Load Shedding on Data Streams

Nesime Tatbul, Uğur Çetintemel, Stan Zdonik @ Brown University

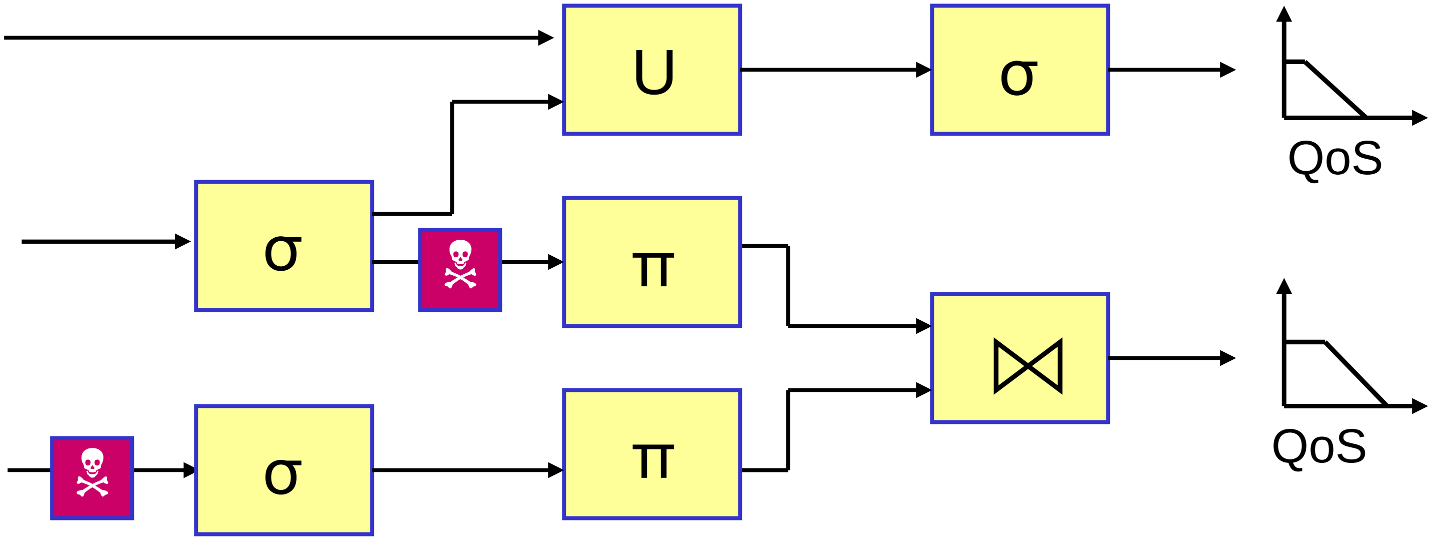
Mitch Cherniack @ Brandeis University

Michael Stonebraker @ M.I.T.

Handling Overload with Load Shedding

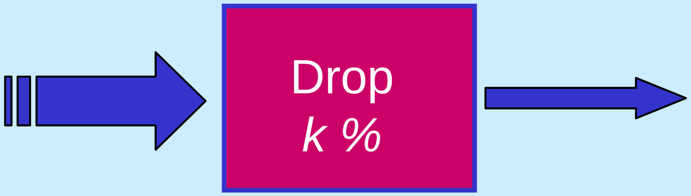
- real-time data pushed from financial data feeds, sensors, and alike
- high and unpredictable data rates
- resource overload => growing queues and late results
- solution: “load shedding”
  - eliminate excess load by dropping data

Load Shedding by Inserting Drops

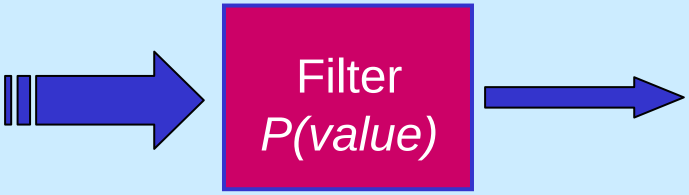


two types of drops:

Random Drop

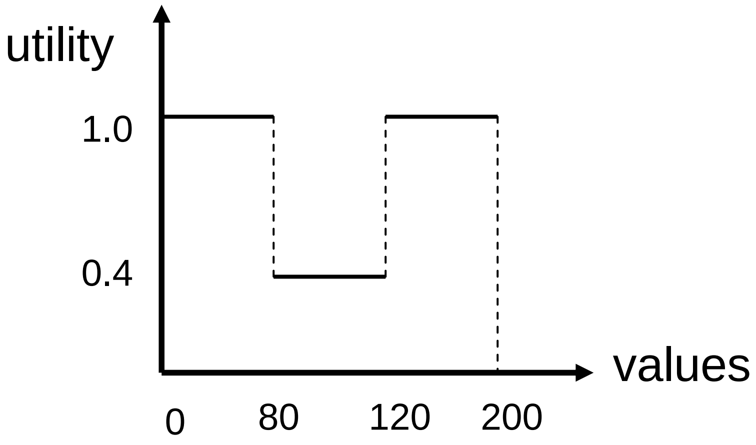


Semantic Drop

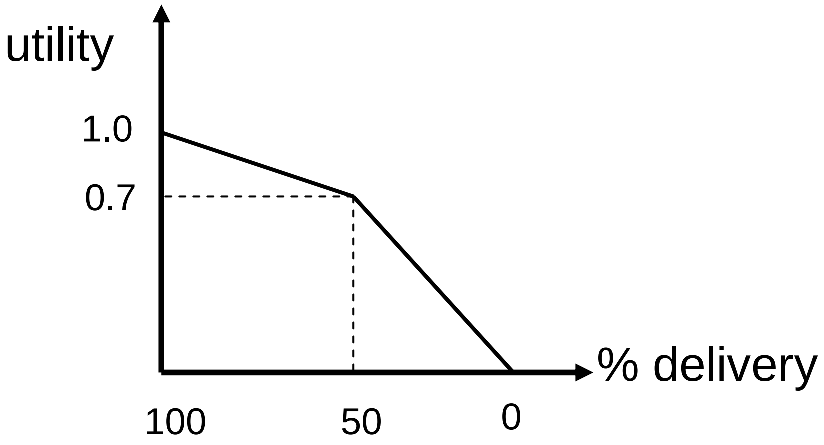


Quality of Service

- Value-based QoS



- Loss-tolerance QoS



- Latency-based QoS is handled by scheduler.

Problem Statement

- N: query network
- I : set of input streams
- C: processing capacity

when $Load(N(I)) > C$, transform N to N' such that

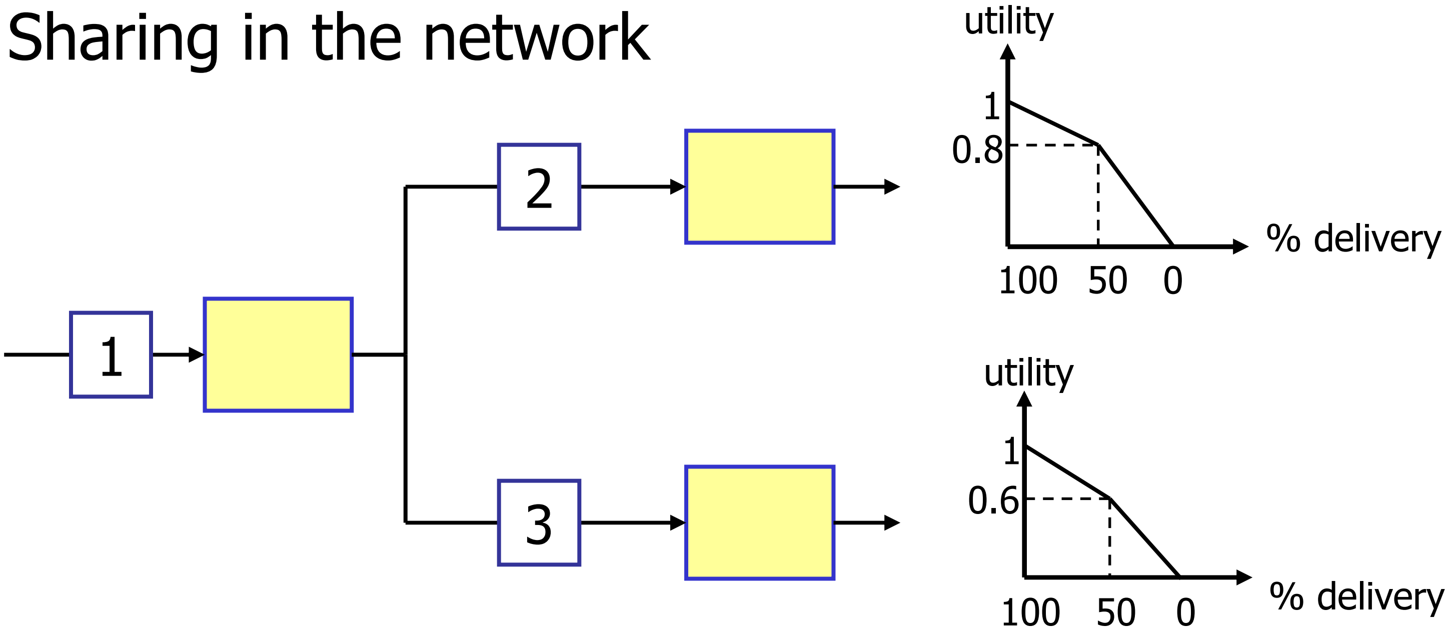
- $Load(N'(I)) < C$
- $Utility(N(I))-Utility(N'(I))$ is minimized

Key Questions

- when to shed load?
- **where to shed load?**
- how much load to shed?
- which tuples to drop?

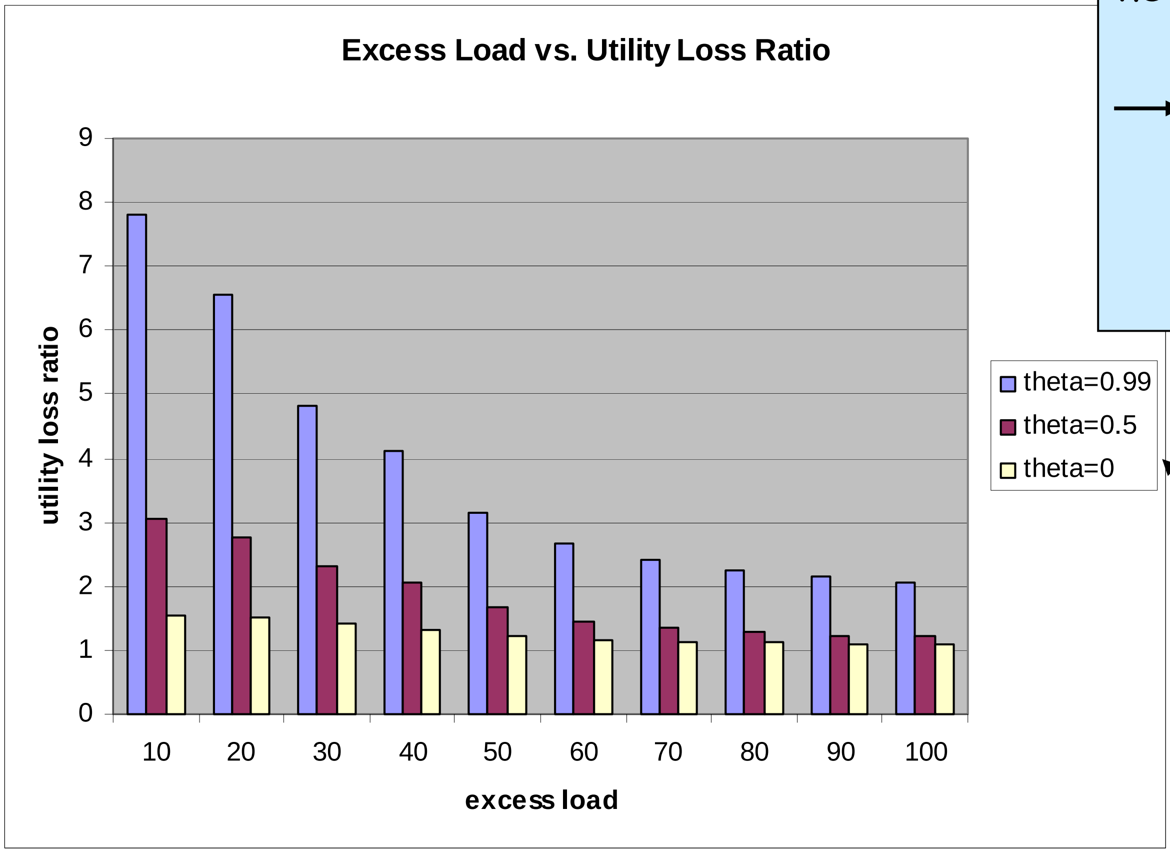
Where to shed load?

- Sharing in the network

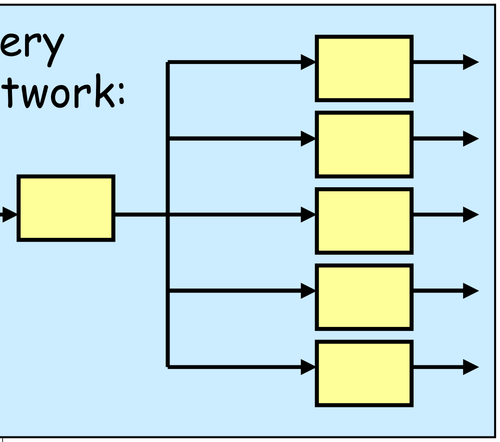


- Maximize load gain and minimize utility loss
- Dropping at inputs is not always the best

Load Shedding vs. Admission Control



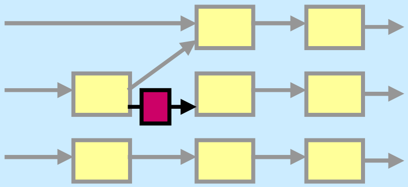
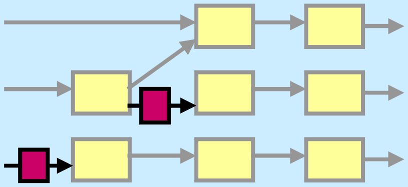
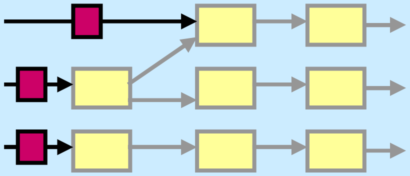
query network:



shows how skewed the utilities are

Materialized Load Shedding Plans

cursor

Excess Load	Drop Insertion Plan
10%	
20%	
⋮	⋮
300%	

shed less!

shed more!

Ongoing and Future Work

- Handling complex operators
  - Joins for the general case
  - Aggregates
- Other resource limitations
  - Memory - windowed operators
  - Bandwidth - Aurora*
  - Power - at sensor level
- Other techniques
  - adjusting window size
  - inserting aggregates

More Information

- come and see Aurora demo @ Sigmod’03
- paper to appear @ VLDB’03
- visit: *http://www.cs.brown.edu/research/aurora*
- email: *tatbul@cs.brown.edu*