# An Internet-wide Distributed System for Data-stream Processing

Gabriel Parmer, Richard West, Xin Qi,
Gerald Fry, and Yuting Zhang

Boston University
Boston, MA
gabep1@cs.bu.edu

Computer Science

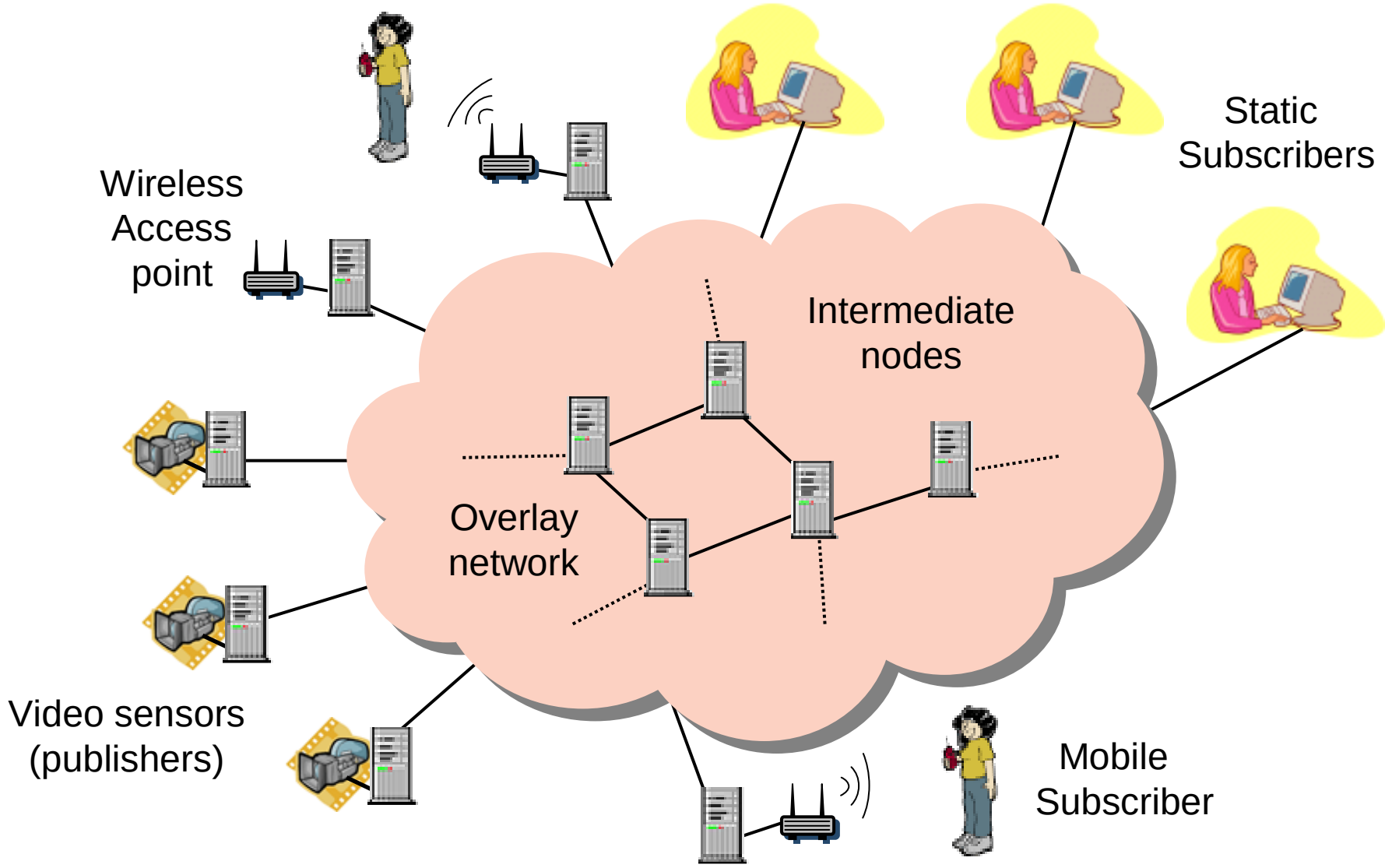
# Introduction



- Internet growth has stimulated development of data- rather than CPU-intensive applications
  - e.g., streaming media delivery, interactive distance learning, webcasting (e.g., SHOUTcast)
- Peer-to-peer (P2P) systems now popular
  - Can efficiently locate data, but not used to deliver it
- To date, limited work on scalable delivery & processing of data streams
  - Especially when these streams have QoS constraints!
- Aim: Build an Internet-wide distributed system for delivery & processing of data streams considering QoS throughout
  - Implement logical network of end-systems
  - Support multiple channels connecting publishers to 1000s of subscribers with individual QoS constraints

# A Data-stream Processing Network

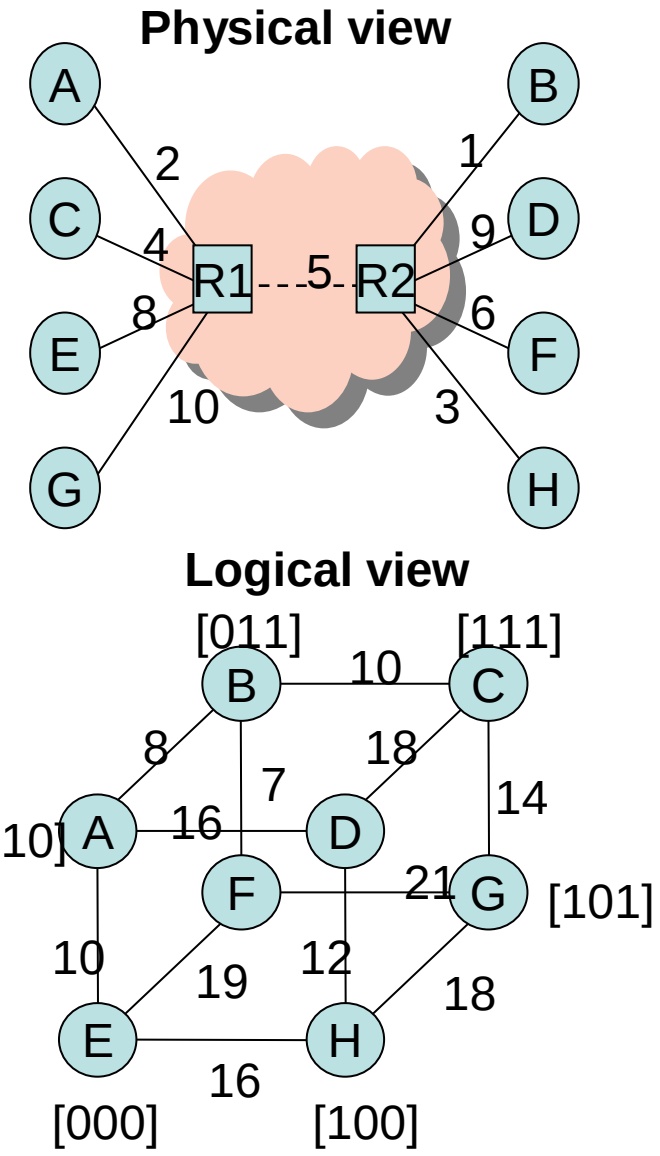




# Properties of k-ary n-cubes



- $M = k^n$ nodes in the graph
- If k = 2, degree of each node is n
- If k > 2, degree of each node is 2n
- Worst-case hop count between nodes:
  - $n\lfloor k/2 \rfloor$
- Average case path length:
  - $A(k,n) = n \lfloor (k^2/4) \rfloor 1/k$
- Optimal dimensionality:
  - n = ln M
  - Minimizes A(k,n) for given k and n



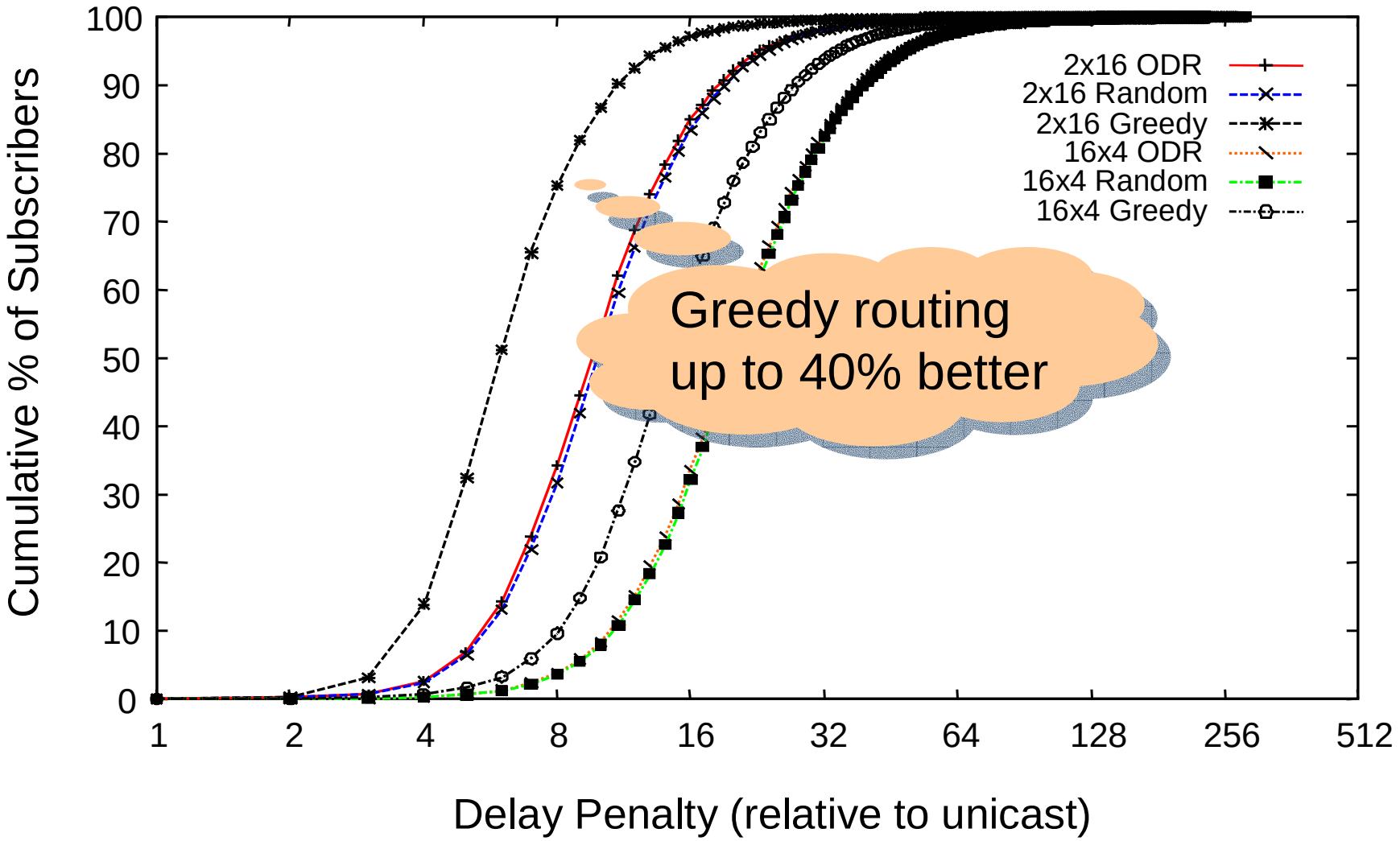
# QoS considerations in k-ary n-cubes



- Methods for considering QoS
  - Routing algorithms
    - Ordered Dimensional Routing (ODR)
    - Random Ordering of Dimensions (Random)
    - Proximity-based Greedy Routing (Greedy)
  - Dynamic node re-assignment
    - Subscribers can exchange their logical identifier with nodes that are closer to the publisher of their data-stream
      - Less hops from publishers to subscribers on average

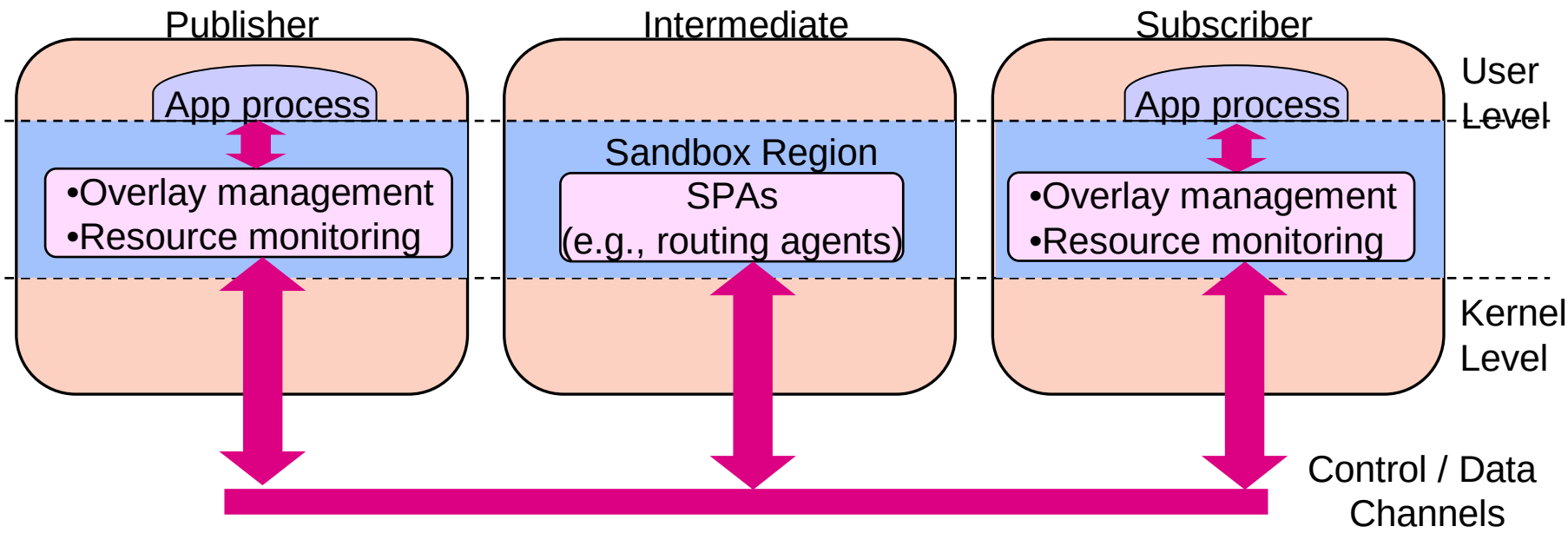
# Optimizations via routing





# End-system Architecture

Computer Science



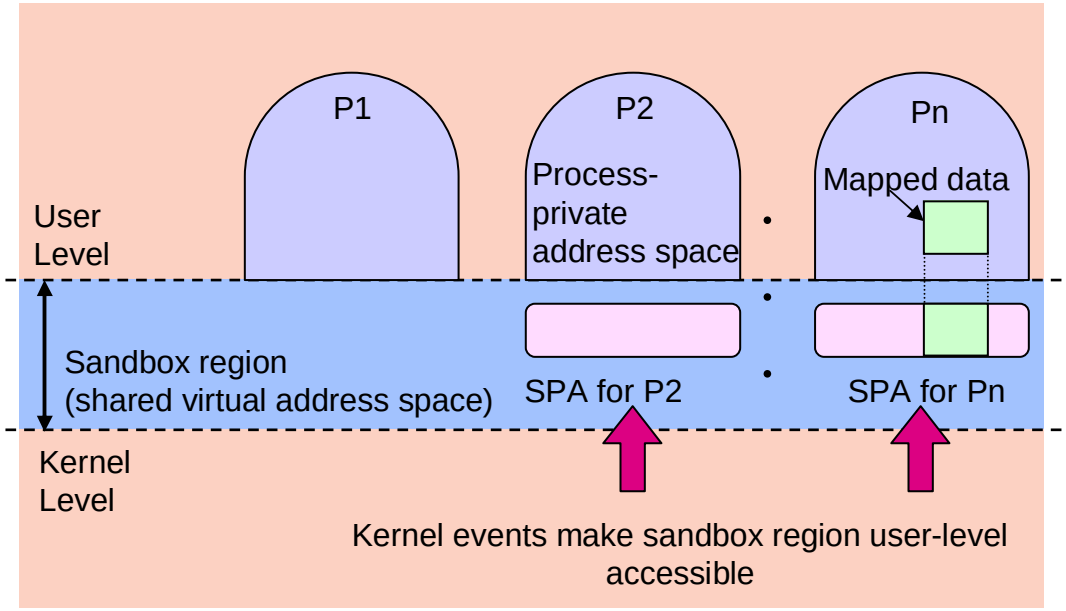

- Modify COTS systems to support efficient and predictable methods for execution of data-stream processing agents (SPAs).
  - Must consider QoS throughout, not only on the network level
- User-level sandboxing for efficient SPAs:
  - Provide efficient method for isolating and executing extensions
  - Provide efficient method for passing data between user-level and network interface (eg. by using DMA)

# User-level Sandbox Implementation



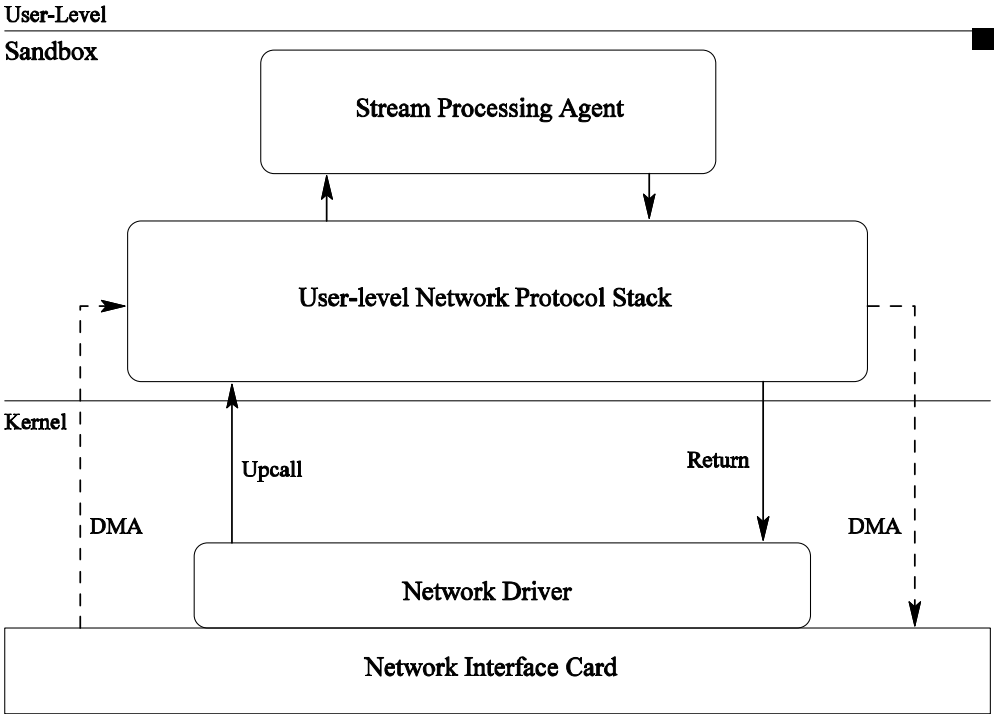
- Modify address spaces of all processes to contain one or more shared pages of virtual addresses
  - Normally inaccessible at user-level
  - Kernel upcalls to execute sandbox extensions
    - This action also flips the protection bits so sandboxed extensions always execute at user-level, thus protecting the kernel

- Can avoid address-space context switching costs when executing extensions because they exist in *all* address spaces



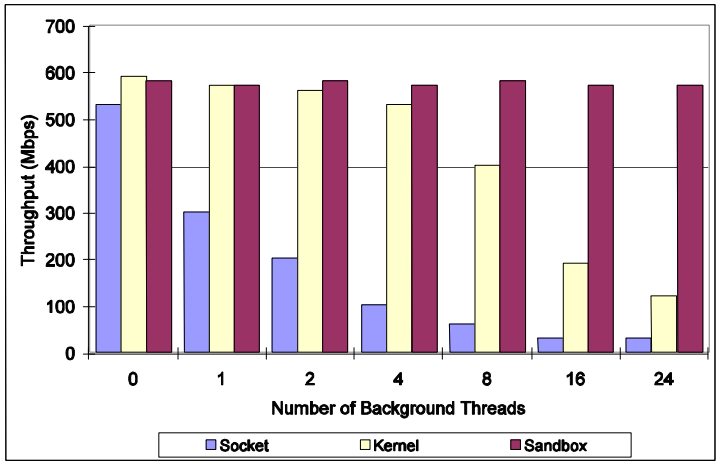
# SPA predictable execution support

Computer Science




- User-level networking stack in sandbox
  - Interacts with the NIC via DMA
  - Can execute and process at interrupt-time because sandbox is resident in *every* address space

- Elimination of extra copies allows for greater efficiency
- Interrupt-time execution allows isolation and predictability



# Conclusions



- Use ideas from overlay routing and user-level sandboxing to implement an Internet-wide distributed system
  - Provide efficient support for app-specific services and scalable data delivery
- QoS is important throughout the entire system and should be considered on the network as well as end-host level