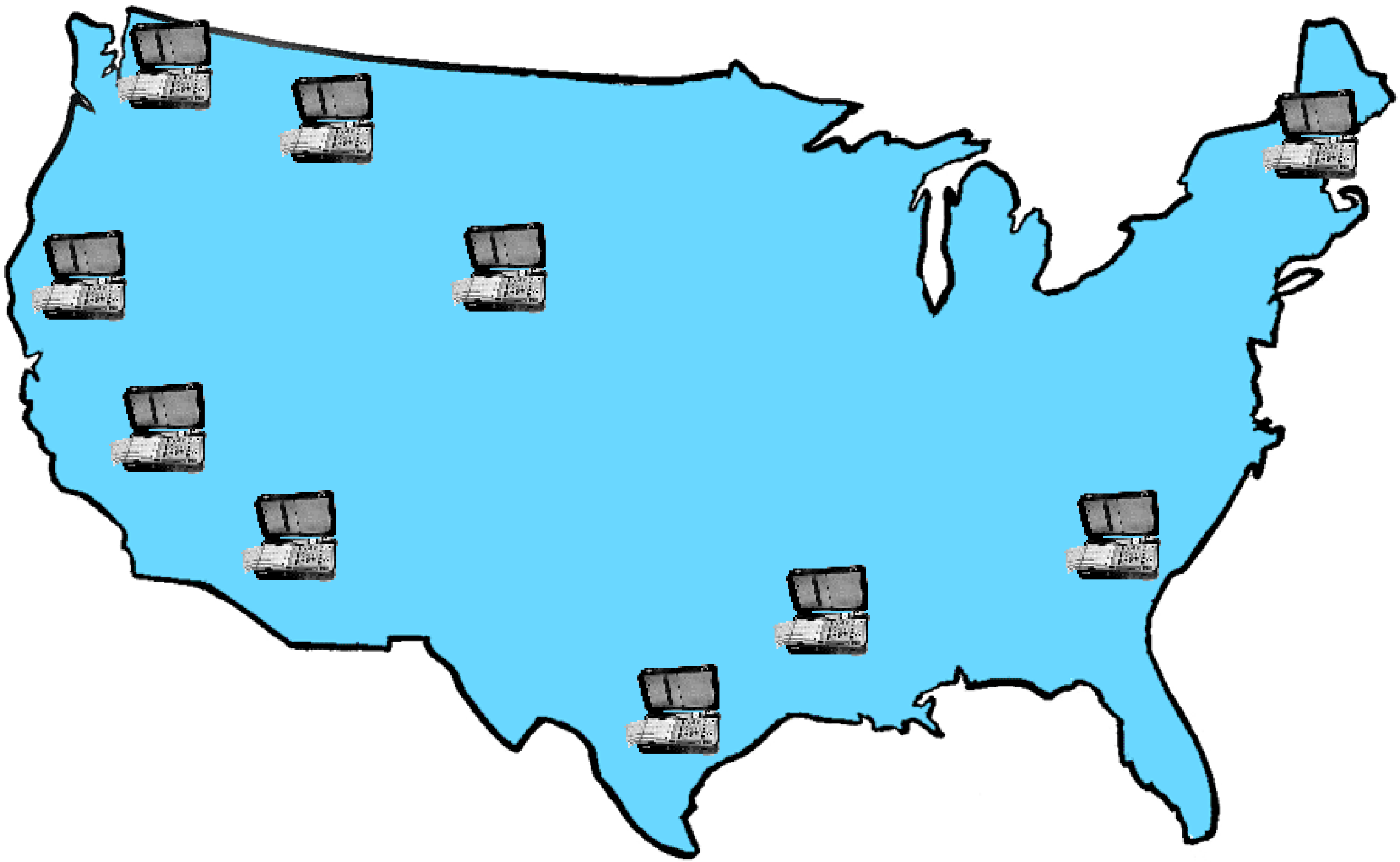
A Cost-Space Approach to Distributed Query Optimization in Stream Based Overlays

Jeffrey Shneidman, **Peter Pietzuch**, Matt Welsh, Margo Seltzer, Mema Roussopoulos

Systems Research Group – Harvard University
Division of Engineering and Applied Sciences

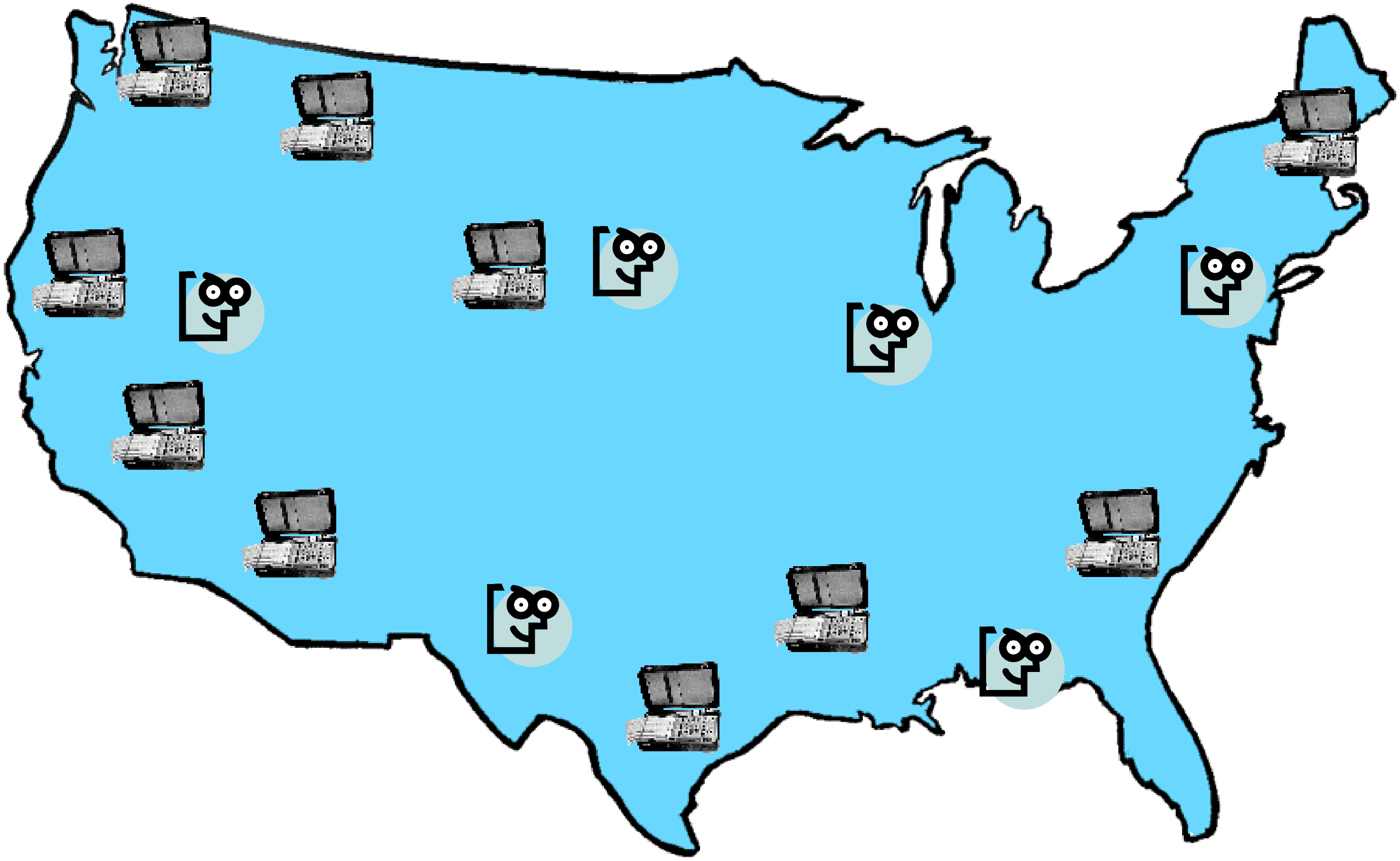
NetDB - April 2005

Data Stream Applications



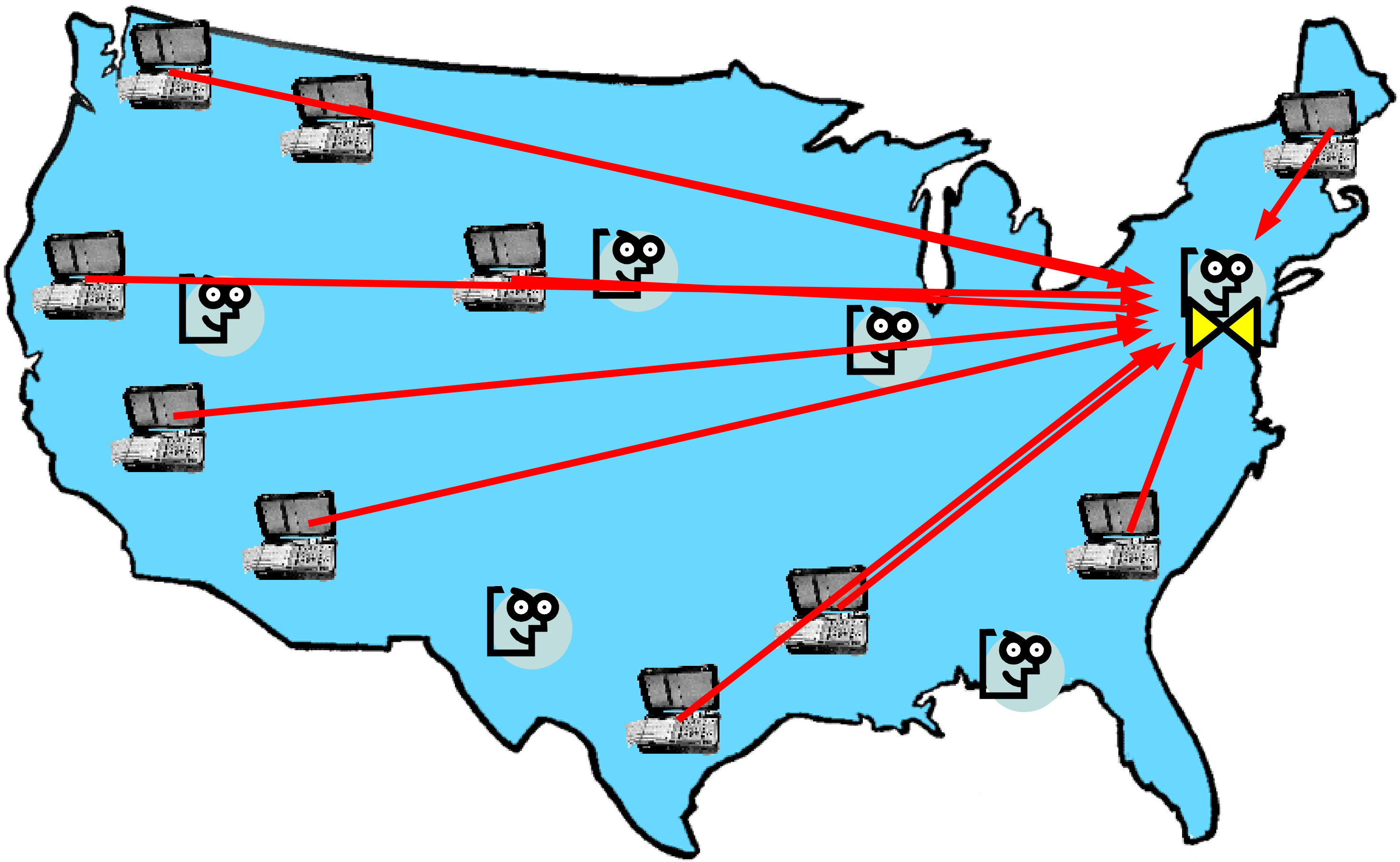
- **Producers** generate real-time data streams
  - Sensor networks, network monitors, financial markets, ...

Data Stream Applications



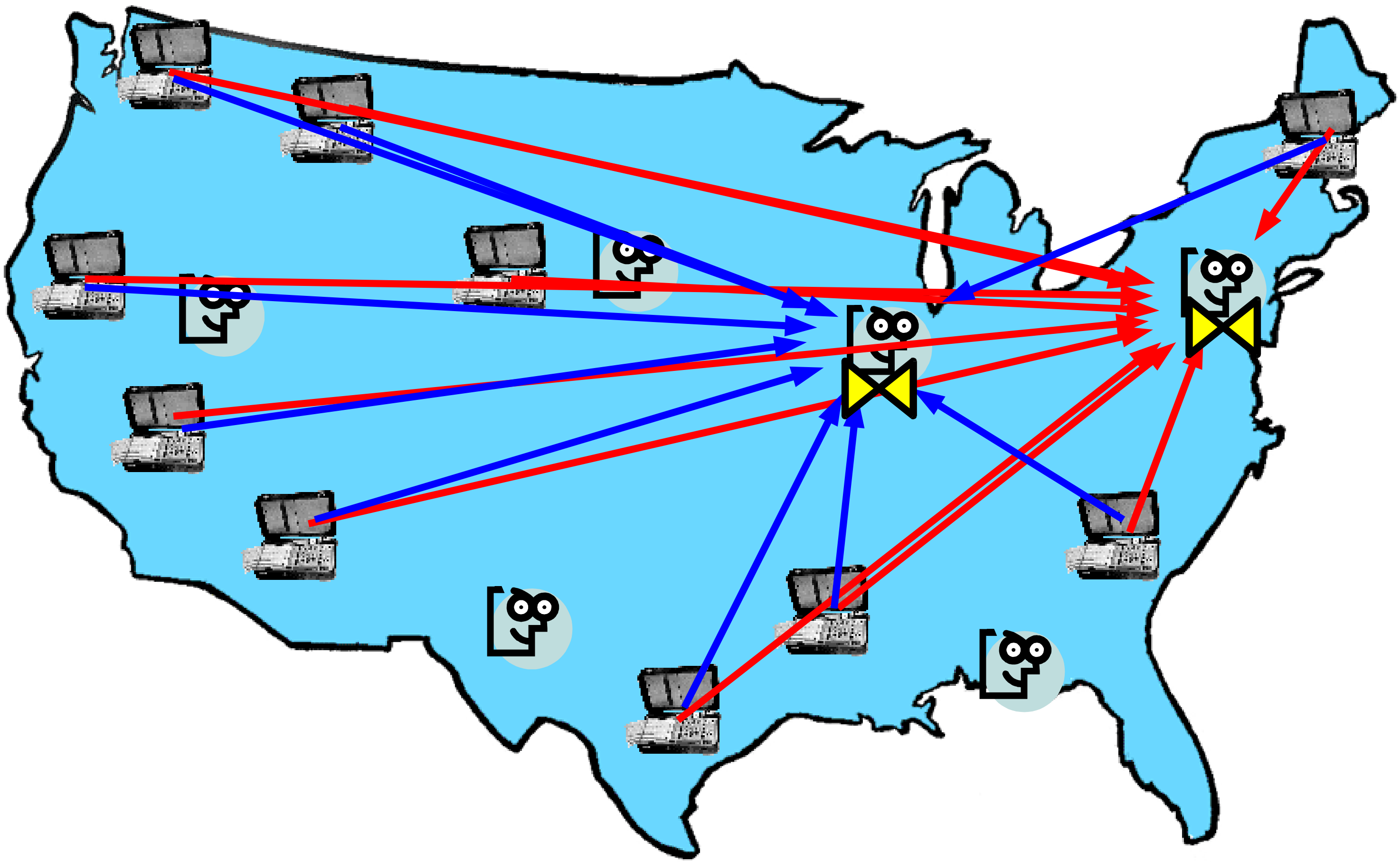
- **Consumers** submit continuous queries

Data Stream Applications



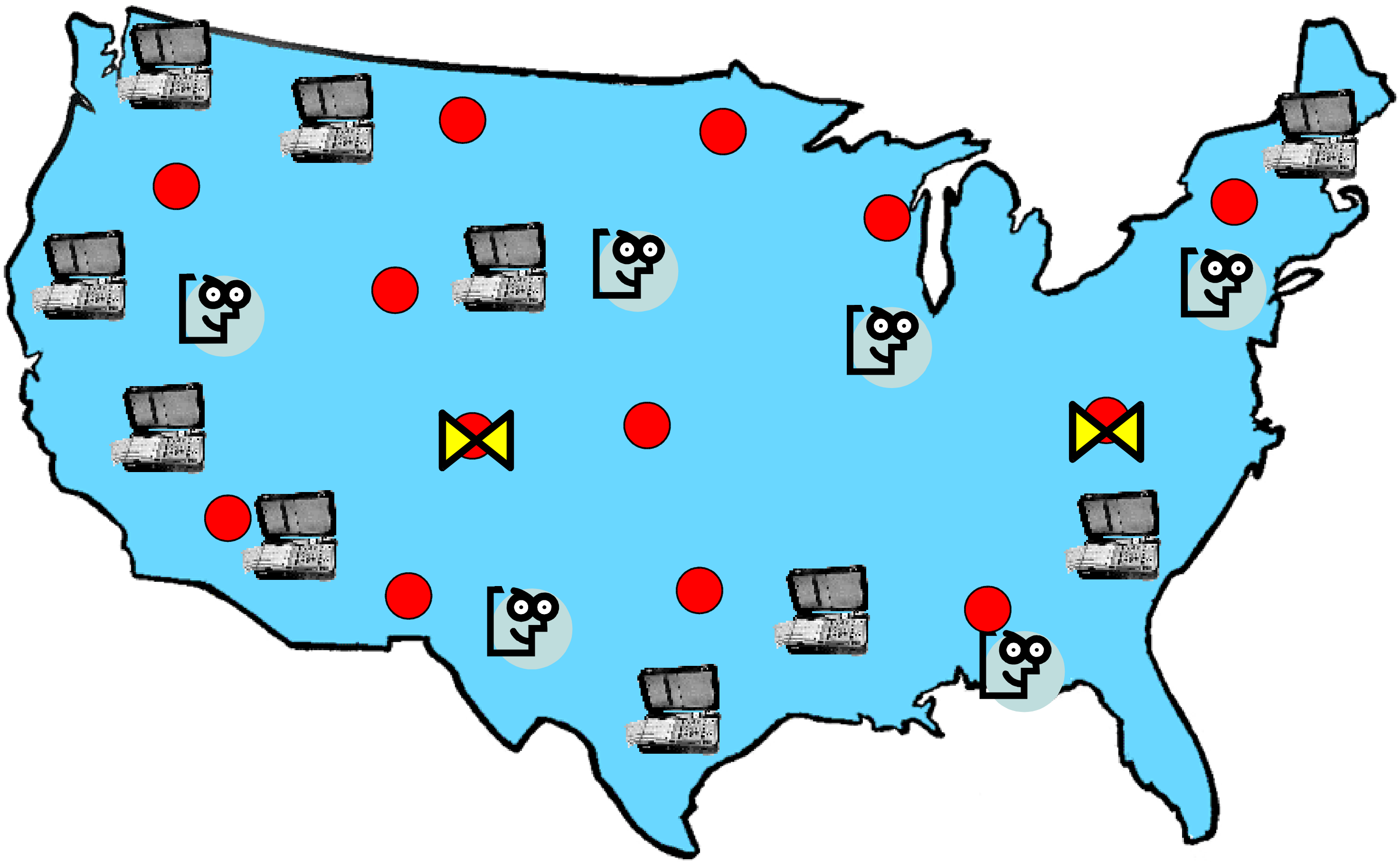
- **Services** (operators) process stream data
- Legacy way for stream processing
  - Stream data to central data warehouse

Data Stream Applications



- Inefficient if multiple consumers with similar interests
  - better: in-network processing

Stream Based Overlay Network (SBON)



- Overlay network of processing nodes
  - Leverage Internet resources
  - Re-use processing
  - Reduce network traffic

☞ **Distributed Query Optimization**

Query Optimization

1) Query Plan Generation
   - Find least-cost logical query plan
2) Operator (Service) Placement
   - Find placement nodes in overlay network for all operators

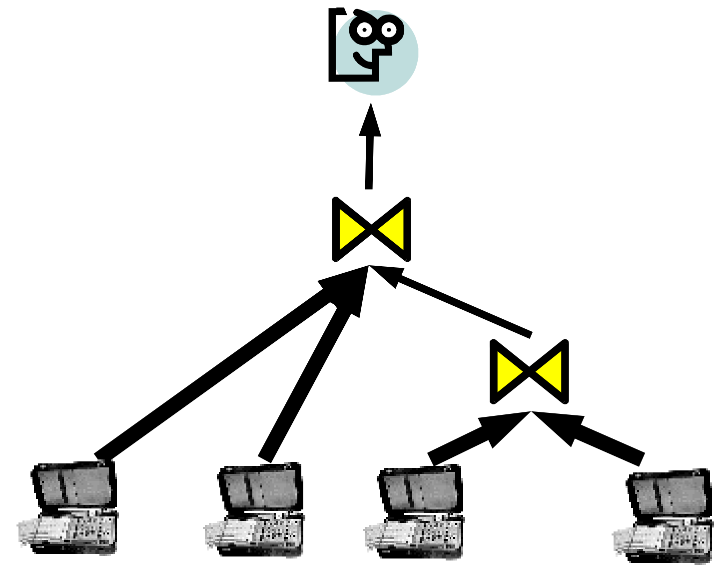
- **Problem**
  - Cost of query plan depends on service placement (network costs dominant)
  - Service placement expensive: many placement locations
  - Changing network dynamics (latency, bandwidth, ...)
- **Our Approach**
  - Reduce the cost of service placement through Cost Space
  - Consider combined cost of query plan and service placement

Overview

- Stream-Based Overlay Network
  - Service Placement
  - Cost Space
    - Virtual Placement
    - Physical Mapping
- Integrated Query Optimization
- Multi-Query Optimization
- Current Work
- Conclusions

Stream-Based Overlay Network

- Network abstraction layer
- **SBON query**
  (multiple producers, multiple opaque services, one consumer)



- **Applications**
  - **Financial data** (Borealis, Aurora)
  - **Network Health** (PHI)
  - **Streaming Scientific Data** (Hourglass, IrisNet)
- **Services**
  - **DB/Stream Operators** (Aggregate, bucket, union, join, wait-for, re-sample, map, filter, sort, ...)
  - **Custom Operators** (Compress, FFT, detect-attack, pattern matching, ...)

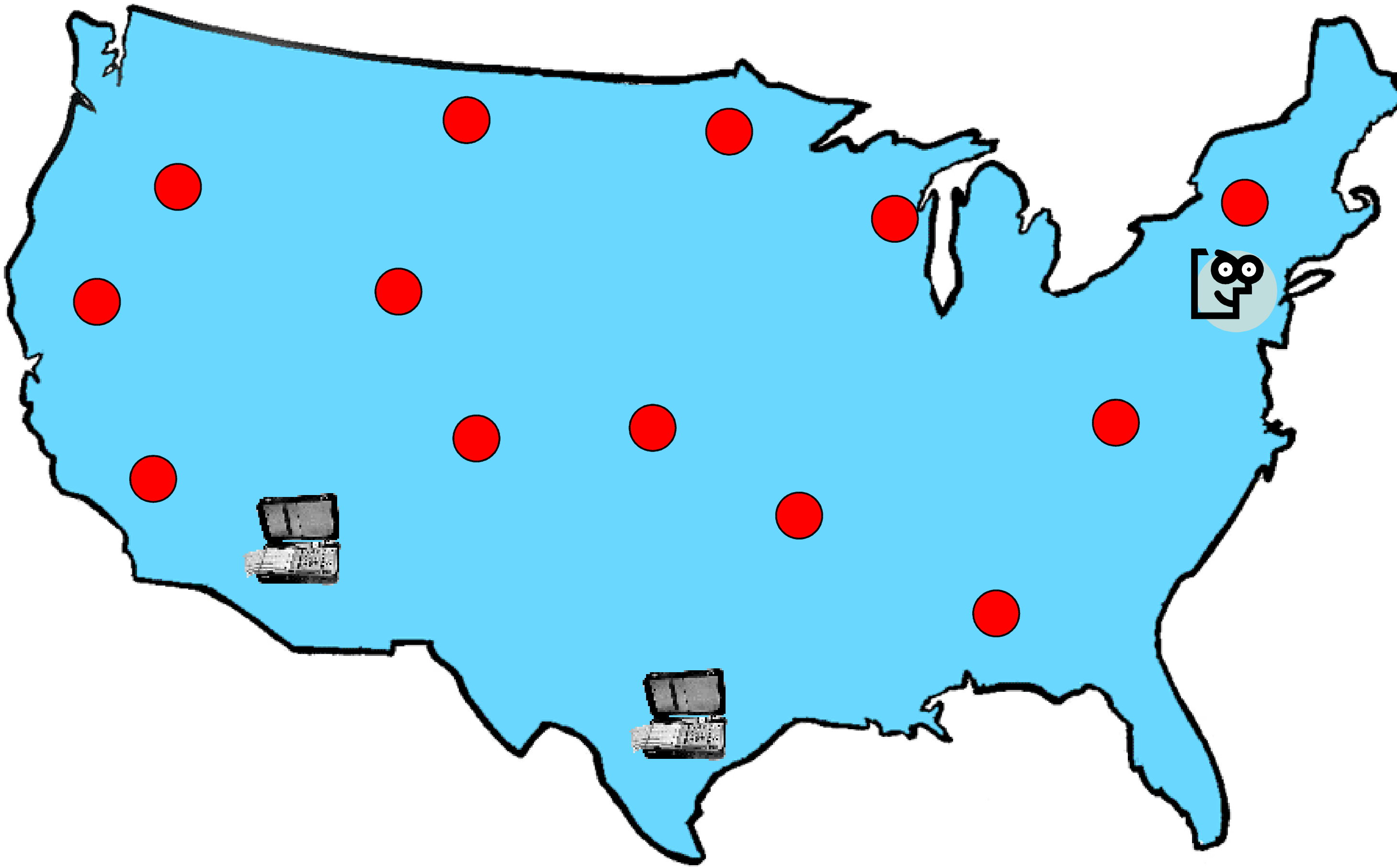
Service Placement Cost

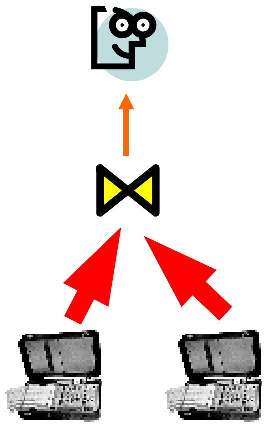
- **Application-centric Costs**
  - Latency, jitter, available bandwidth, ...
- **Global Costs**
  - Network utilization (network links, routers, ...)
  - Resource contention (node & network link stress, ...)
- **Idea**
  - Reduce latency and minimize the effect on others
  - ☞ Keep **network utilization** for a query as low as possible
  - Minimize the amount of **in-flight traffic**

    $\sum DR * Lat$

    - Product of data rate and latency
    - Assumes that high latency network links are more costly
      - Large geographic distance
      - Network congestion

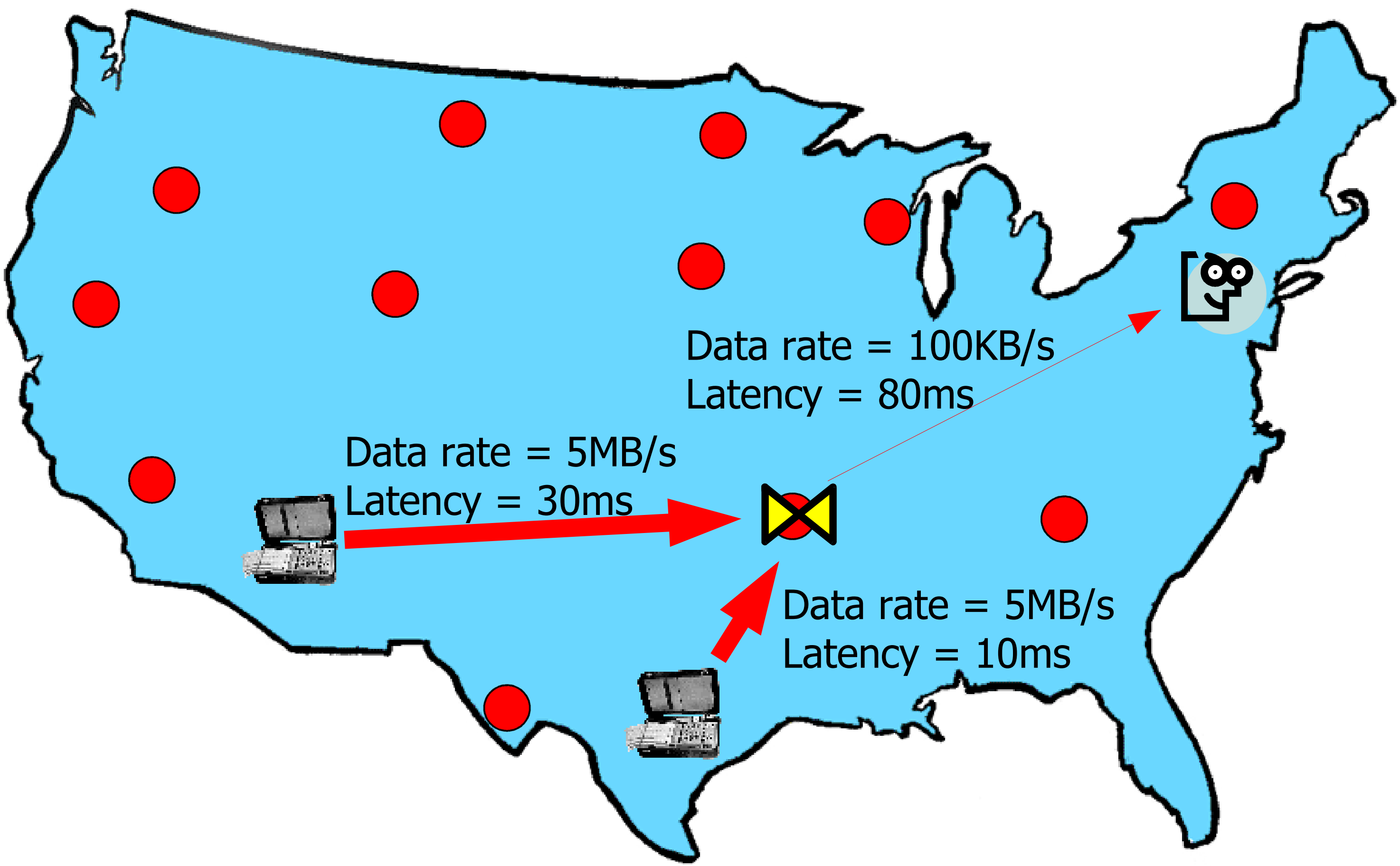
Service Placement I





- Need to instantiate the query in the SBON
  - Know or measure the **selectivity** of a service

Service Placement II



$\sum$ DR * Lat = 208KB

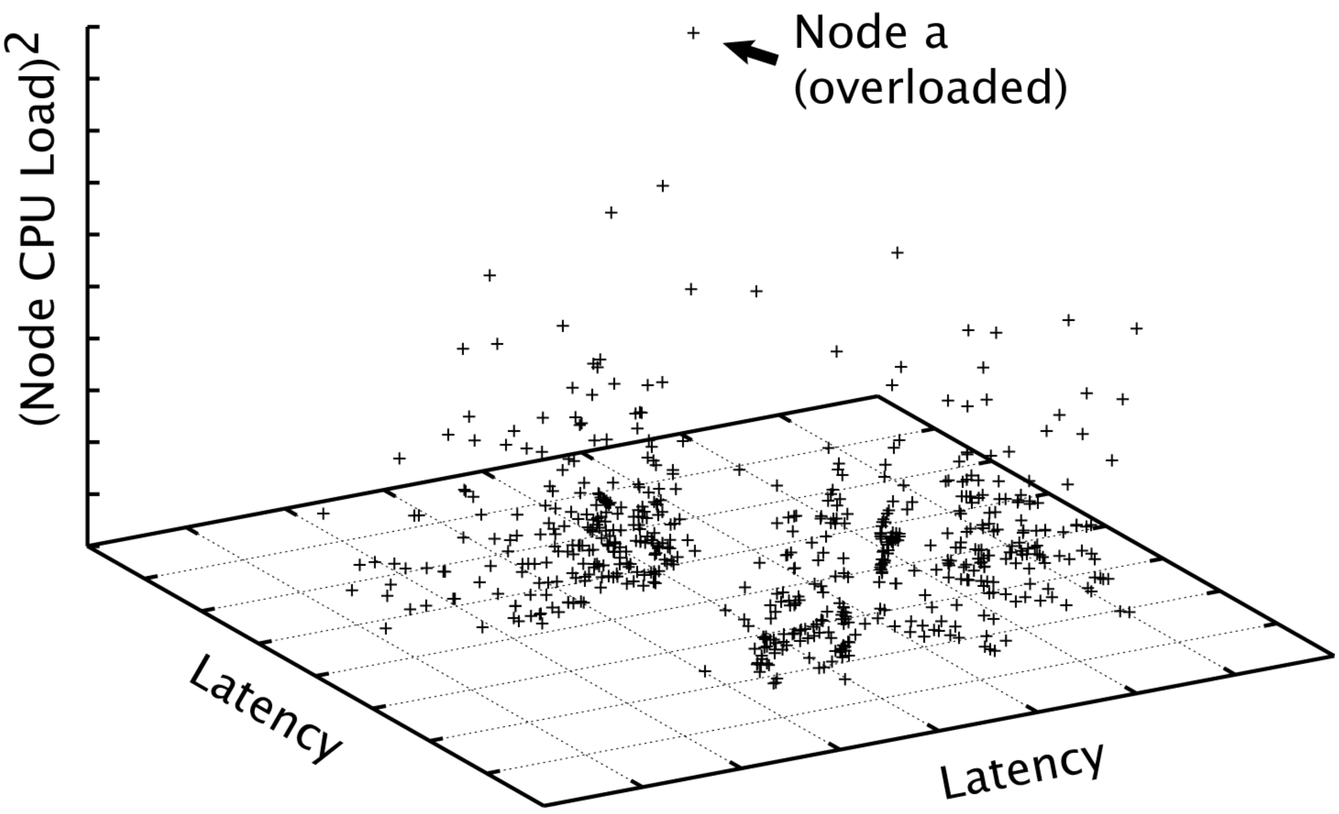
- Calculate cost per query
- but: too many overlay nodes to probe individually

Cost Space I

- Metric space that expresses costs for placement decisions
  - Euclidean distance between two points is the cost of routing data between nodes
- Dimensions encode different costs
  - **Vector costs** between two nodes
    - Latency, jitter bandwidth, ...
  - **Scalar costs** for a single node
    - CPU load, available memory, uptime, ...
- Advantages
  - Placement in mathematical space
  - Can be maintained in a decentralised fashion
    - Network coordinates for latency (Vivaldi, ...)
  - Adapts to changing network conditions

Cost Space II

- **Latency/Load** cost space
  - 2 dimensions for latency
  - 1 dimension for load (with weighting function)



Service Placement in Latency/Load Space

1. **Virtual Placement**

Calculate placement solution in latency space

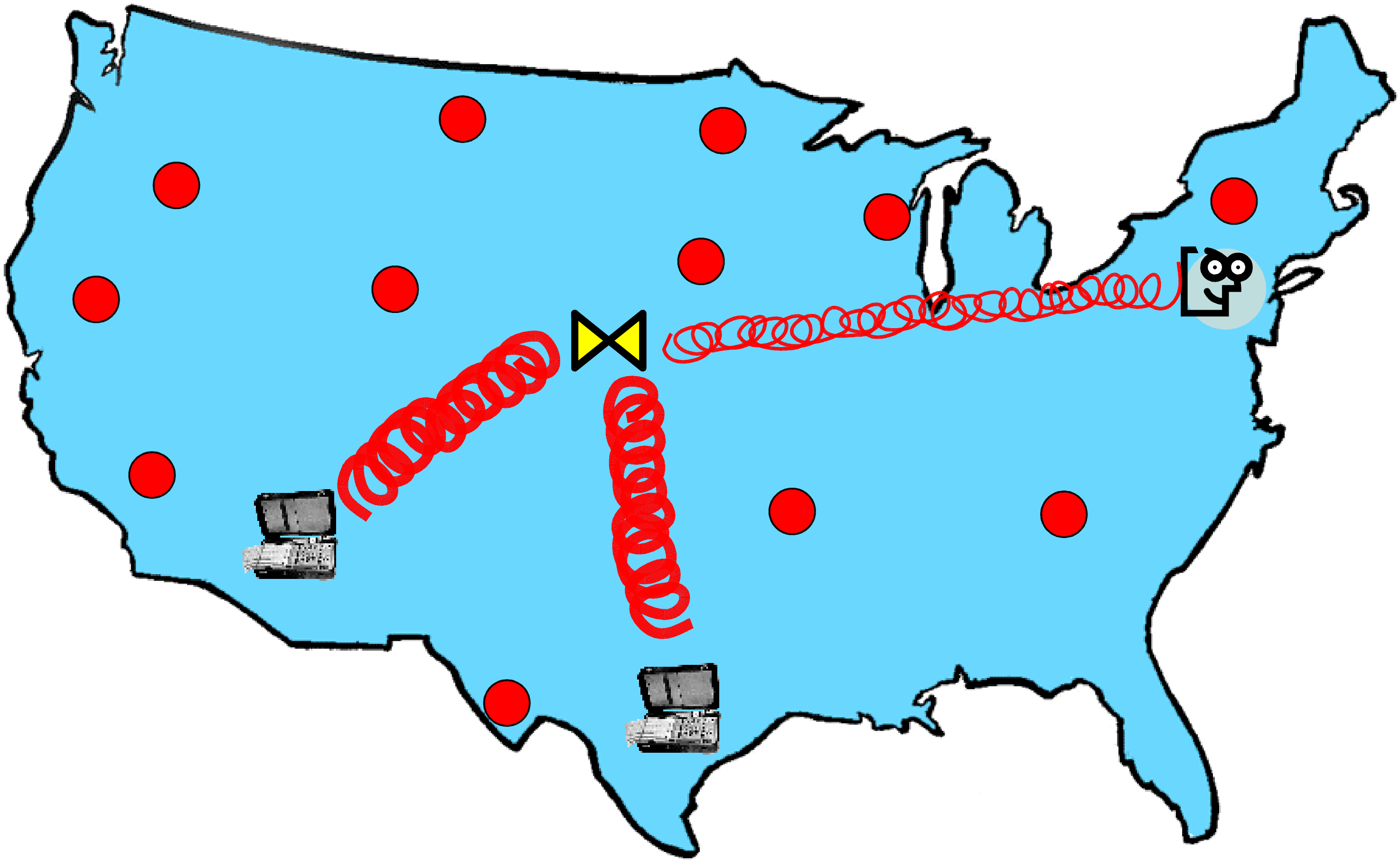
- Use **spring relaxation** to approximate best placement location in latency space

2. **Physical Mapping**

Map solution back to physical space

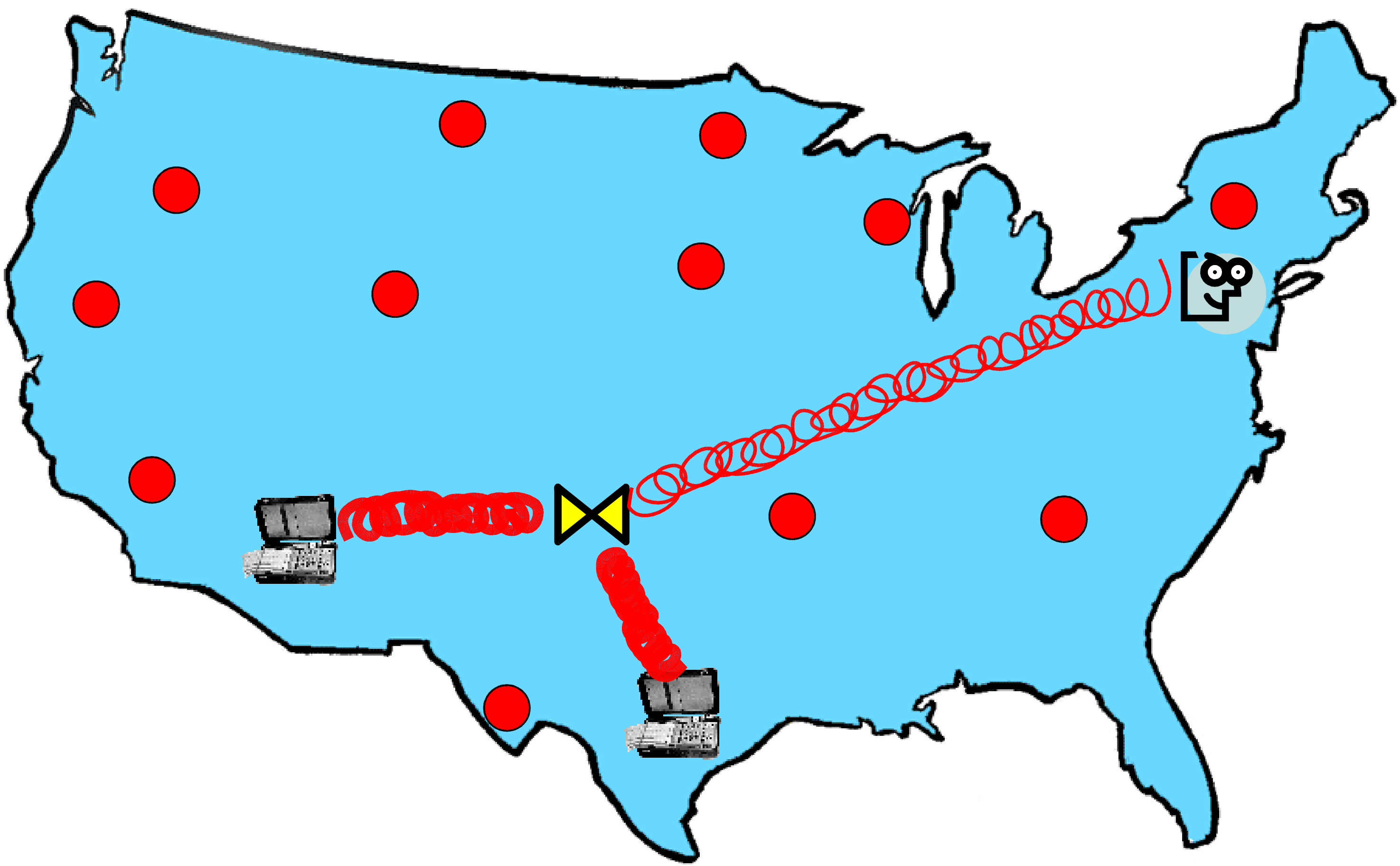
- Locate physical node closest to computed solution

Virtual Placement



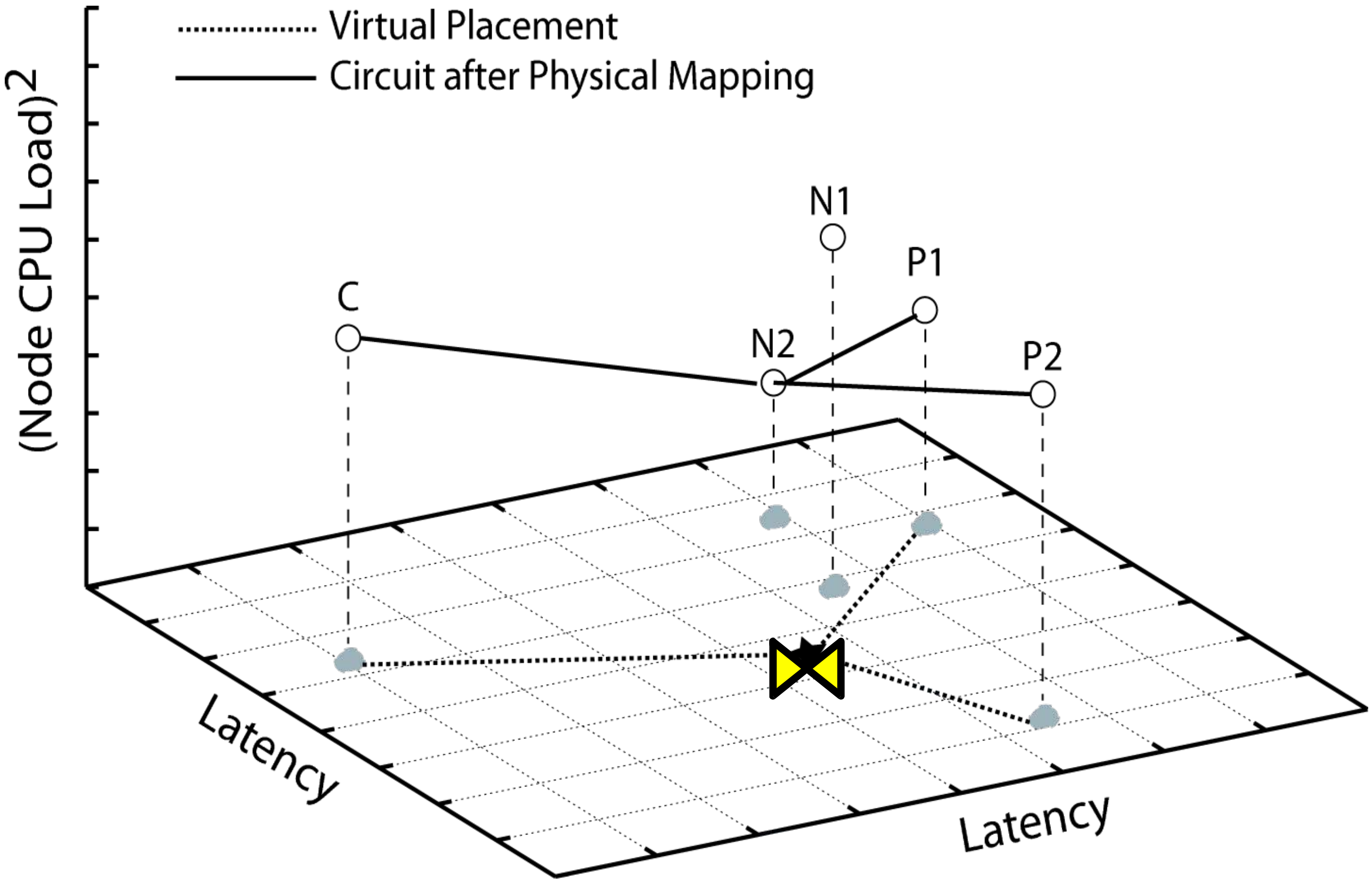
- **Relaxation Placement**: Model links as springs
  - Spring extension = Latency of link (Lat)
  - Spring constant = Data rate of link (DR)

Virtual Placement



- **Relaxation Placement**: Model links as springs
  - *Spring extension* = Latency of link (Lat)
  - *Spring constant* = Data rate of link (DR)

Physical Mapping



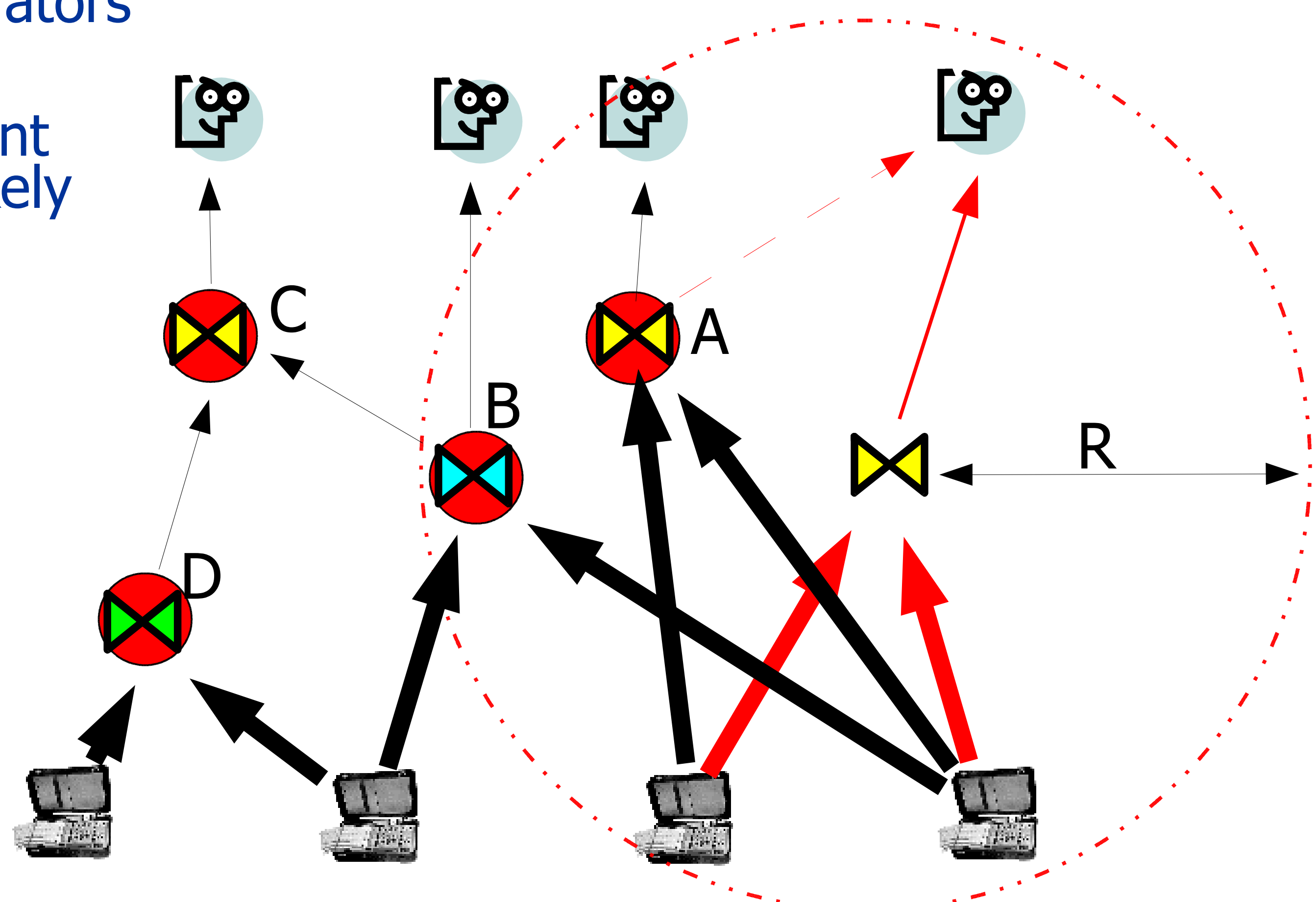
- DHT lookup to find closest existing node to desired coordinate
  - Use space-filling Hilbert curve to store n-dimensional cost space coordinate in 1-dimensional DHT

Integrated Query Optimization

- Cost space allows us to integrate plan generation and service placement by reducing the cost of service placement
- **Query Set-up at any node**
  - Generate candidate set of query plans
  - Place each query plan to calculate total cost
  - Instantiate least cost plan
- **Local query re-optimization**
  - Each node hosting an operator re-evaluates local placements
  - Migrate operator when placement changes

Multi-Query Optimization

- Find re-usable services to save processing and network resources
- Consider sphere of radius R in cost space
  - Only reuse operators within sphere
  - Plans with distant placement unlikely to be useful



Current Work

- Running SBON implementation on PlanetLab that supports
  - Load/latency cost space
  - Service migration
  - Simple Java application and Borealis
- Other metrics for cost spaces
  - Bandwidth, jitter, reliability, ...
- Interaction between SBON optimizer and application
  - Interfaces to describe service and data semantics to SBON
  - Decomposition of services, coverage among services, ...

Conclusions

- Large-scale data stream apps require new infrastructures
  - Support for in-network stream processing
  - **Stream-Based Overlay Network (SBON)**
- **Query optimization** faces new challenges
  - Vast search space for service placement
  - A good logical query plan may lead to only bad placements
- **Cost spaces** are a useful abstraction to address this
  - Reduce the cost of service placement decisions
  - Virtual placement and physical mapping
  - Decentralized, flexible, and adaptable to network dynamics
  - Discovery of existing services for multi-query optimization

Thank You. Any Questions?

The Hourglass Project

http://www.eecs.harvard.edu/~syrah/hourglass

hourglass@eecs.harvard.edu

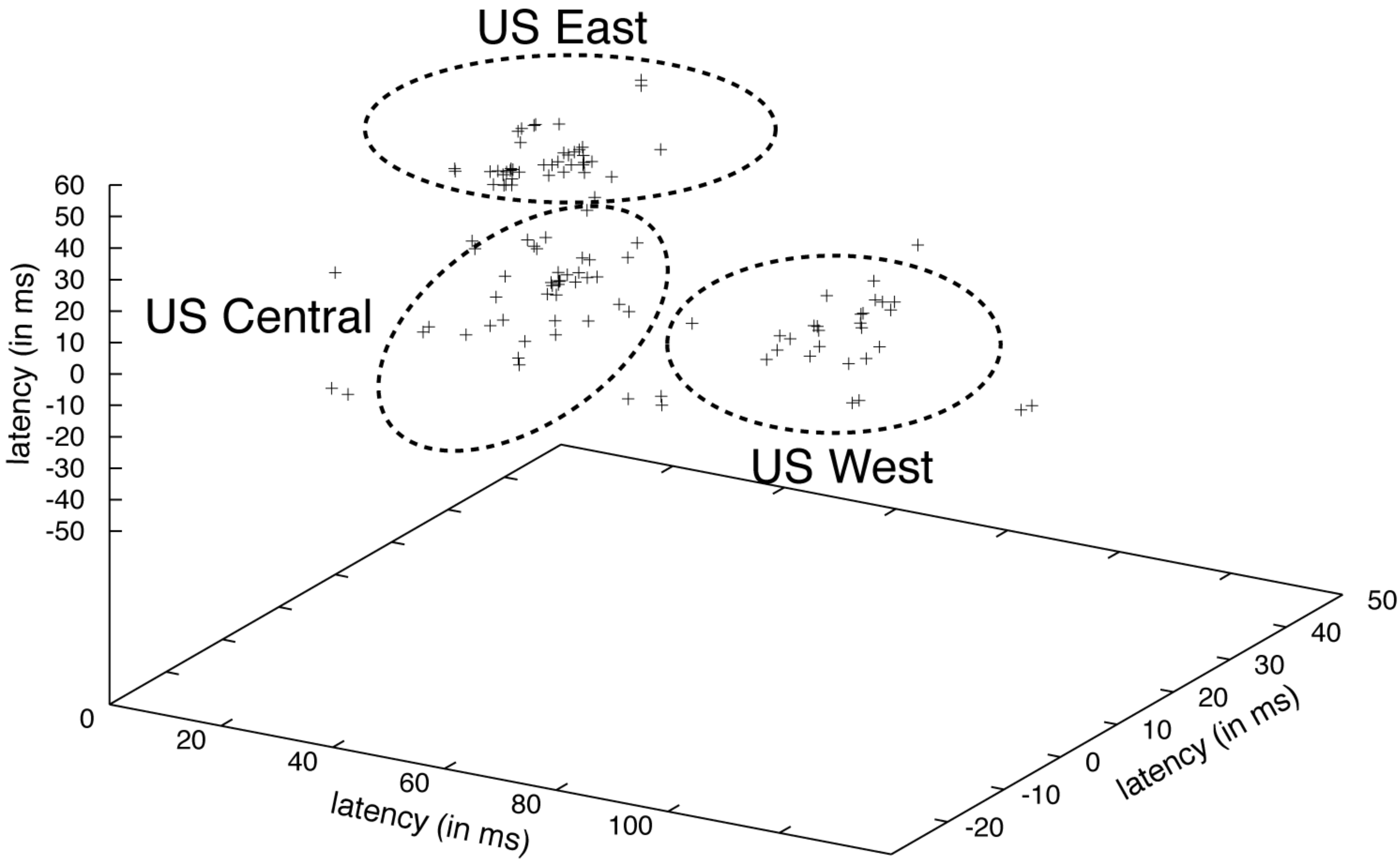
Peter Pietzuch

http://www.eecs.harvard.edu/~prp

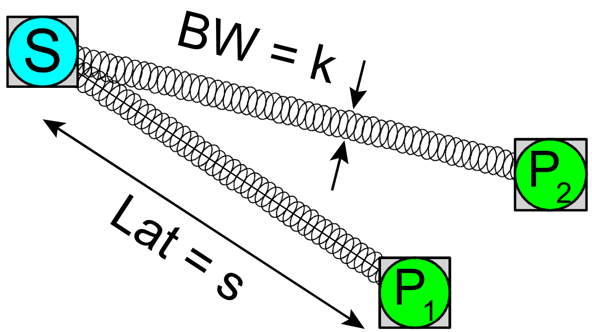
prp@eecs.harvard.edu

Backup Slides

PlanetLab Latency Space



Spring Model



- Network of springs tries to minimize potential energy E

$F = \frac{1}{2} * k * s$

- where k is the spring constant and s is the spring extension

$$\Sigma E = \Sigma F * s = \Sigma \frac{1}{2} * k * s^2$$

- where E is the potential energy

$$\Sigma [BW * Lat]^2$$

- Cost function for placement