Path Optimization in Stream-Based Overlay Networks

Peter Pietzuch,
prp@eecs.harvard.edu

Jeff Shneidman, Jonathan Ledlie, Mema Roussopoulos, Margo Seltzer, Matt Welsh

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

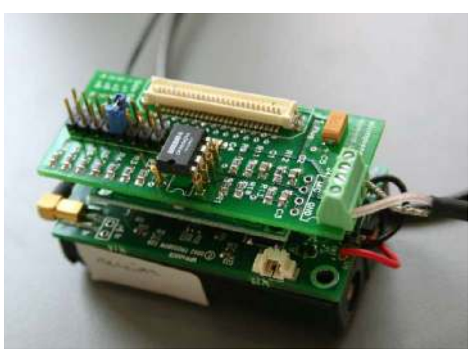
Intel Research, Berkeley – November 2004

Volcano Monitoring

- Logging seismic activity of a volcano (Tungurahua in Ecuador)
  - Sensors (Motes) sample low-frequency infra-sound
  - Many sensors survey the physical structure ➡ mountain tomography

- Need to get data to users

- Goals
  - Satellite up-link from base station at volcano
  - Fuse data from ground sensors and satellites
  - Push queries into network
  - Collaboration between research institutions



Emergency Medical Care

- Sensor support for medical applications
  - Motes attached to patients collect vital sign data (pulse ox, heart rate, EKG, ...)
  - PDAs carried by EMTs receive data and enter field reports
  - Ambulance correlates with patient records at hospital
  - "*Generate a list of the top 10 most critical patients at a disaster site.*"
- Characteristics

  - Real-time stream data
  - Many heterogeneous data sources
  - Partial network connectivity

Application Features

- Large number of distributed data producers
- In-network, real-time processing of data streams
  - Leverage the resources in the network
  - Aggregation close to data producers
- Multiple applications sharing data producers

➡ **Need for a reusable and efficient Internet infrastructure for distributed data collection and processing**

- Scalable, distributed, fault-tolerant implementation
- Optimization techniques for efficient resource utilization
- Fast deployment of novel applications

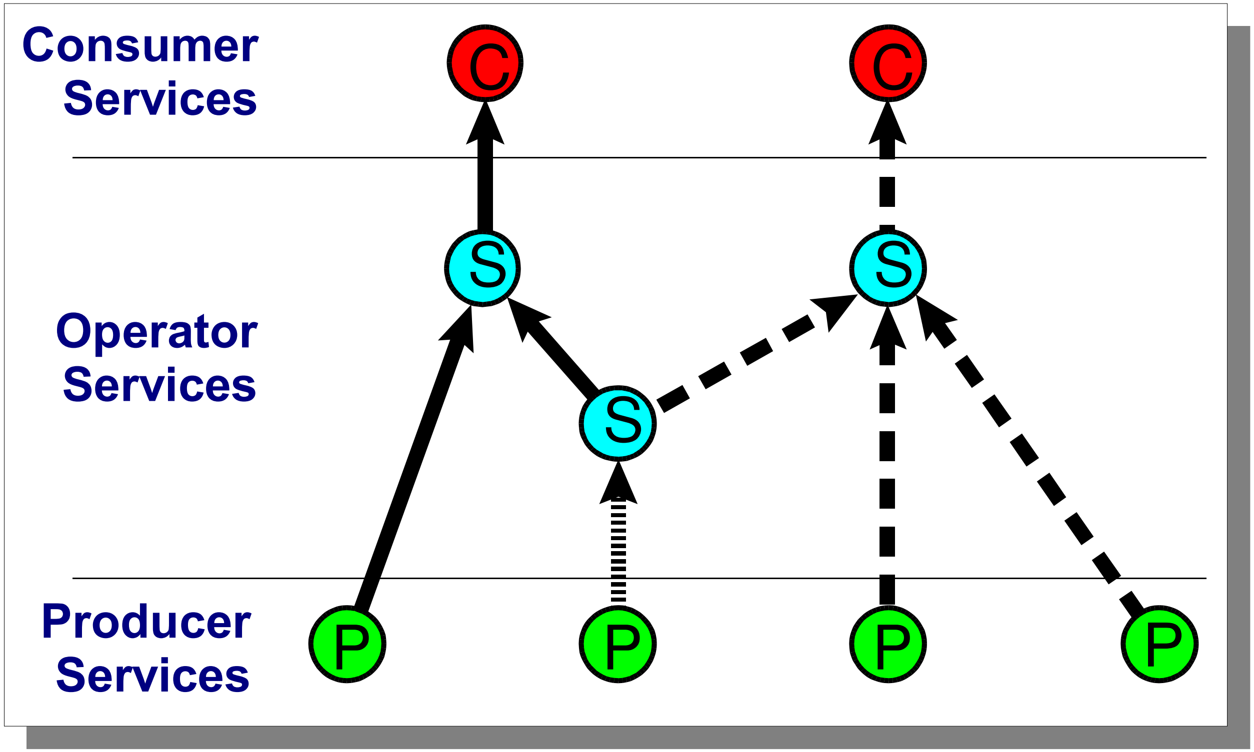
➡ **Stream-Based Overlay Network**

Overview

- Part I: Stream-Based Overlay Networks
  - Circuits
  - Services
  - Hourglass prototype
- Part II: Data Path Optimization
  - Placement Problem
  - Relaxation Placement Algorithm
  - Evaluation
- Conclusions

Part I: Stream-Based Overlay Networks

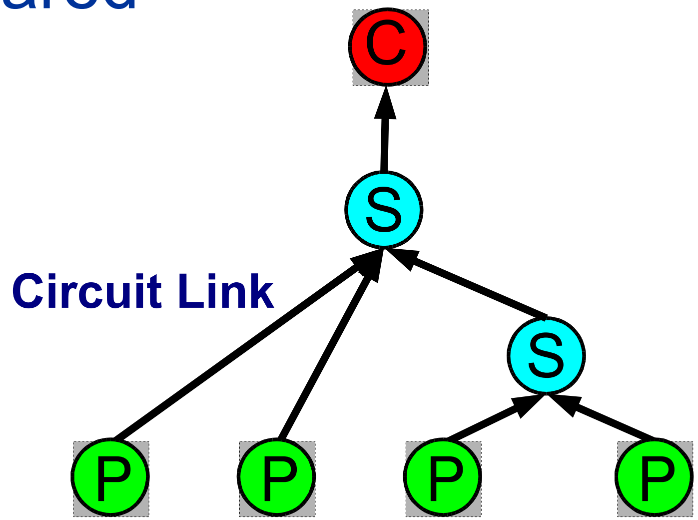
- Stream data from producers to consumers with in-network processing
- **Services**
  - Consumer
    - Apps, users, ...
  - Producer
    - Sensor nets, DBs, ...
  - Operator
    - Filter, join, aggregation, ...
- **Circuits**



- Features
  - Overlay network of nodes using the Internet
  - Connection-oriented: applications establish circuits
  - Efficient data transport through overlay network

Circuits

- Data paths in the overlay network
  - Established by applications to satisfy data need
  - Equivalent to logical query expression
  - Trees with a single consumer services as root
- Services & circuit links can be shared across circuits

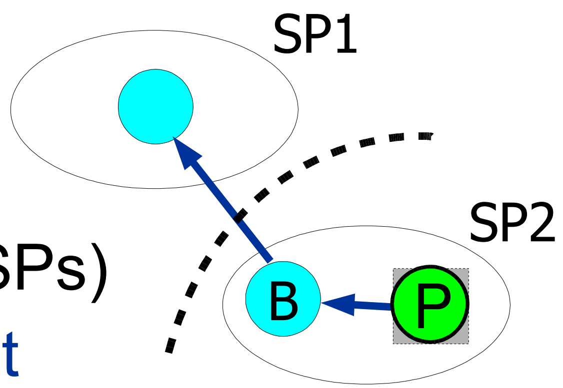


Services

- Generic Service Model
  - No single data model
    - Support relational, semi-structured, binary, ...
  - No fixed set of operators
    - User-defined services for different applications
  - Assumptions about properties of operators
    - Selectivity, aggregation, ...
- Services Classes
  - **Pinned Services** C P
    - come with predefined locations at nodes
    - e.g. Producer & Consumer services, storage-backed, ...
  - **Unpinned Services** S
    - can be instantiated at different locations
    - are placed or unplaced

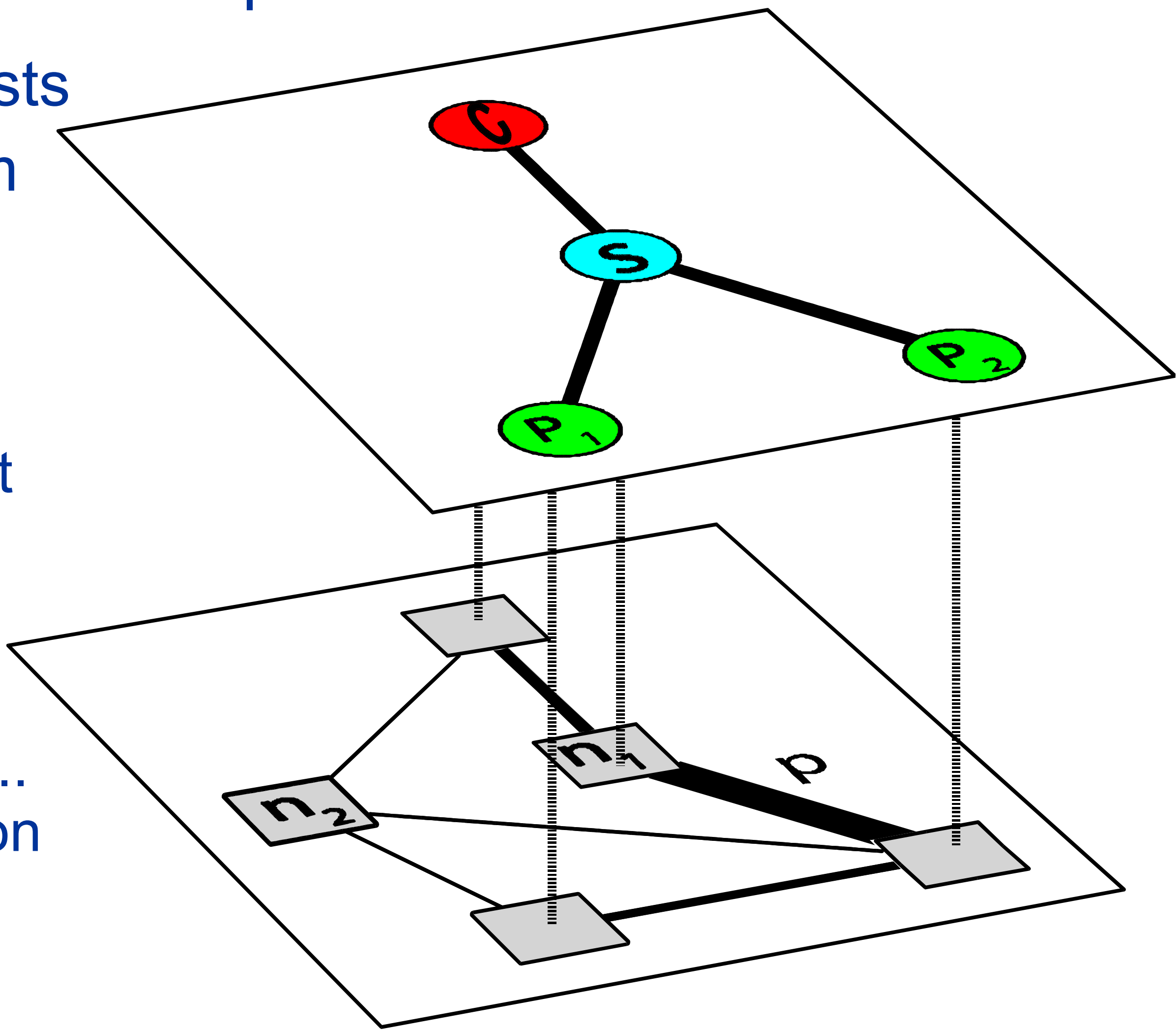
The Hourglass System

- Prototype implementation of an SBON
  - TinyDB sensor networks as Data Producers
  - Semi-structured data model with XML-defined circuits
  - Deployed on ModelNet, PlanetLab, and the Internet
- Support for **disconnected operation**
  - Circuits may cross network boundaries
  - Services are part of **Service Providers** (SPs)
  - Disconnection between SPs is transparent to application
  - **Buffer Services** are instantiated on demand



Part II: Path Optimization

- Where are unpinned services placed in the SBON?
  - Placements have costs
  - Optimization problem
- Cost functions
  - **Application costs**
    - Path delay in circuit
    - Jitter in circuit
  - **Global costs**
    - Network utilization
      - Links, routers, ...
    - Resource contention
      - Node stress
      - Link stress



Placement Problem

- Service placement should reflect underlying physical network topology
  - **Bandwidth-Latency** (BW-Lat) product for network utilization
    - Amount of traffic in transit in the network
    - Avoid long latency links when possible

$$\sum BW * Lat$$

- Search for optimal solution too expensive
  - Traditional optimization approaches exponential in number of services
  - Also require global knowledge of nodes & links
    - Instead, strive for efficient, approximate solution
- Requirements for placement algorithm in SBONs
  - *Scalable* with decentralized implementation
  - *Adaptive* to changing network and stream conditions
  - *Efficient* placement decisions with respect to cost functions

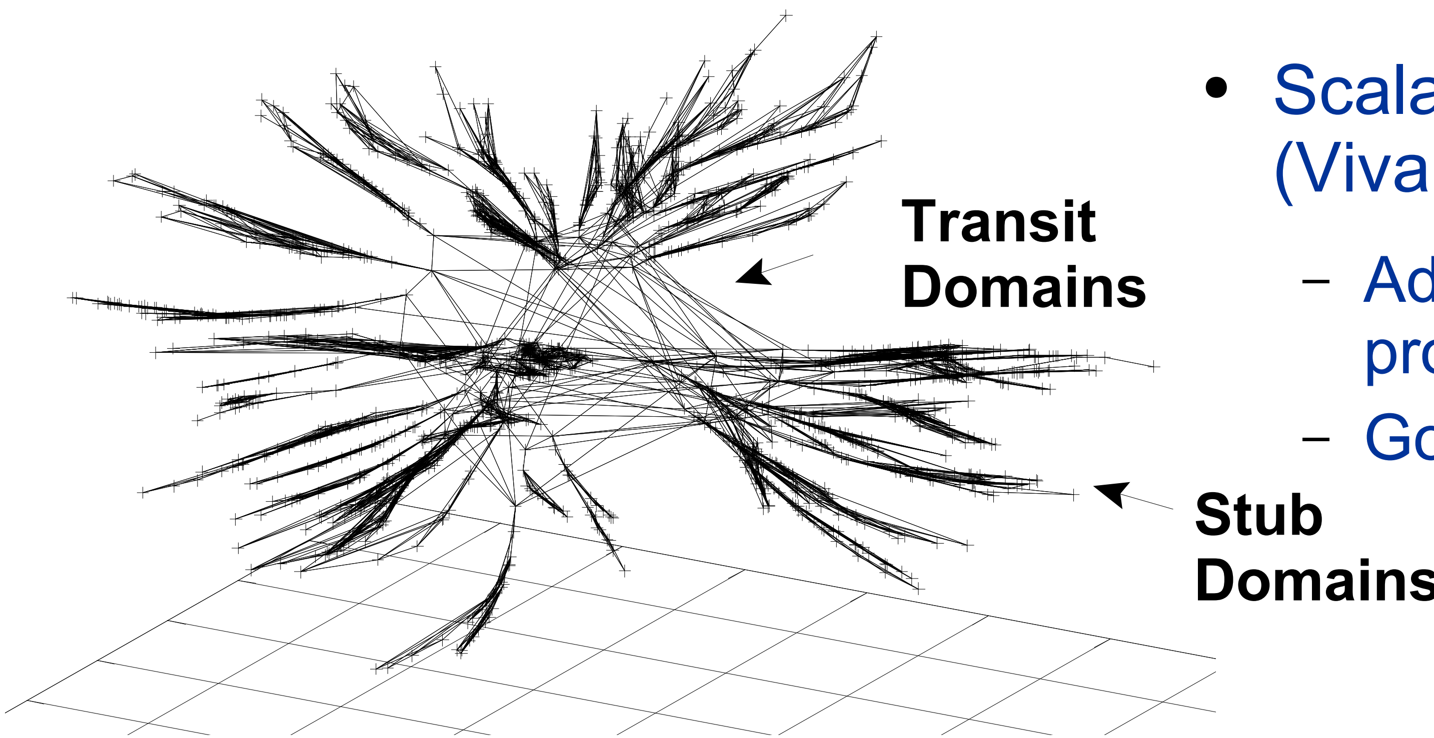
Relaxation Placement

- Latency information crucial for cost of placement
- Solve the problem in a virtual latency space

1. Calculate placement solution in latency space
   - Use spring relaxation to approximate best placement location in latency space
2. Map solution back to physical space
   - Locate physical node closest to computed solution

Latency Space

- Every node has an artificial **latency coordinate**
- Metric space with distance ≈ communication latency
- Efficient encoding of global topology knowledge



1550-node transit stub topology in latency space

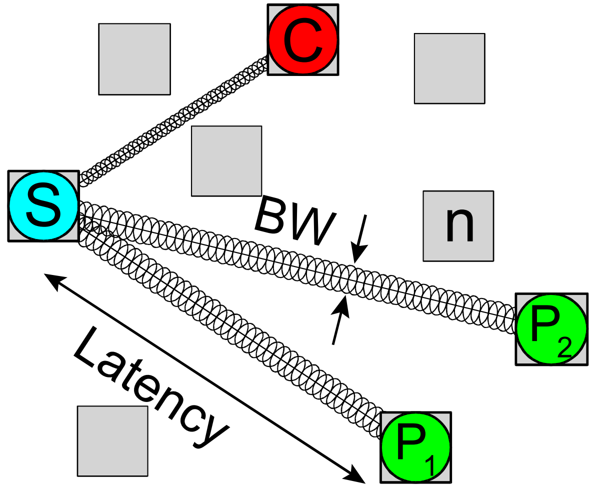
- Scalable implementation (Vivaldi [Dabek04])
  - Adaptive with little probing overhead
  - Good latency prediction
- Transit domains at center; stub domains at edges

Spring Relaxation

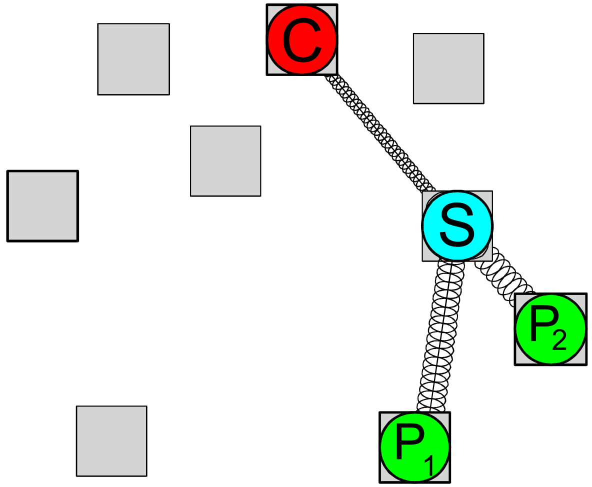
- Model circuits as a network of springs
  - *Spring extension = latency* of circuit link Lat
  - *Spring constant = bandwidth* of circuit link BW
- Minimize network utilization:

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

  - Springs "pull" according to bandwidth usage
  - Take selectivity and fan-in of services into account



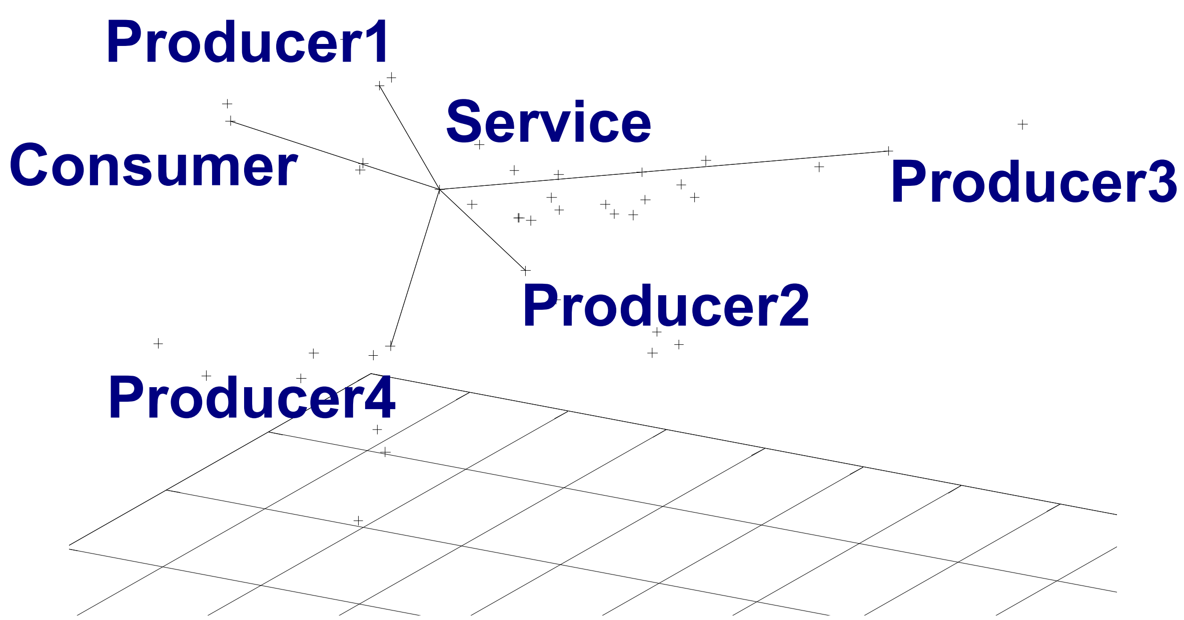
Before relaxation



After relaxation

Relaxation Algorithm I

- Each unpinned service executes the Relaxation algorithm
  - Service determines its updated coordinate in relation to its circuit neighbors
  - After convergence, the unpinned service may migrate to a different physical node
- Use DHT to map virtual coord. to closest physical node
  - Use *Hilbert curves* to map 4-d latency coord. to 1-d DHT key
  - DHT routes to closest existing latency coord.
  - Consider k-closest nodes and find possible placement



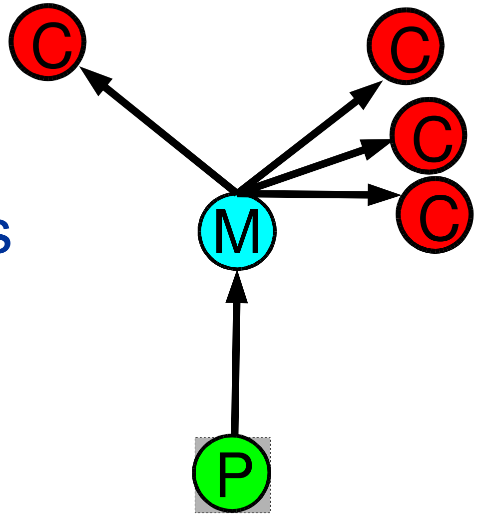
Actual Relaxation placement in latency space with 42 nodes

Relaxation Algorithm II

- **Mapping error** from latency to physical space
  - Error is less than 13% for transit-stub and 5% for PL topology of network diameter
- Advantages
  - Decentralized implementation
    - No global state about all nodes or circuits
    - Local knowledge only
  - Little probing overhead for latency information
  - Adaptable to changes in circuit structure and network conditions

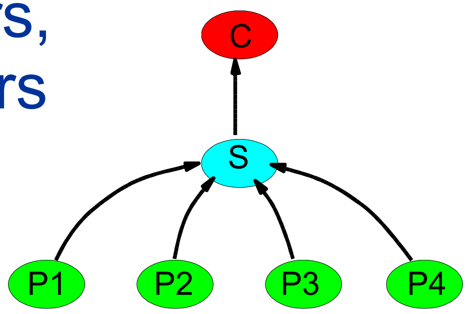
Cross-circuit Optimization

- Relaxation placement makes placement decisions involving multiple circuits
  - Consider **circuit graph** as network of springs
  - Placement decision for a new circuit may influence previous placements
- Simple form of cross-circuit optimization: **Multicast (M/C) Services**
  - Producers reused among multiple services
  - A circuit analyzer inserts M/C services on demand
  - Place M/C service close to consumers
  - Placed like other services using Relaxation placement



Evaluation

- Experimental Set-up
  - Experiments in discrete-event simulator
  - *PlanetLab* and *Georgia Tech transit-stub* topologies
  - 1000 simple circuits with 4 pinned producers, 1 unpinned service, and 1 pinned consumers



- Evaluation Goals
  - Network utilization (**BW-Lat product**)
  - Application delay penalty (**Delay Stretch**)
  - Resource contention (**Node stress**)

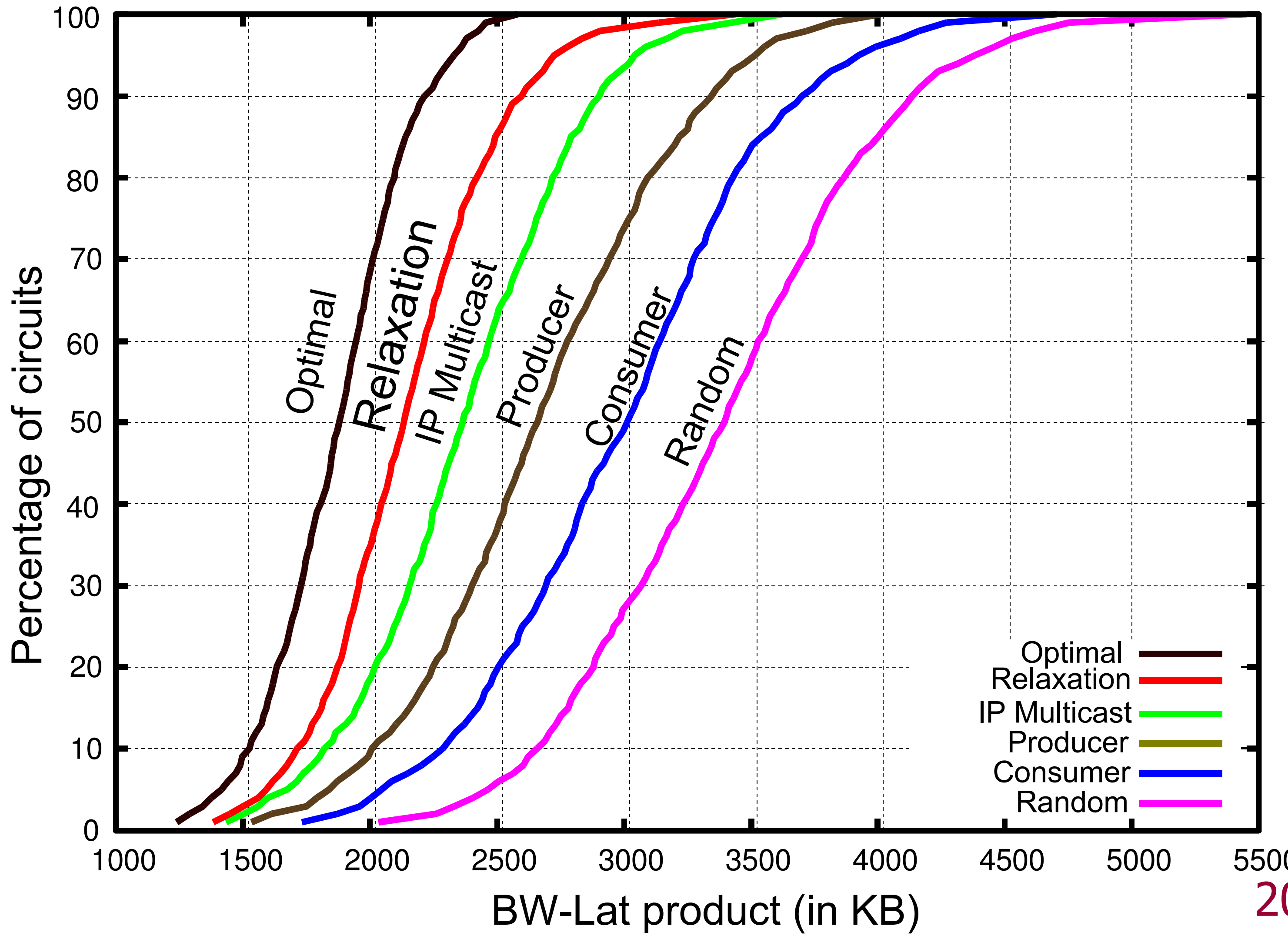
Placement Algorithms

- 7 Placement algorithms in simulator:

Optimal	Placement at optimal node - exhaustive search for optimal solution
Relaxation	Placement found by Relaxation algorithm - centralized implementation
IP Multicast	Placement at IP Multicast routers - requires network support
Producer	Placement at producer node - some stream-processing systems, e.g. Borealis
Consumer	Placement at consumer node - centralized data warehouse
Random	Placement at random node - worst case comparison

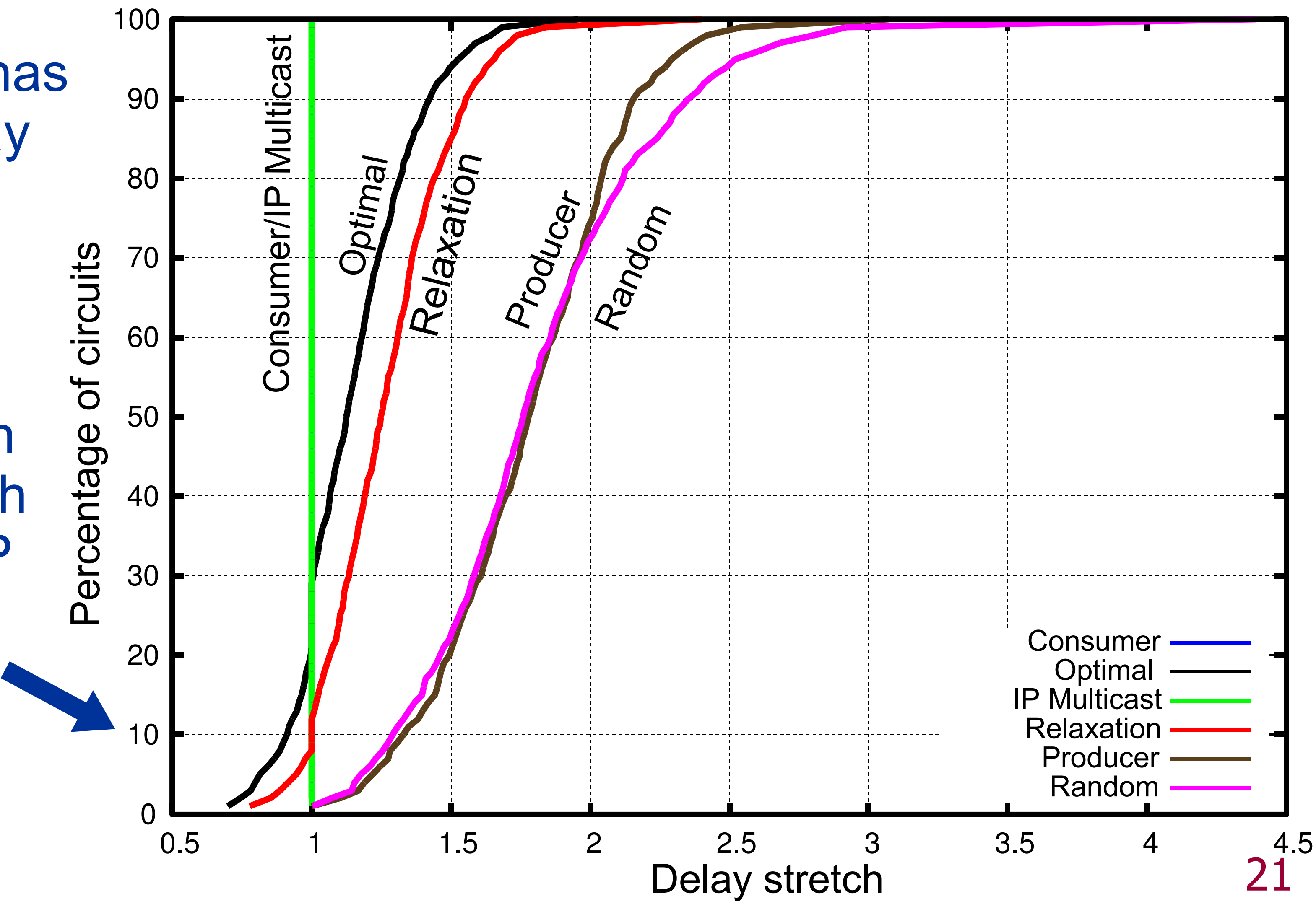
Network Utilization

- CDF of traffic in transit (BW-Lat product) for 1000 circuits
- Relaxation close to Optimal
- IP Multicast reduces hops not latency
- Producer better than Consumer



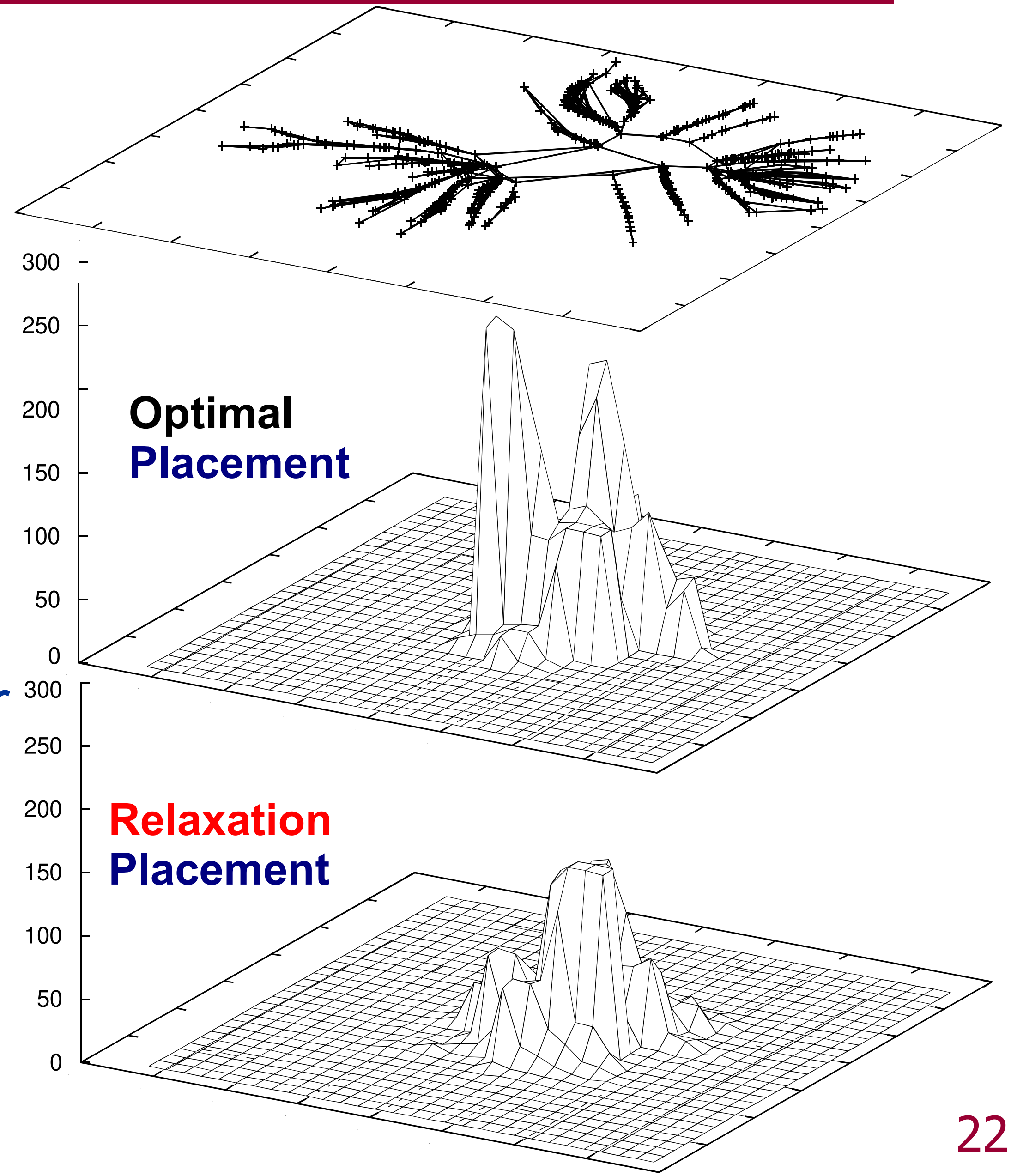
Application Delay Stretch

- CDF of delay stretch (ratio of optimal) for 1000 circuits
- Consumer/ IP Multicast has smallest delay
- Relaxation close to Optimal
- Small fraction of circuits with better than IP delay



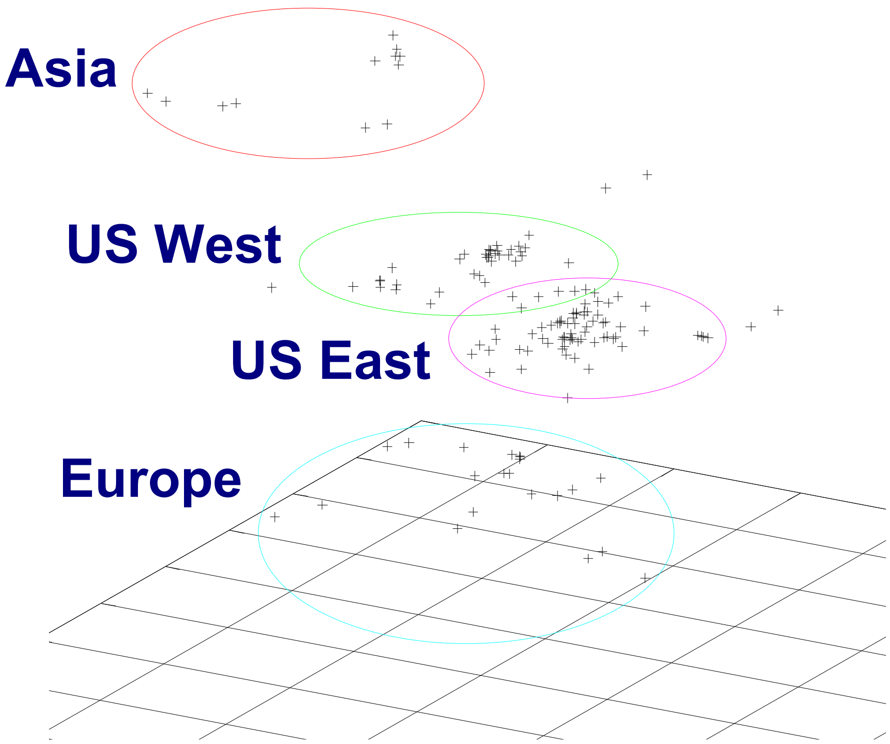
Resource Contention

- Distribution of service placement
  - Load-balancing?
- Transit-domains more popular for service placement
  - Traffic goes there anyway
  - Enable transit domains for service placement
- Maximum number of placed services
  - Spreading the load
  - “Power of 2 choices”



PlanetLab Results

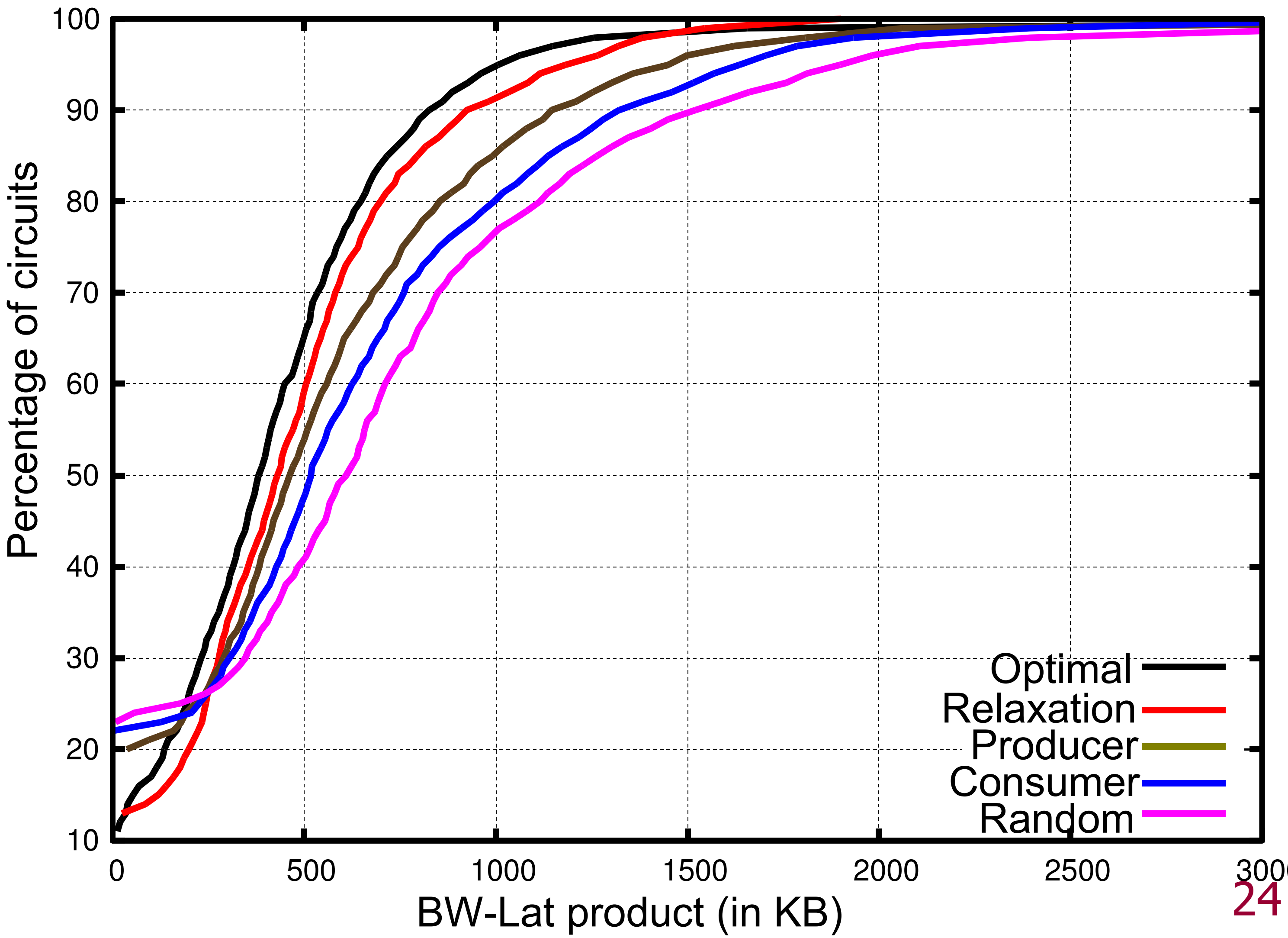
- Validated transit-stub results with PlanetLab topology
- Used simulator to place circuits off-line with all-pairs ping data
- Deployed circuits with **Hourglass** for application delay measurements



PlanetLab in latency space

Network Utilization - PlanetLab

- CDF of traffic in transit (BW-Lat product) for 1000 circuits
- Similar results for transit-stub topology
- Small fraction of measure-ment anomalies in all-pair ping data



Related Work

- *Distributed Stream Processing*
  - **Medusa/Aurora/Borealis** (MIT): Stream processing with load management; market-based optimization; network-aware placement [Ahmad04]
  - **IrisNet** (Intel Research): M/W for sensor apps; hierarchical node organization; semi-structured data model
  - **PIER** (Berkeley): Distributed database based on DHT
  - **Grid** (OGSA): Large-scale resource sharing
  - **DistCED** (Univ. of Cambridge): Pattern detection in steams
- *Distributed Query Optimization*:
  - **Mariposa** (Berkeley): Economic approach to query opt.
- *Sensor Network Programming*
  - **Cougar** (Cornell), **TinyDB** (Berkeley): relational data model; limited optimization capabilities

Future Work

- Fully-decentralized implementation on PlanetLab
  - Adaptable to network dynamics and circuit evaluation
  - Convergence results for distributed relaxation
  - Investigate circuits used by realistic applications
- Explore potential of cross-circuit optimization
  - Large-scale circuit optimization
    - “Traditional” distributed query optimization techniques
    - Service decomposition and service reuse
  - Dynamic circuits
    - Discovery of pinned services
    - Modify pinned services depending on demand

Conclusions

- SBONs enable future sensor applications
  - Service placement is a crucial problem in SBONs
  - Efficient network utilization is important
- Relaxation Placement
  - Spring relaxation technique in latency space
  - Scalable decentralized implementation
  - Supports cross-circuit optimization
- Evaluation
  - Relaxation placement is close to optimal in terms of network utilization
  - Low delay penalty
  - Need for load-balancing mechanisms

Any Questions?

The Hourglass Project

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

hourglass@eecs.harvard.edu

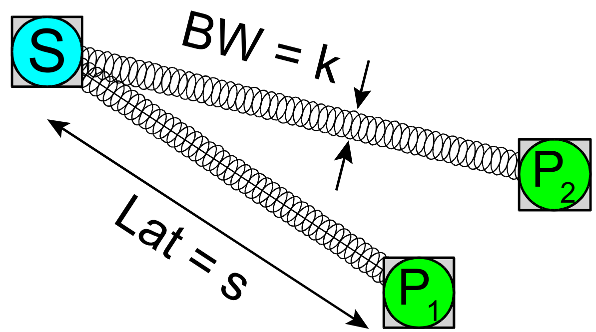
Peter Pietzuch

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

prp@eecs.harvard.edu

Backup Slides

Spring Model



- Network of springs tries to minimize potential energy E

$F = ½ * k * s$

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

$$\Sigma E = \Sigma F * s = \Sigma ½ * k * s^2$$

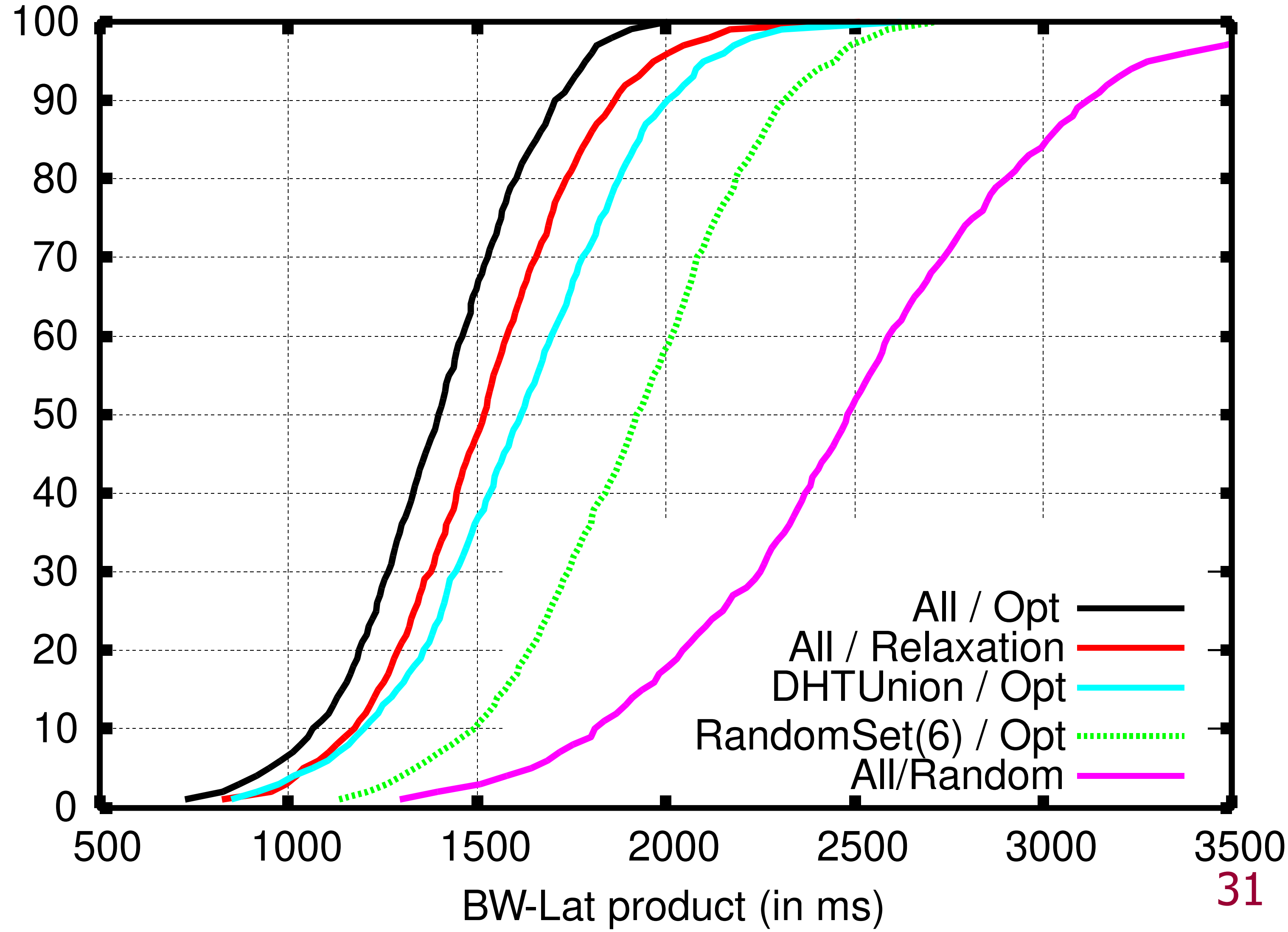
- where E is the potential energy

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

- Cost function for placement

Network Utilization – DHT Placement

- CDF of traffic in transit (BW-Lat product) for 1000 circuits
- DHT picks node along the DHT routing paths
- Low quality of candidate set
- RandomSet considers 6 random nodes



Related Work II

- *Distributed Stream Processing*
  - **Astrolabe** (Cornell): Hierarchical attribute aggregation for DS management; Gossiping for faster attribute propagation
  - **Grid** (OGSA): Large-scale resource sharing
- *Distributed Query Optimization*: limited amount of distribution
- *Service Discovery*: DHT-based solutions