processing power with current utilization.
3. The currently used network bandwidth bw_i for all processors i.
4. The application objects O_{ij} and services S_{ij} located on every processor i.

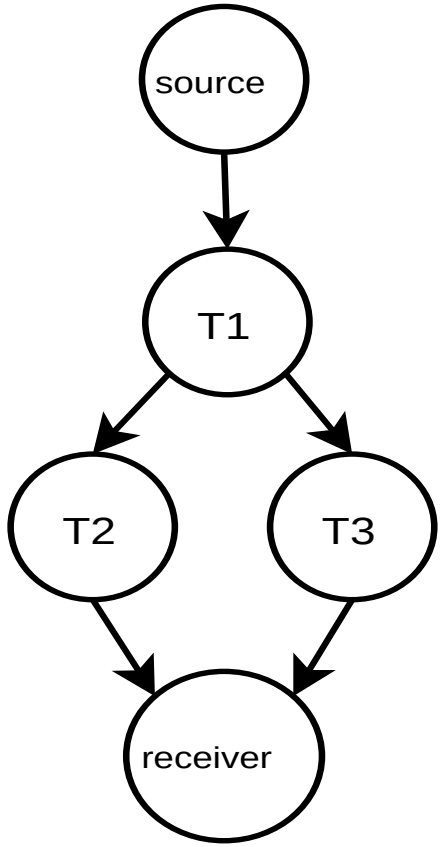
Inter-Domain Information

Every Resource Manager maintains summarized information for other domains, propagated to it by its Resource Manager peers:

1. A list of other domains D_k in the system, along with the identifiers of the corresponding Resource Managers RM_k.
2. A summary of the available application objects $SumO_k$ and the available services $SumS_k$ in each domain. The summaries can be obtained using Bloom Filters.

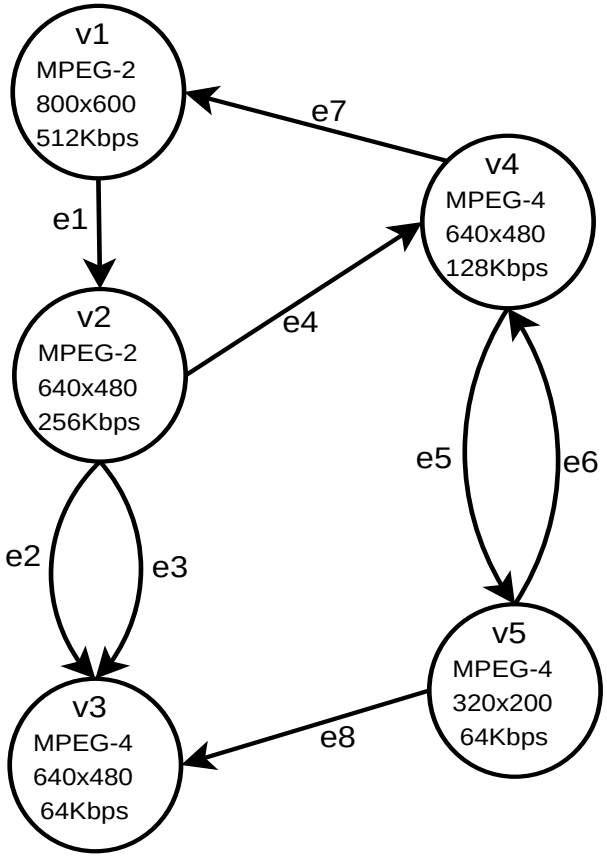
Application Service Graph

- We model an application task as a sequence of invocations of objects and services distributed across multiple processors.
- The application service graph G_s represents the objects and services invoked by the application tasks and the corresponding resource requirements.



Resource Graph

- The Resource Manager maintains a resource graph G_r for its domain.
- The aim of the resource graph G_r is to identify the services available at the peers and assign those services to the requesting applications.
- The resulting assignment should meet the application deadline and fulfi ll its resource requirements.



Resource Manager's Goals

Each Resource Manager is responsible for:

- Composing the application service graph G_s, so that the QoS requirements of the application will be met.
- Ensuring that the load among the peers of its domain is fairly balanced.

In order to evaluate how fair a particular load distribution is, the Resource Manager employs the Fairness Index metric [Jain et al., 1984].

Fairness Index Definition

The Fairness Index of the load distribution $\bar{l}$ among the set of peers P_D of a domain D is defined as:

$$\mathcal{F}(\bar{l}_{P_D}) = \frac{(\sum_{p \in P_D} l_p)^2}{|P_D| \cdot \sum_{p \in P_D} l_p^2}$$

- The fairer the system is, the higher the index.
- Bounded: The fairness of a totally fair system is 1, while that of a totally unfair system is 0.
- Independent of size or scale of distribution.
- Reasonably modified when the load of a single node changes.

Fairness Index Example

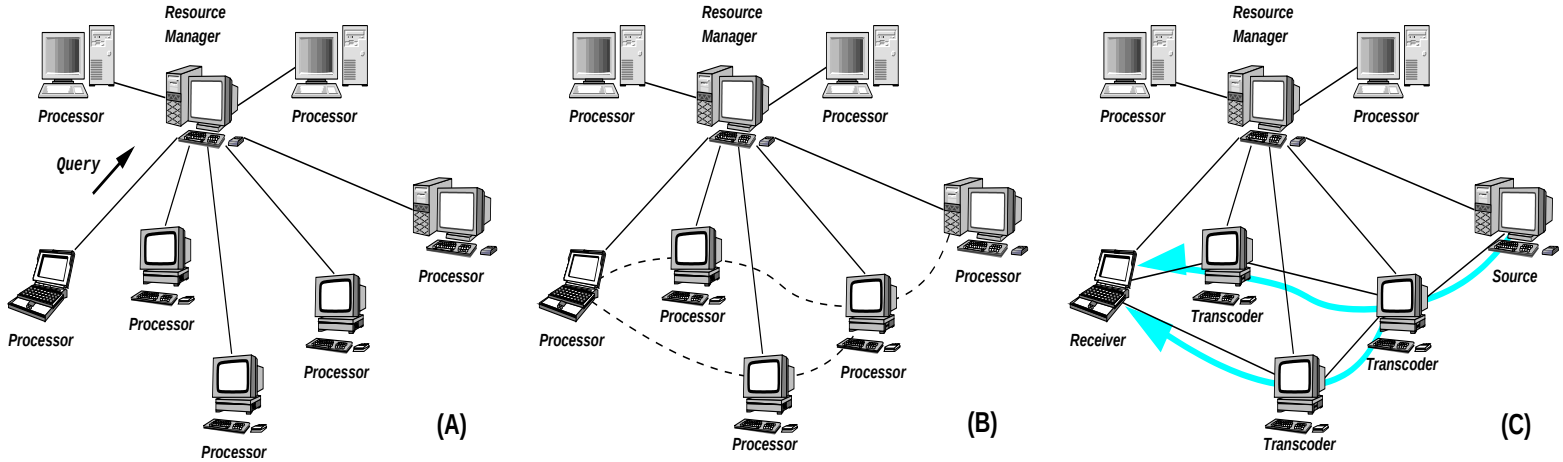
Assuming the utilization of each of 4 nodes is 80%. The fairness index is:

$$\mathcal{F}(\bar{l}) = \frac{(0.8+0.8+0.8+0.8)^2}{4\cdot(0.64+0.64+0.64+0.64)} = \frac{3.2^2}{4\cdot 2.56} = \frac{10.24}{10.24} = 1$$

When the utilization of a single node is 100%, while for the rest of the nodes it is 20%, then:

$$\mathcal{F}(\bar{l}) = \frac{(0.2+0.2+0.2+1)^2}{4\cdot(0.04+0.04+0.04+1)} = \frac{1.6^2}{4\cdot 1.12} \approx 0.57$$

System Operation



- A user submits a query to the Resource Manager.
- The Resource Manager uses the task allocation algorithm to assign the tasks to peers.
- Application execution begins.

Task Allocation Algorithm

The Resource Manager uses the resource graph to search for configurations in the domain that satisfy a requested service. Specifically:

- It uses the Breadth-First-Search (BFS) algorithm to search for services (edges) connecting the initial and fi nal requested application states (vertices)
- It calculates which paths satisfy the deadlines by utilizing the current load information.
- It prunes the possible solutions using the requested QoS requirements.
- Among the allocations that satisfy the QoS requirements, it chooses the one that results to the maximum fairness of the load distribution among the peers.

Resource Utilization Feedback

- *Intra-Domain Propagation*
  - Every peer monitors current processor and network load, computation and communication times of the executing applications.
  - It periodically propagates load information to the Resource Manager of the domain.
- *Inter-Domain Propagation*
  - Resource Managers gossip summarized information of the available objects and services available in their domains.
  - The summaries have to be updated only when peers join or leave the system or new services become available.

Adaptive Resource Management

During a service session a variety of global and local factors may affect the performance of the system:

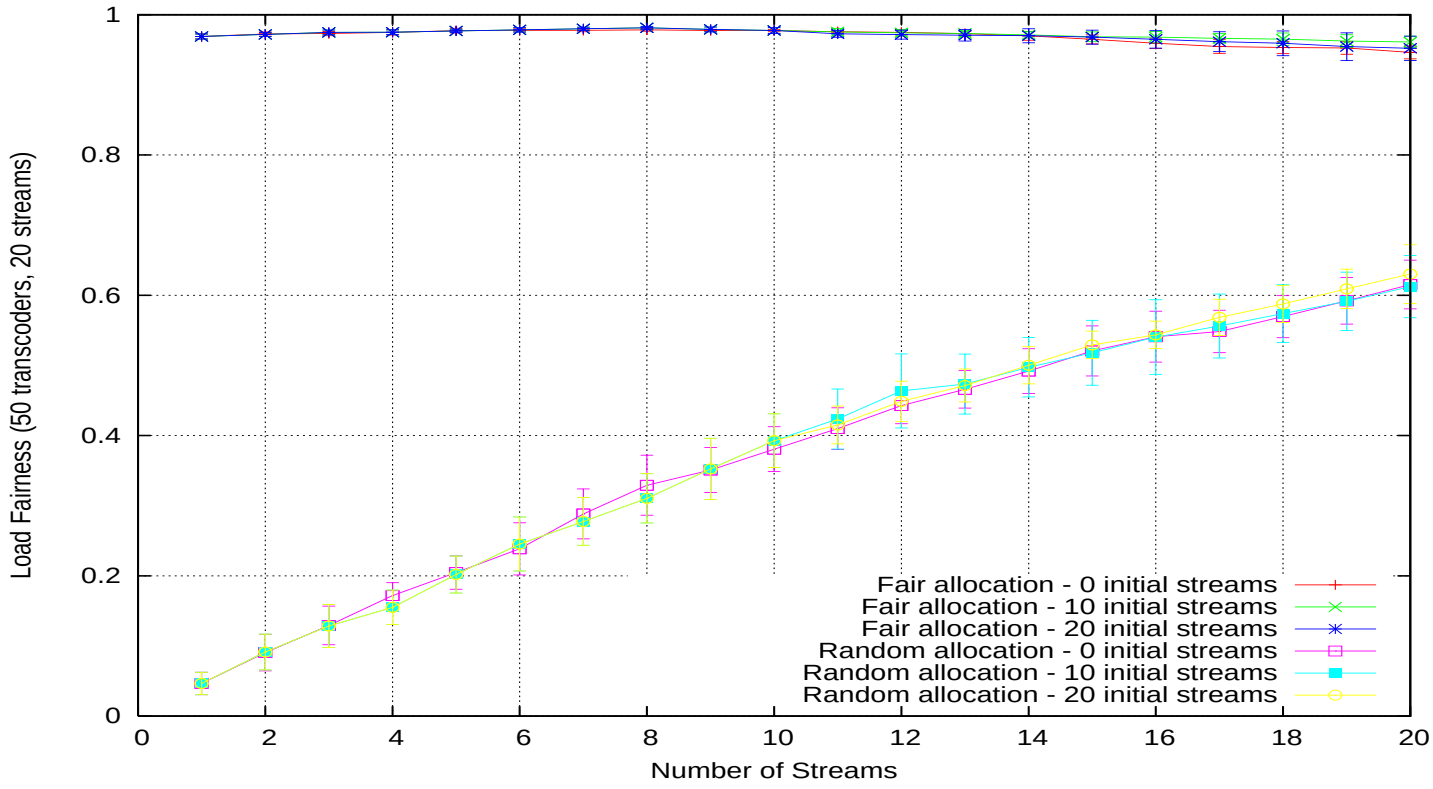
- Changes in the infrastructure.
- Changes in the number of service sessions.
- Changes in the local resource conditions.
- Changes in the user requirements.

When the Resource Manager detects that peers are overloaded, it can reassign tasks or redirect them to other domains.

Preliminary Performance Evaluation

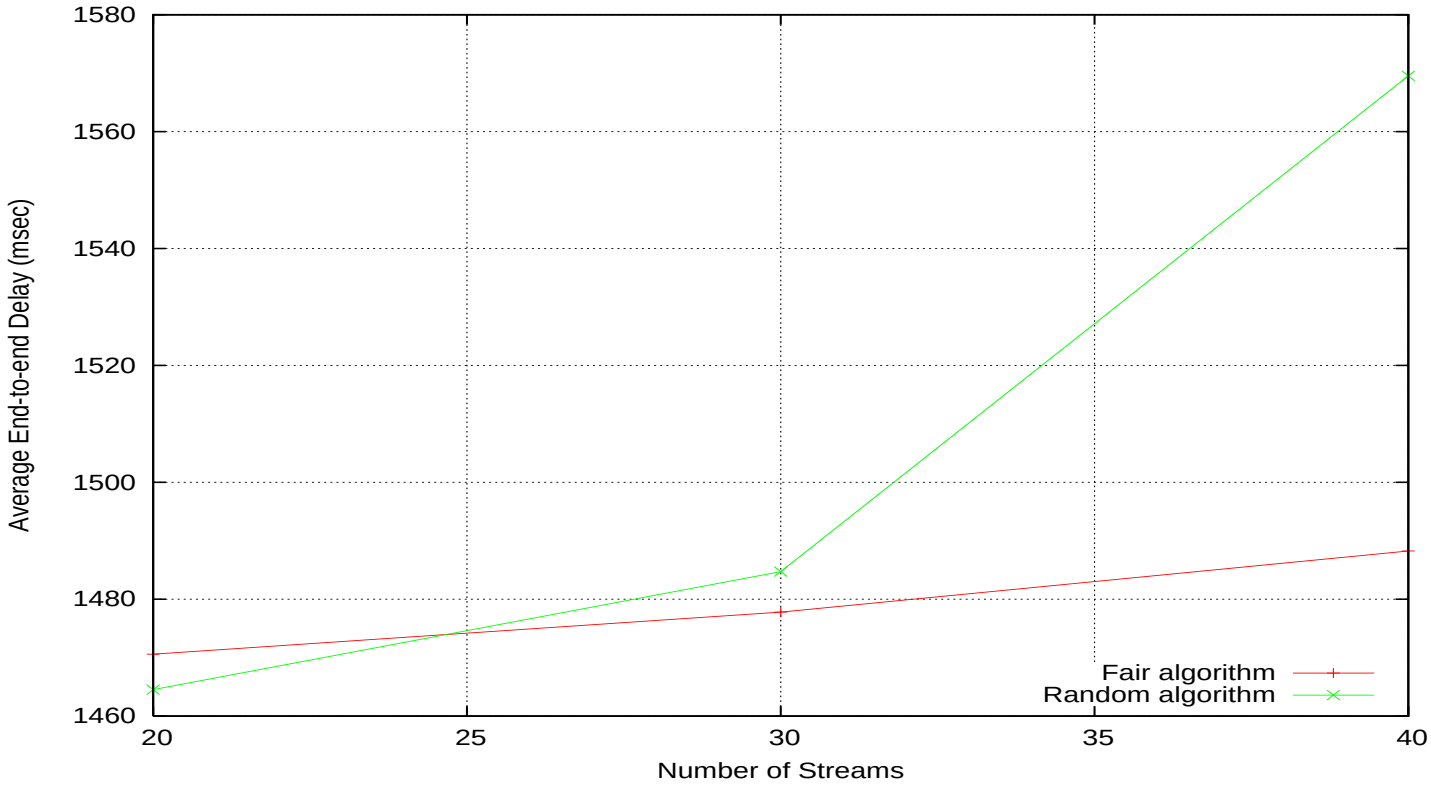
- We used peer-to-peer media streaming and transcoding as a distributed real-time application example.
- We implemented a simulator that utilizes an underlying physical topology of 1476 routers generated with GT-ITM.
- We simulated cross traffic by using random probabilities.
- 1000 nodes in total, 400 sources, 50 transcoders, 40 streams.
- We compared the fair task allocation scheme against a random task allocation.

Fairness



The fair task allocation algorithm constantly balances the load.

Average End-to-End Delay



The fair task allocation algorithm guarantees bounded latency.

Conclusions and Future Work

- We have described an adaptive resource management architecture for large-scale, real-time, peer-to-peer middleware.
- The system utilizes resource managers that maintain resource and service graphs, and employs a fair task allocation algorithm.

Future Work:

- Investigate the performance of our system in more detail.
- Compare our architecture to a totally decentralized approach.

Thank you!

Questions / Comments?

Adaptive Resource Management in Peer-to-Peer Middleware

Thomas Repantis, Yannis Drougas, Vana Kalogeraki

Department of Computer Science and Engineering

University of California, Riverside

{trep,drougas,vana}@cs.ucr.edu

http://www.cs.ucr.edu/~{trep,drougas,vana}

Application Tasks Information

The Resource Manager keeps the following information for each application task t:

- $Deadline_t$: the time interval, starting at task initiation within which the task should complete, specified by the end user.
- $Importance_t$: a metric that represents the relative importance of the application, specified by the end user.
- $Execution\_time_t$: the estimated amount of time from initiation to completion of application task t.

Overlay Construction

- The network is unstructured and self-organizing. The maximum number of processing peers a Resource Manager can manage depends on its own capabilities.
- The requirements for becoming a Resource Manager are:
  - Sufficient bandwidth
  - Sufficient processing power
  - Sufficient uptime

Task Allocation Example