

Distributed Pricing in P2P Networks

David Hausheer¹

*¹Swiss Federal Institute of Technology, ETH Zürich
Computer Engineering and Networks Laboratory TIK, Zürich, Switzerland
E-Mail: hausheer@tik.ee.ethz.ch*



*Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology, ETH Zürich*



Outline

- Motivation for Pricing in P2P
- Basic P2P Pricing Protocol
 - Price Determination
 - Price Valuation
 - Price Dissemination
- Pricing Design
 - Pricing APIs
- Conclusions



Motivation for Pricing in P2P

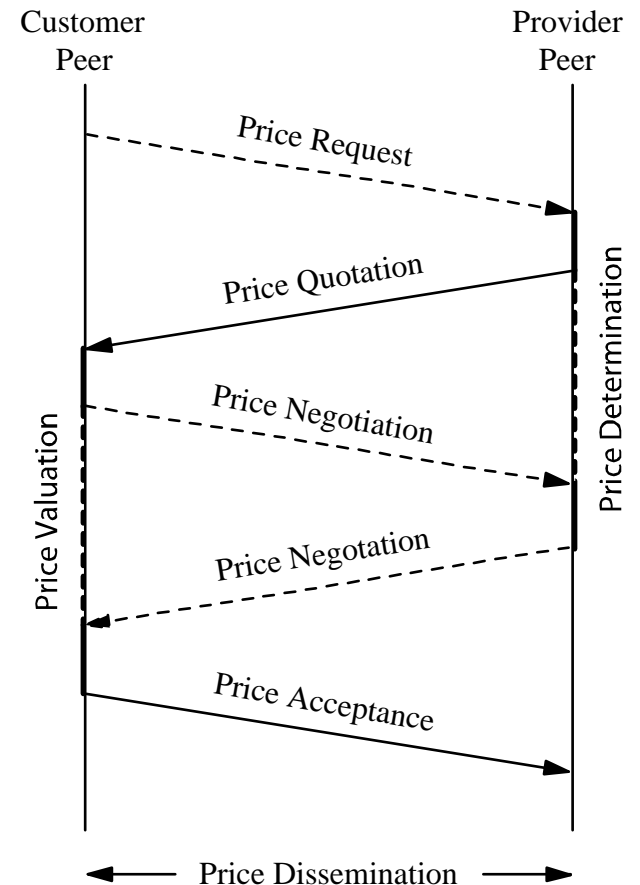
- Observations in current P2P file sharing systems
 - There is a lot of “crap” content being offered by users
 - Peers are not peers (heterogeneous infrastructure)
- Accounting “counts” all bits the **same**
 - There is **no differentiation** between good bits and bad bits
 - It is **scenario-specific** (how many files are 10 min connection worth?)
 - Users have **no control** (I want to provide my files for free, but I can't)
- Pricing enables **differentiation, generalization** and **control**
 - I give you 10 Britney Spears songs for 1 Madonna song
 - I can't offer files, but I have lots of computing power
 - If my PC is busy, I want you to pay the double
- But this is user-specific!
 - That's why we intend the users to determine the price



Basic P2P Pricing Protocol

□ Pricing comprises of three important mechanisms:

- Price **determination** => How a provider determines a price
- Price **valuation** => How a customer values a price
- Price **dissemination** => How prices are communicated, negotiated and accepted between providers and customers



Price Determination

- Pricing is **flexible** but offers a set of **built-in strategies** to determine a price
- **Additional strategies** can be created enabling any business model
- Built-in mechanisms:
 - Pricing based on **price quotes** ('market-based pricing')
 - Determines a price based on price quotes from other providers
 - Pricing based on **current demand**
 - Calculates the price for a service based on the current load
 - Pricing based on **user ratings**
 - Uses rating information from other users to determine the price
 - Fixed pricing configured by **rules**
 - Fixed or range-limited set of price coefficients to be used
- A user can enable, disable and combine individual pricing strategies, or create additional strategies, configured by **local rules**
- Final prices might be **negotiated** between customer and provider
 - Support of complex **auction models**



Price Valuation

- Pricing should reveal the **value** or **utility** of a good
 - A rational user buys the good, if a price is lower than **expected** utility
 - **Rating** is needed to know what can be expected
 - Rating is “pricing” done by the consumer
- Pricing (and rating) need to be done in a **distributed** way
 - Only a user can value, e.g., the utility of a service
 - A user can **change anything** locally in his code anyway
 - **Rules** can limit the prices, but enforcement is needed (exclusion?)



Price Valuation (cont'd)

- The underlying mechanisms are similar to **price determination**
 - A customer might actually use the **same strategies** and **information** to value the price for an offered service
 - A provider **can** use information from other users (he knows the good)
 - A customer **has** to rely on other information (unless he fully trusts the provider)
- Ratings need to be **aggregated** appropriately
 - Only scalable if done in a **distributed** way (=> Eigenrep paper)
 - Ratings need to be **weighted** according to user-interests, trust-levels...
- What is the **incentive** for a user to provide ratings?
 - Users might get paid for ratings



Price Dissemination

- How prices get from providers to customers
 - Customer **asks** provider for price (pull method)
 - Provider regularly **advertises** its prices (push method)
 - Customers might **subscribe** for price advertisements
- More scalable approaches:
 - Prices are stored in a **distributed hash table**
 - Price messages are **cached** at intermediate peers
 - How long should a price remain valid?
 - What about self-interested peers?
 - Requests are **routed** according to price information
 - Two processes at once (search & valuation)
- **Negotiation** enables further pricing approaches
 - Auction models might be supported

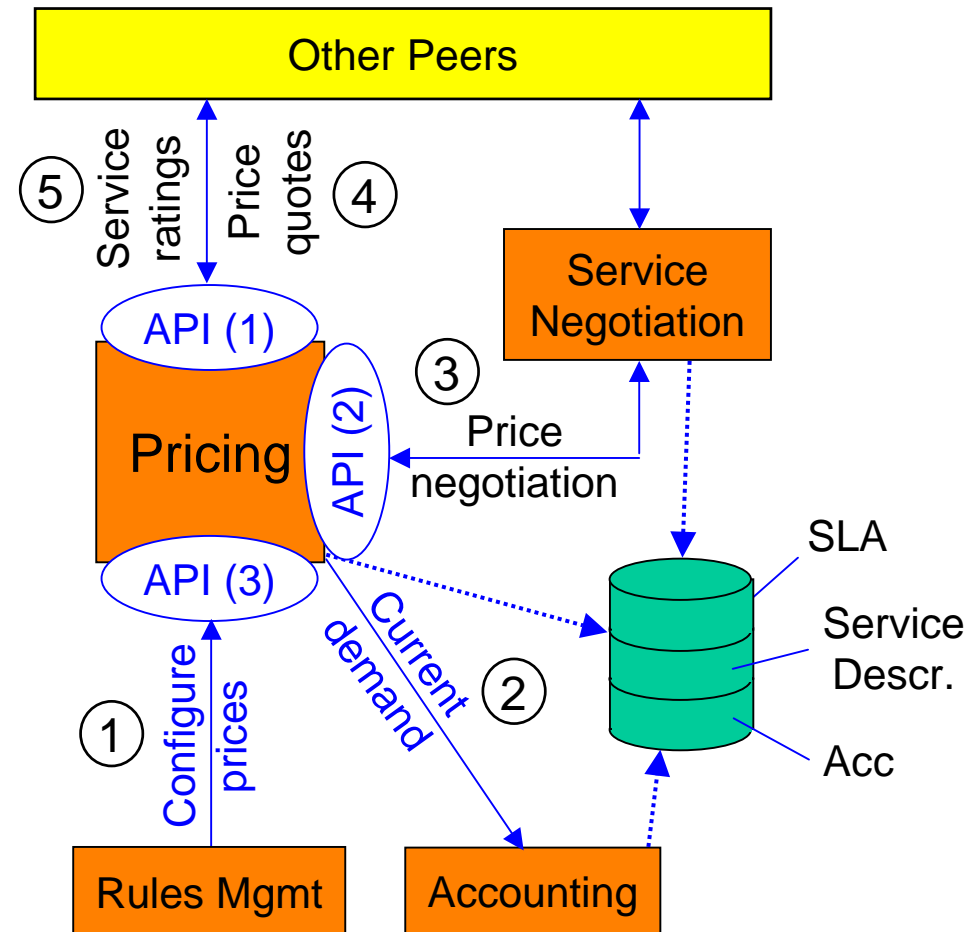


Pricing Design

Pricing API:

- Support of
 1. Fixed pricing by rules
 2. Pricing based on demand
 3. Pricing based on negotiation
 4. Pricing based on price quotes ('market prices')
 5. Pricing based on user ratings

- Methods for pulling, posting and calculate a tariff, service rating, and approval



Pricing API (1)

- Module internal methods, provided to **pricing module instances** on other peers
 - public void **getServiceTariff** (ServiceId id, boolean getUpdates, PricingListener pl);
 - public void **postServiceTariff** (PeerId pid, ServiceId sid, ServiceTariff t);
 - public void **getServiceRating** (ServiceId, boolean getUpdates, PricingListener pl);
 - public void **postServiceRating** (PeerId pid, ServiceId sid, ServiceRating r);
- Default is **pull method**
- getUpdates=true adds a peer on a list of recipients (to trigger the more efficient **push method**)



Pricing API (2)

- Methods provided to the **service negotiation** module
 - public void **requestTariff** (SLA proposedSLA, PricingListener pl);
 - public void **approveServiceTariff** (ServiceTariff t, PricingListener pl);
- **requestTariff** calculates and returns a tariff for a proposed SLA. Service Negotiation stores the agreed tariff in the SLA, where Charging can look it up.
- **approveServiceTariff** valuates a proposed service tariff in the negotiation process. Pricing approves ServiceTariff, offers new ServiceTariff or stops negotiation.



Pricing API (3)

- Methods provided to the **rules management** module
 - public void **configurePricing** (ServicePricingPolicy p);
- **configurePricing** enables the rules management module to configure pricing.
 - Upload a new method that calculates service tariffs
 - Configure pricing to use a specific tariff method to calculate the tariff for a service
 - Might be a built-in method or a new method uploaded by the rules management module
 - Additionally, a pricing policy can limit the range of allowed price coefficients in a tariff.



Conclusions

- Pricing is necessary to enable **service differentiation** in P2P
- The proposed pricing design supports a set of **built-in pricing strategies** and enables additional strategies to be created by tariff developers
- Price determination is done by the individual peers to support the **user-specific** aspects of a tariff
- Pricing works with different accounting patterns
 - Pricing “**weights**” the units accounted for with any accounting pattern
 - In token-based accounting pricing determines the amount of tokens to be spent per file
 - KARMA uses pricing together with an **account holding** technique

