A Semantic-based Middleware for Multimedia Collaborative Applications

Agustín J. González

Advisor: Dr. Hussein Abdel-Wahab

Doctoral Dissertation Defense

Old Dominion University

February 2000

Outline

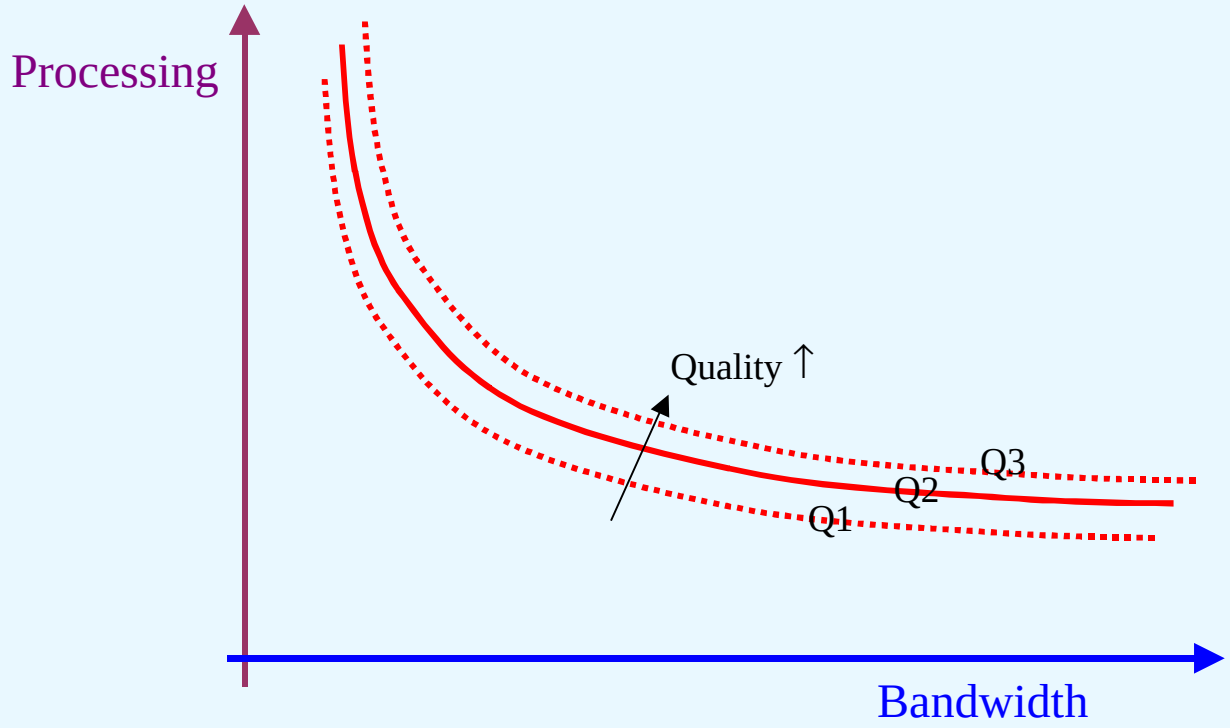
- ***Introduction***
- ***Middleware***
  - Objectives
  - Extension of operating systems network services
  - Stream synchronization
  - Floor control framework
  - Protocol for dynamic image transmission
- ***Experimental results***
- ***Conclusions***

Introduction

- ***Large-scale Multimedia Applications*** more
  - Desktop computer performance increase
  - Internet growth in bandwidth and # of hosts
- ***A challenging class of applications***
  - Processing power & bandwidth
  - Scalability more
  - Heterogeneity (Ethernet/modem, WinNT/Solaris, MPEG/H263)
  - Timely data delivery
- ***Traditional services***
  - Network layer: UDP & TCP (real time was not a concern)
  - Operating systems: Abstractions are not adequate for multimedia.
    - Example: Real time is not well supported.
- ***Gap between multimedia requirements and system services*** more

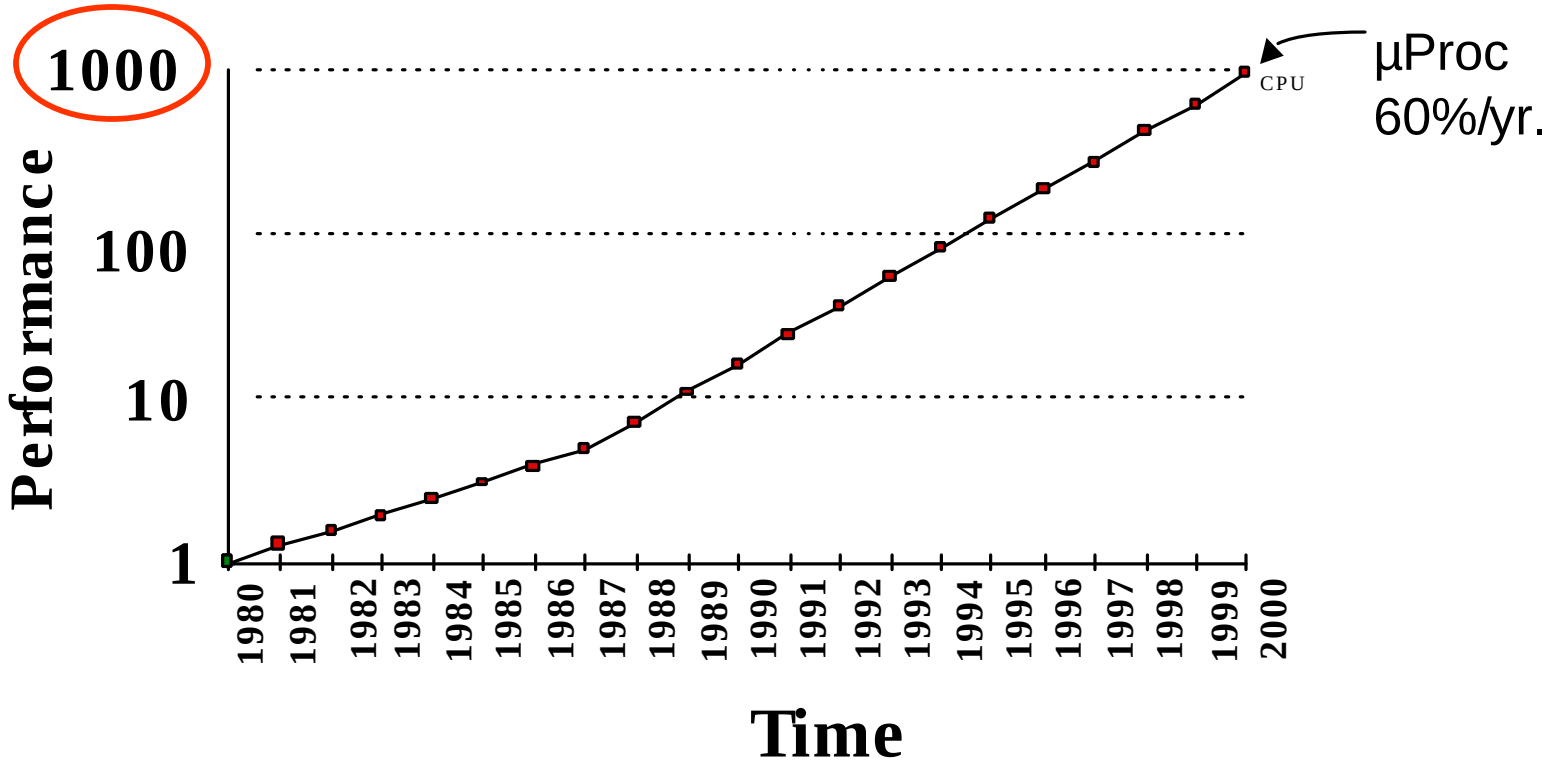
Outline

Multimedia Resource Requirements



CPU performance

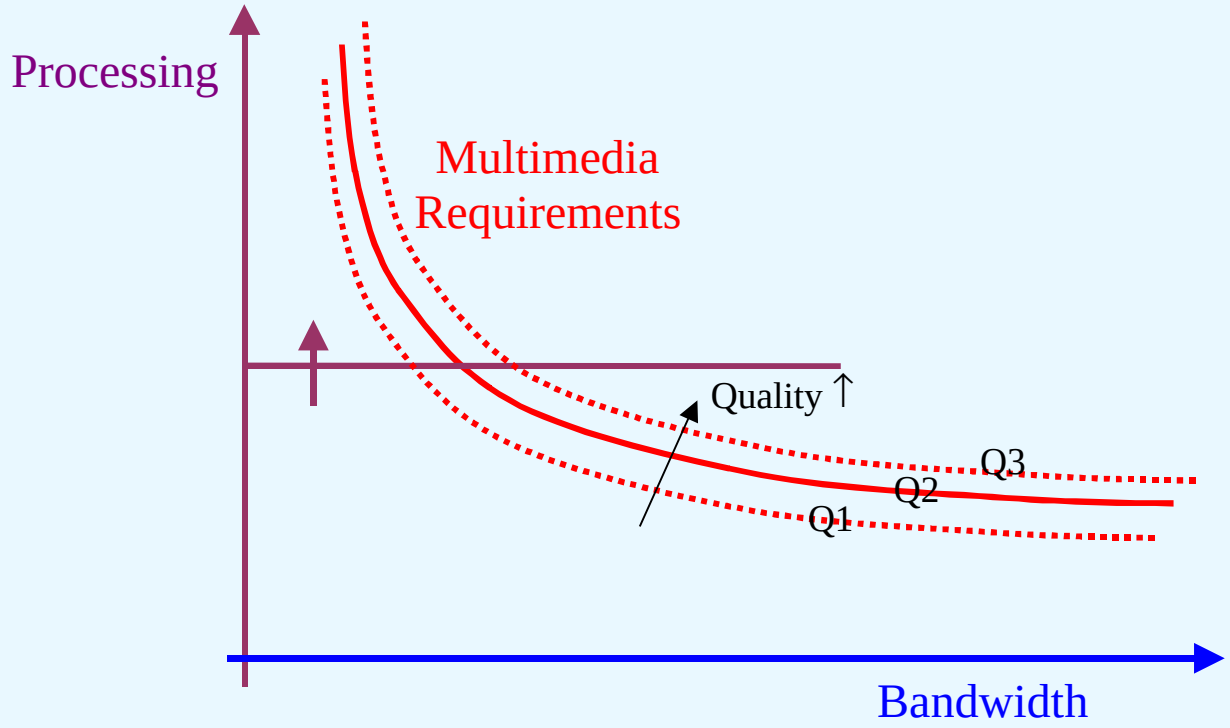
Processor Performance Increase



Source: Dr. David Patterson University of Virginia Distinguished Lecture Series, May 19,1998. http://www.cs.berkeley.edu/~pattrsn/talks/Stanford.pdf

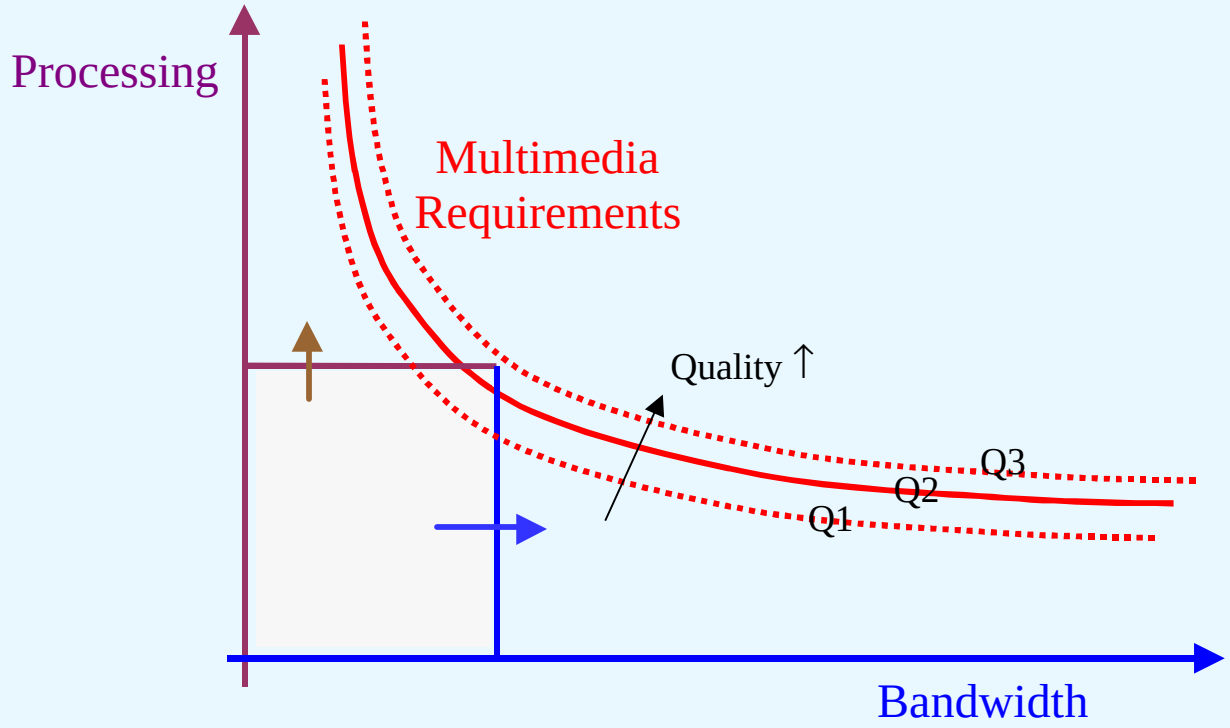
Effect on MM

Multimedia Resource Requirements



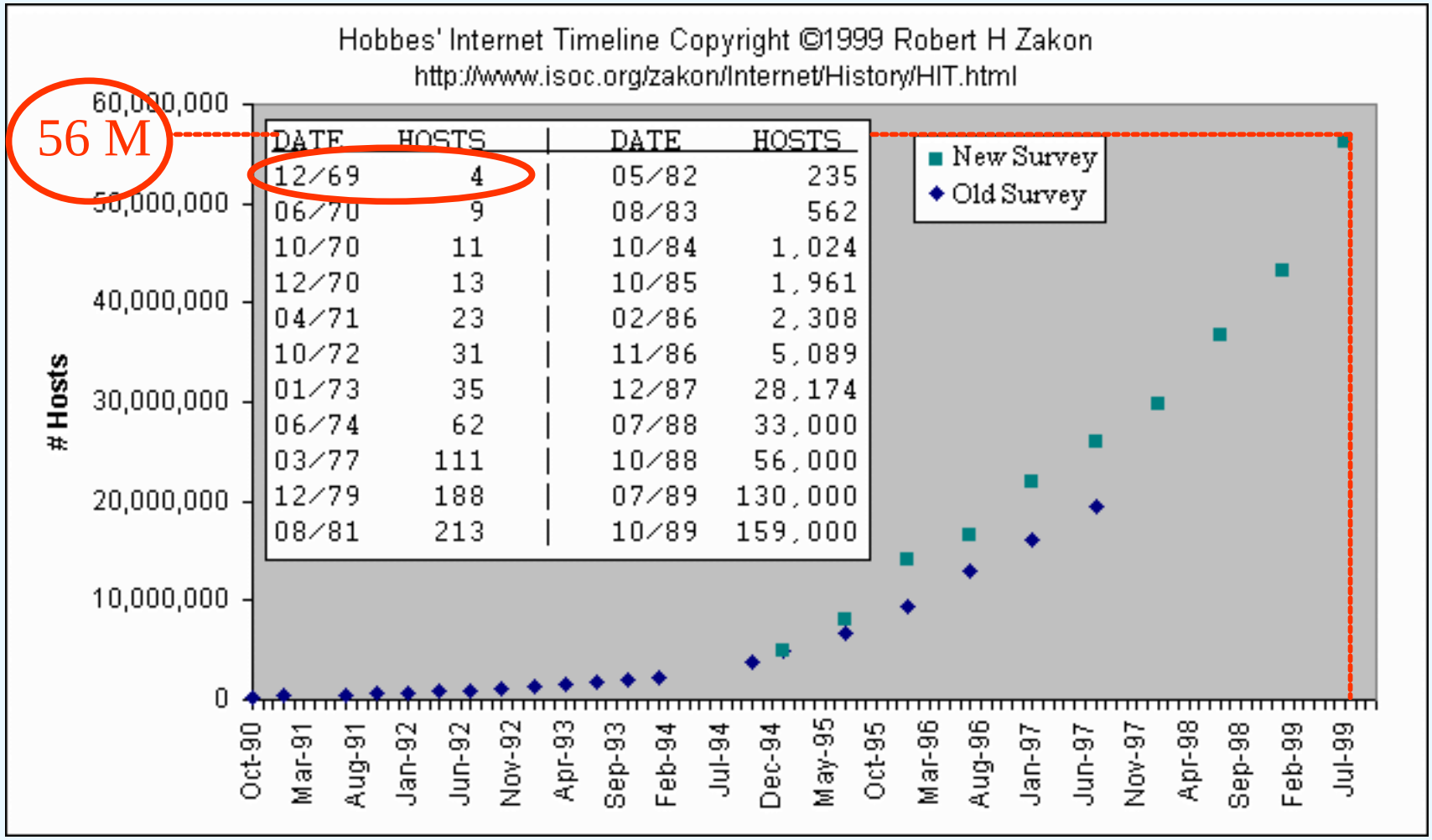
Bandwidth

Multimedia Resource Requirements



High processing + high bandwidth + Others

Internet Growth



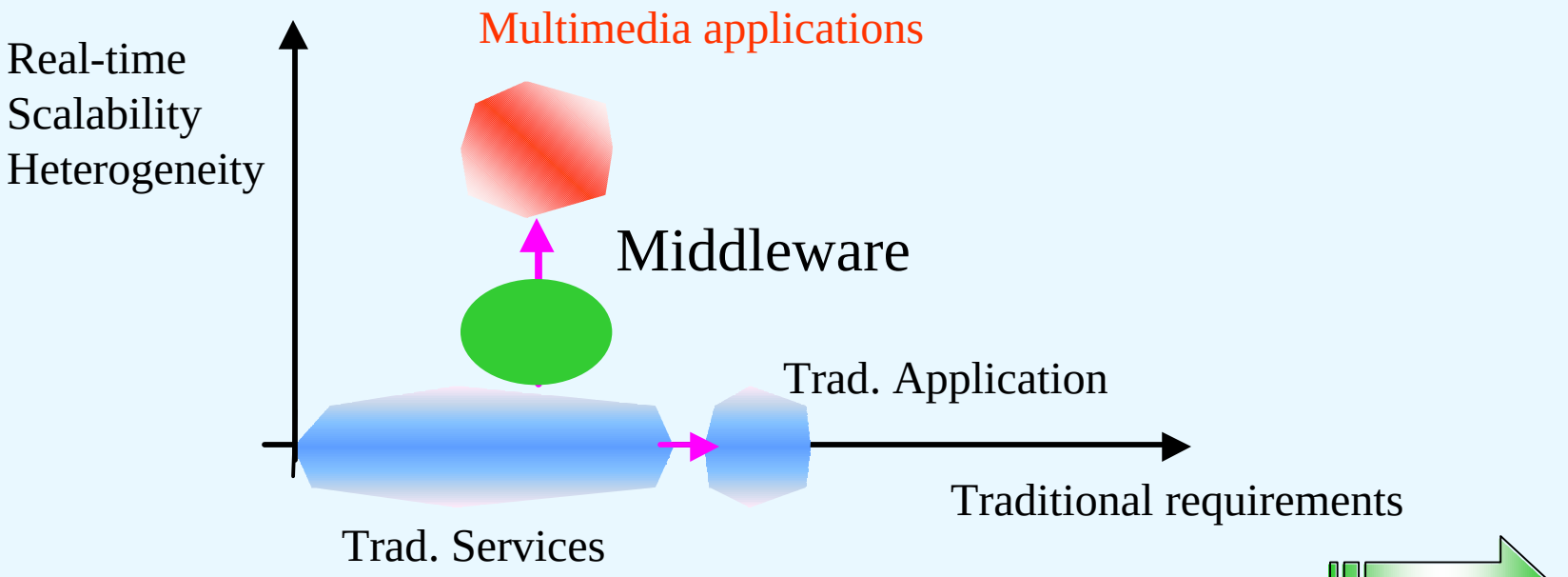
DATE	HOSTS	DATE	HOSTS
12/69	4	05/82	235
06/70	9	08/83	562
10/70	11	10/84	1,024
12/70	13	10/85	1,961
04/71	23	02/86	2,308
10/72	31	11/86	5,089
01/73	35	12/87	28,174
06/74	62	07/88	33,000
03/77	111	10/88	56,000
12/79	188	07/89	130,000
08/81	213	10/89	159,000

Introduction

Gap between system services and application requirements

⇩

Developers need to fill this gap by implementing common services for multimedia applications.



outline

Objective

"Our main objective is to investigate and propose heterogeneous, scalable, reliable, flexible, and reusable solutions and enhancements to common needs in developing multimedia collaborative applications."

Needs we addressed:

* Extension of network services
* Media synchronization
* Floor control
* Data sharing

Outline

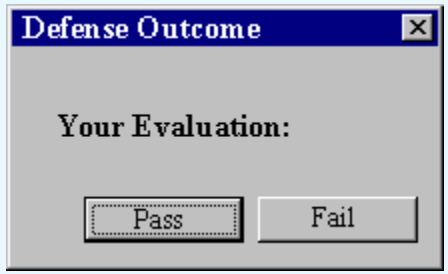
Extension of Network Services

- ***New services***
  - Asynchronous data reception more
  - Quality of service monitoring more
  - Transmission traffic rate control more
- ***New convenient facilities***
  - Unified Multicast/Unicast API more
  - Efficient buffer management for **A**pplication **D**ata **U**nit more

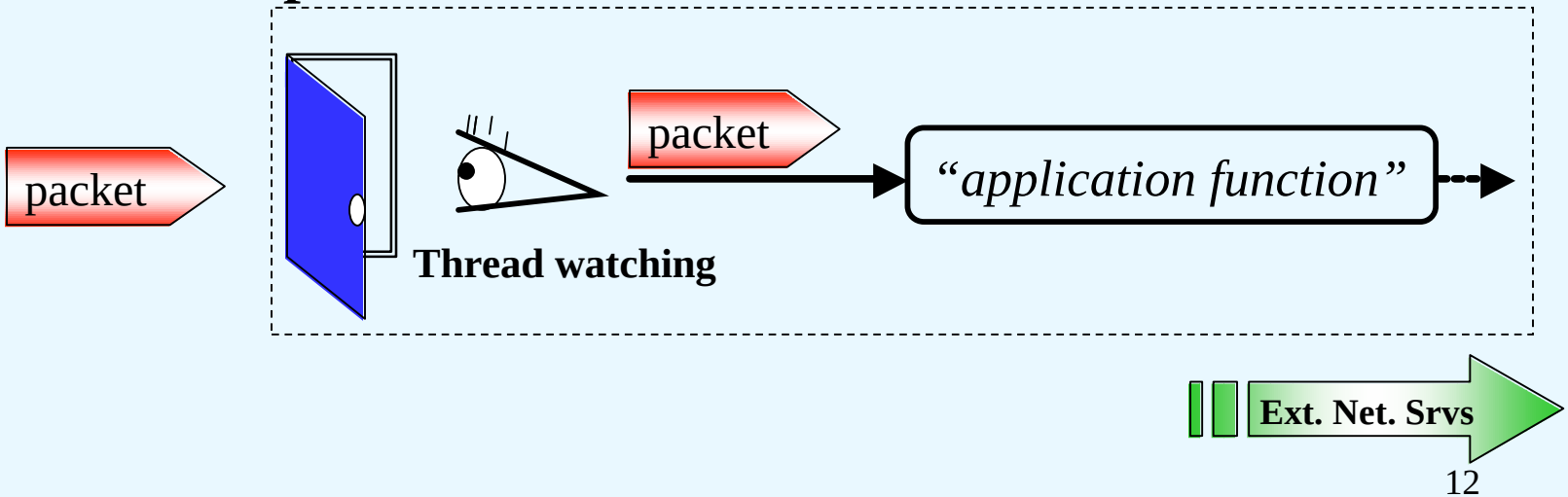
Outline

Asynchronous data reception

Event-driven model

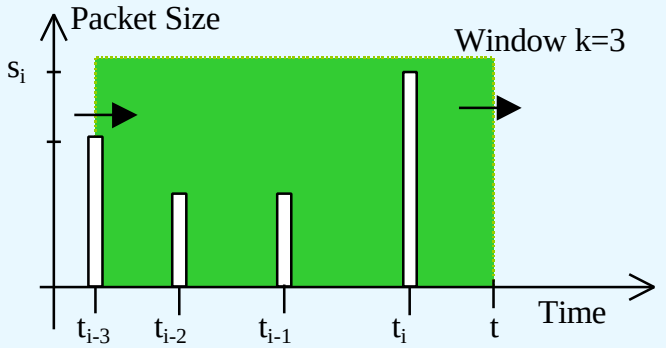


Network packet arrivals

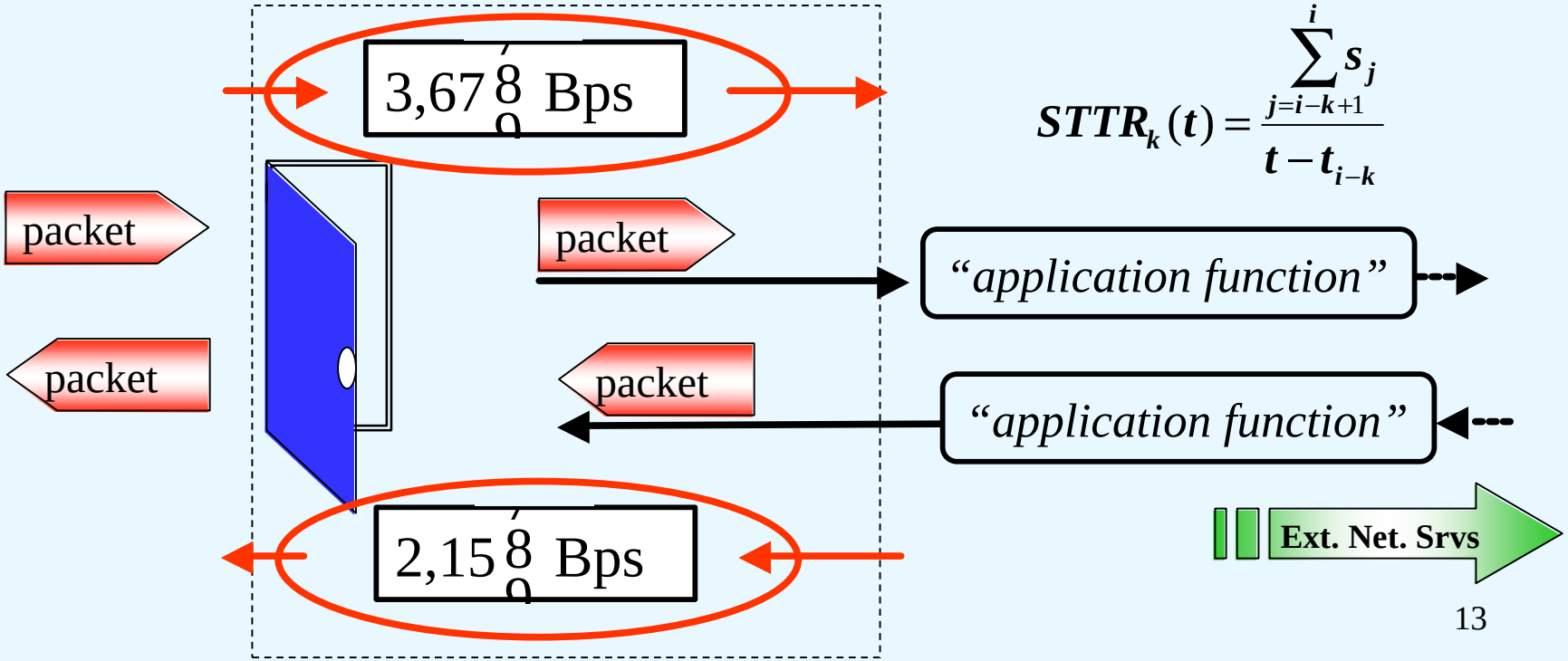


Quality of services monitoring

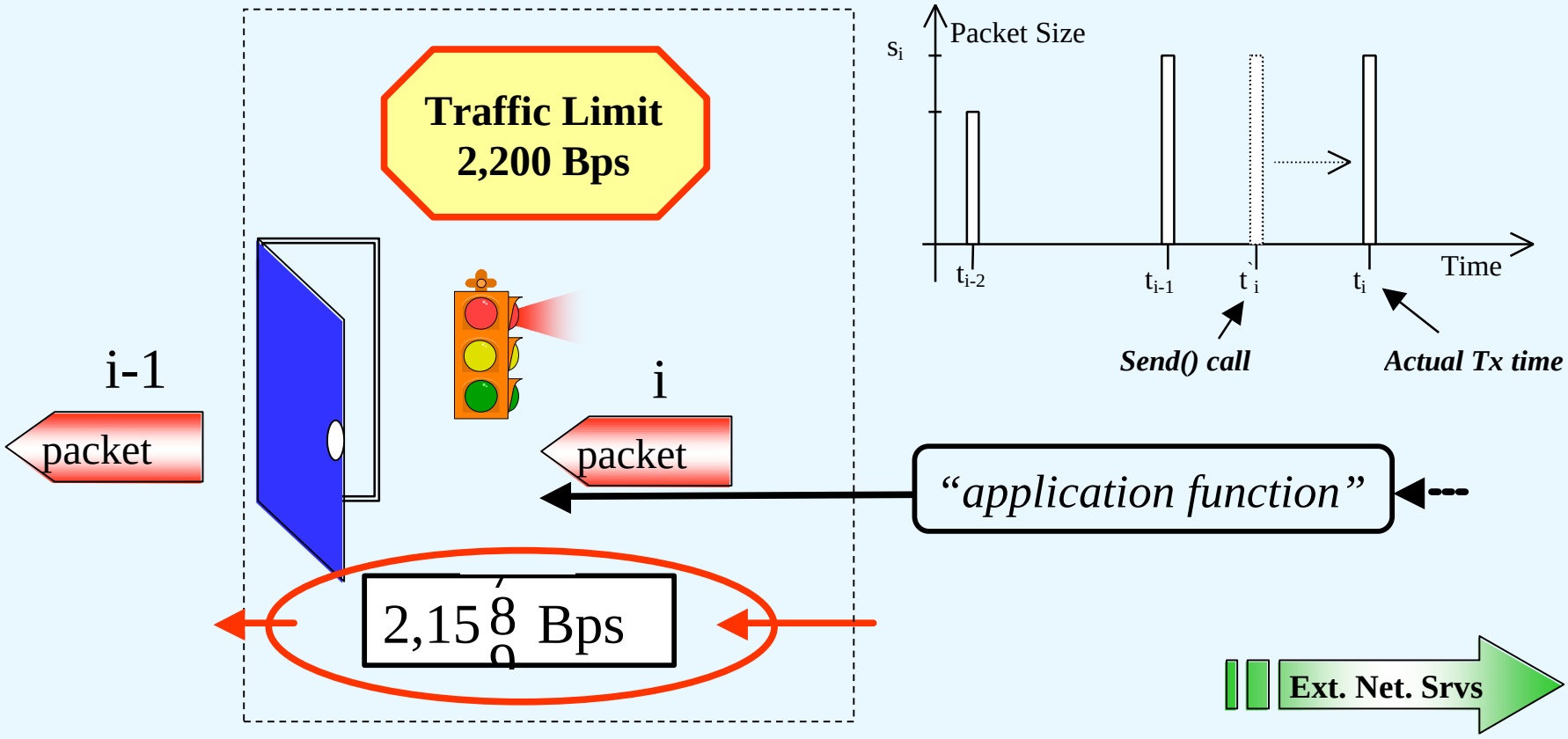
Traffic monitoring



$$STTR_k(t) = \frac{\sum_{j=i-k+1}^{i} s_j}{t - t_{i-k}}$$



Transmission traffic rate control



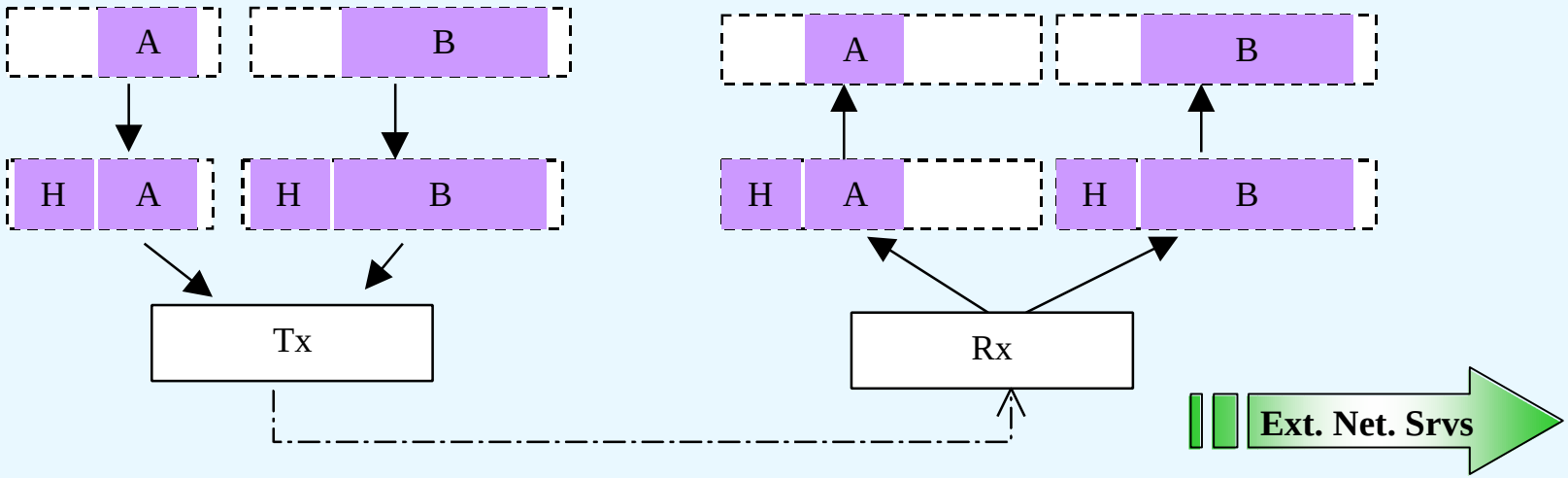
Unified Multicast/Unicast API

$$\left(\begin{Bmatrix} \text{Multicast address} \\ \text{Unicast address} \end{Bmatrix}, \quad \{\text{port}\} \right)$$

- ***Datagram transmission***
  - A *send* to a machine or a multicast group does not make a difference.
- ***Datagram reception***
  - if the given IP address is a multicast, join group.
  - if address is not multicast, do not bind (I'm client).

Efficient buffer management for **A**pplication **D**ata **U**nit

* Goal: to prevent payload movements in memory
* Sender modules create an output buffer that can hold following “headers” and “tails” **.**
* Receiver module needs to allocate worst case buffer size.



Stream Synchronization

- ***Problem: processing times and network delays are not deterministic.***
- ***The objective of synchronization is to faithfully reconstruct the temporal relationship between events ("pieces of data").***

- ***Main characteristics of our solution:***
  - It depends on one-way messages only
    - No need of feedback
  - It only requires sender's and receivers' clock rates to be constant.
    - These clocks might be off.
    - These clocks might even have different rates of change.
    - No need of globally synchronized clocks
  - It supports policies to handle late packets and delay adjustments.

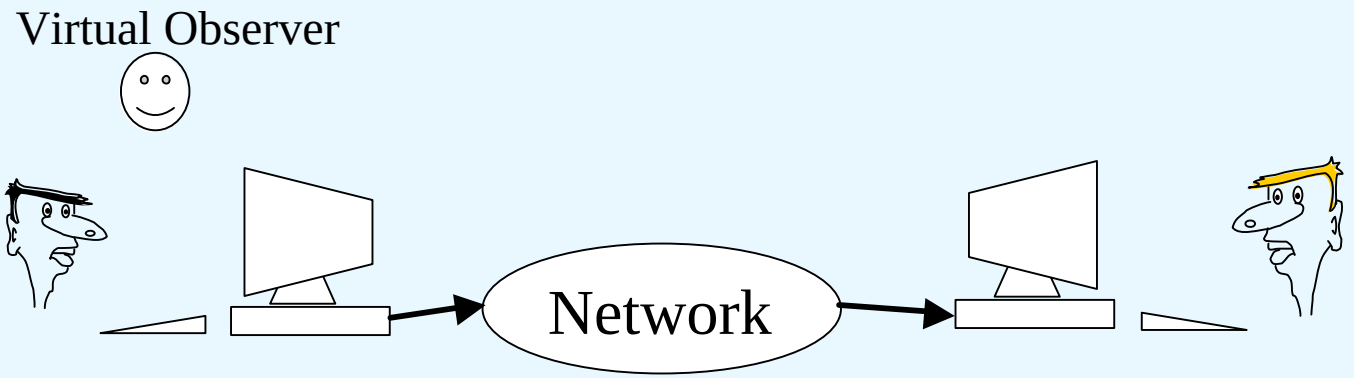
Details

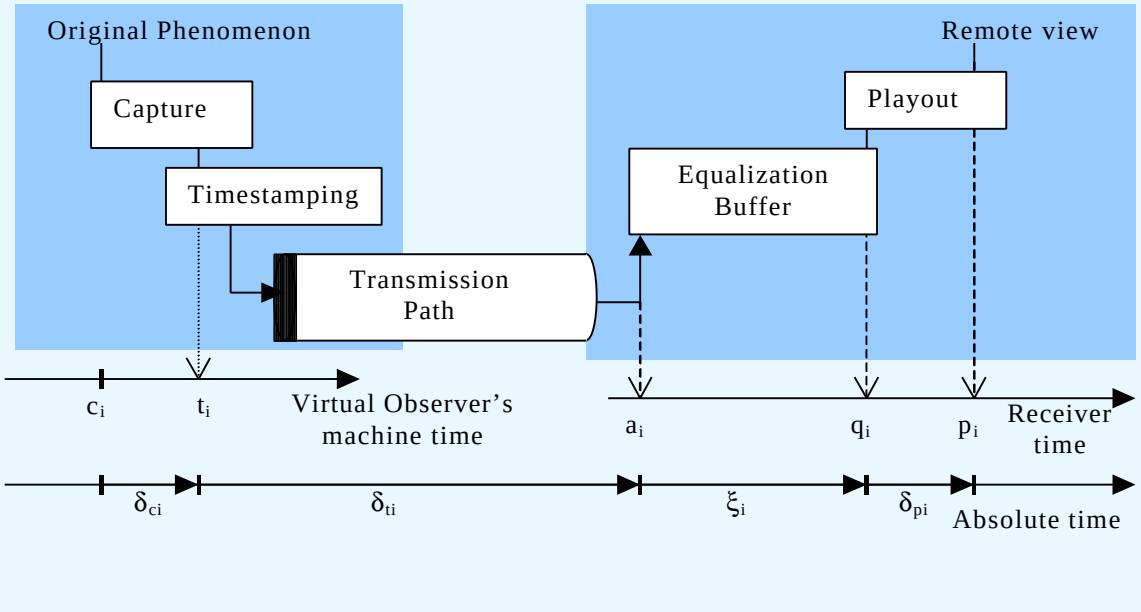
Stream Synchronization (details)

- ***Time model*** more
- ***Intra-stream synchronization*** more
- ***Inter-stream synchronization*** more
- ***Clock skew estimation and removal*** more

Outline

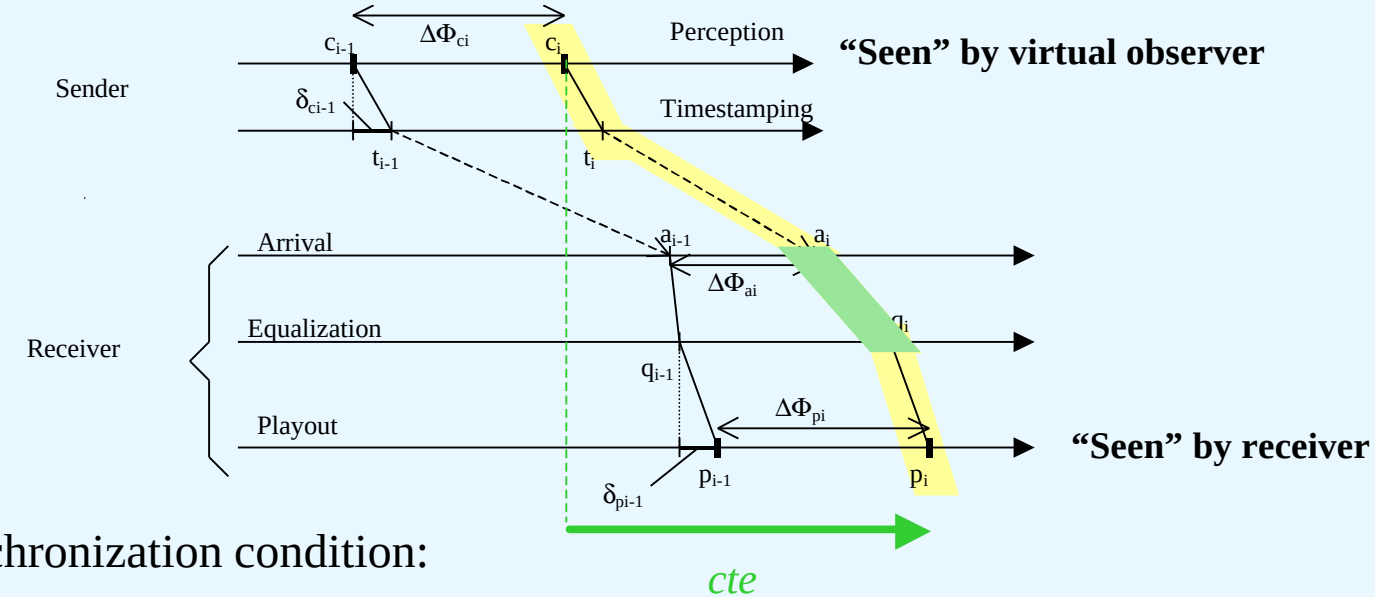
Time Model





Sync

Intra-stream Synchronization (model)

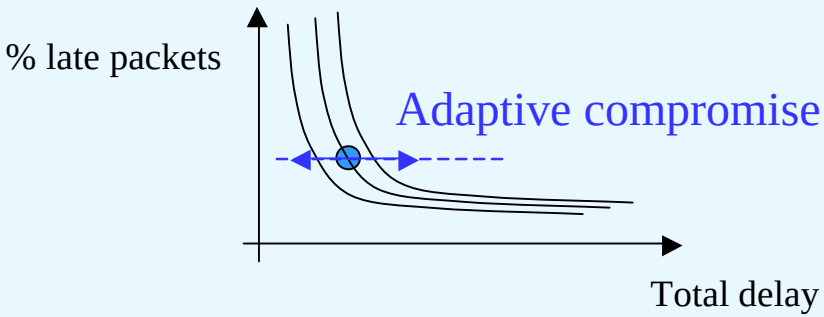


Synchronization condition:

$$p_i = c_i + cte$$

Virtual delay

Tradeoff:



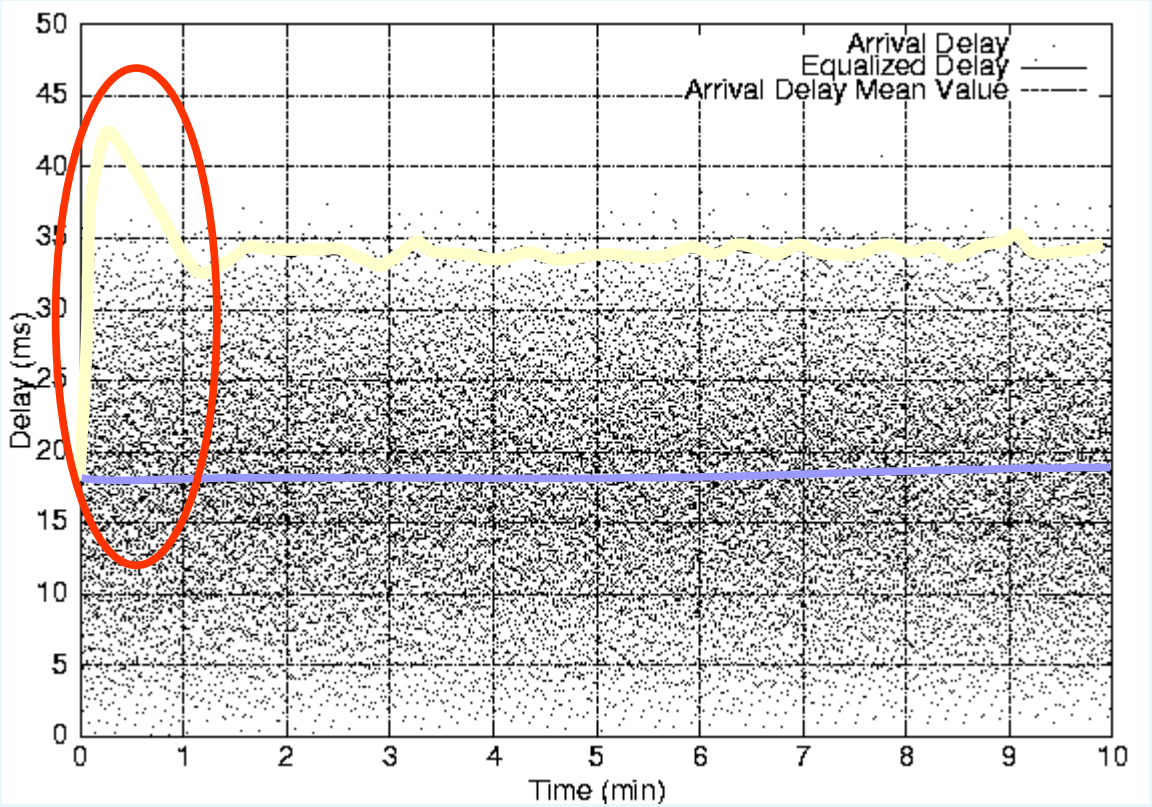
Solution

Intra-stream Synchronization (solution)

Adjust “virtual delay” to achieve a given % of late packets

Estimator for % of late packets: $$l_i = a\, l_{i-1} + (1-a)*\begin{Bmatrix} 1 & \text{for late arrival} \\ 0 & \text{otherwise} \end{Bmatrix}$$

NASA MBone 1% late packets

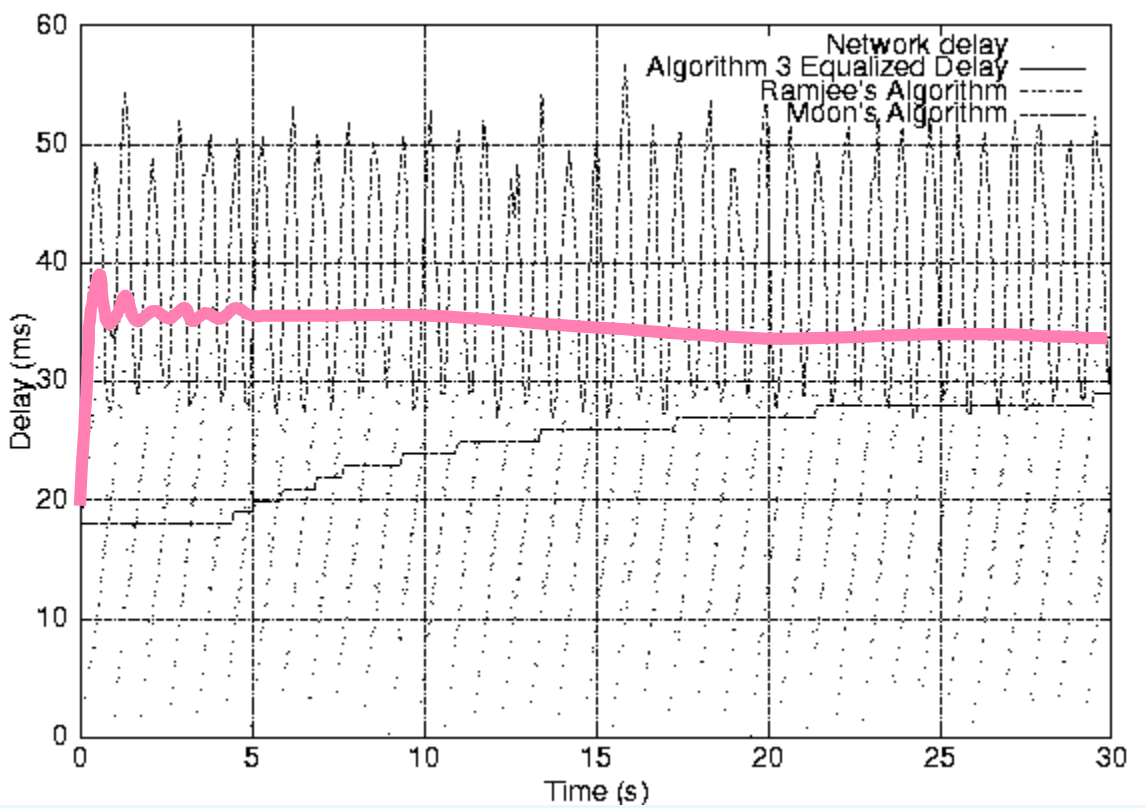


Slow start
1 min !

more

sync

Fast start refinement

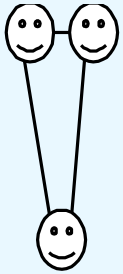


Less than 5 s !

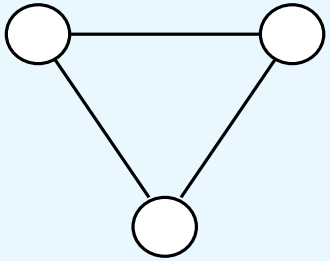
sync

Inter-stream Synchronization

Global synchronization model v/s Differentiated synchronization model

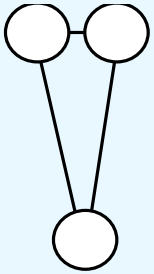


Actual network delay



Global Sync Model

Synchronizes streams coming from anywhere with worst case delay

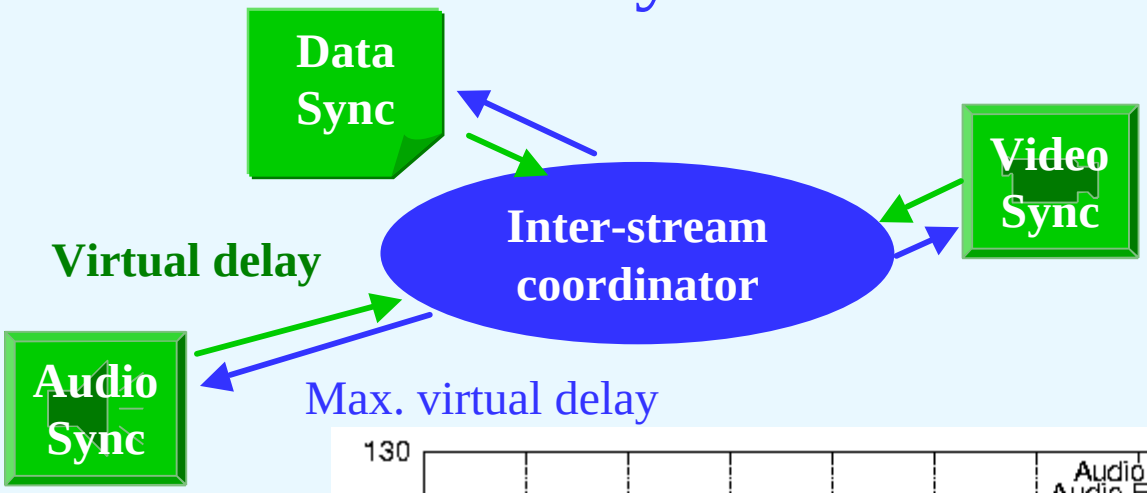


Differentiated

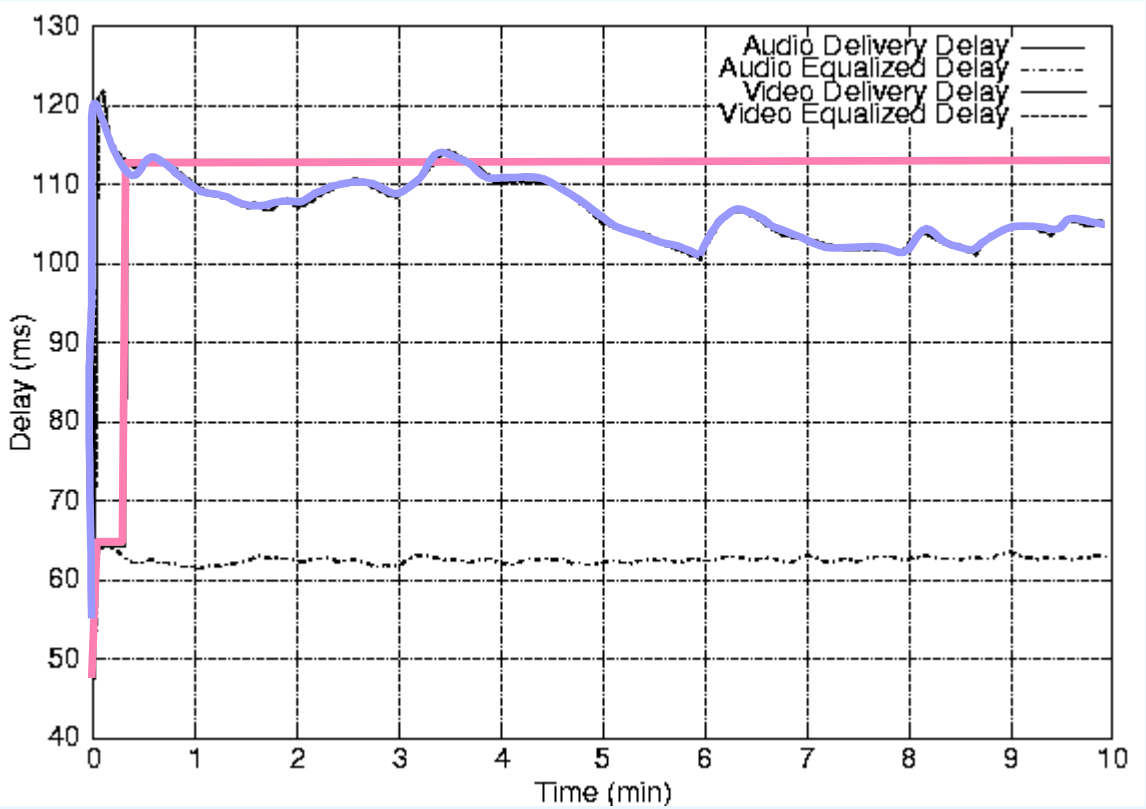
Synchronizes streams coming from one virtual observer

Solution

Inter-stream Synchronization (solution)



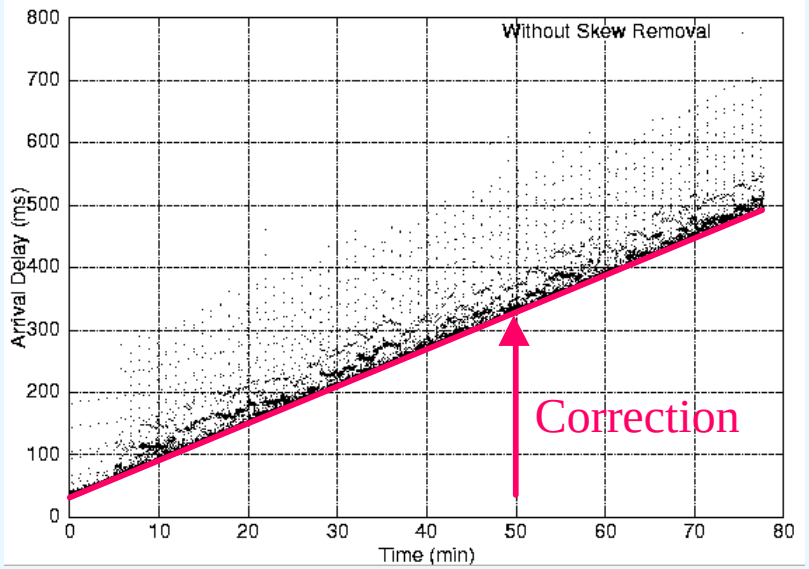
Max. virtual delay



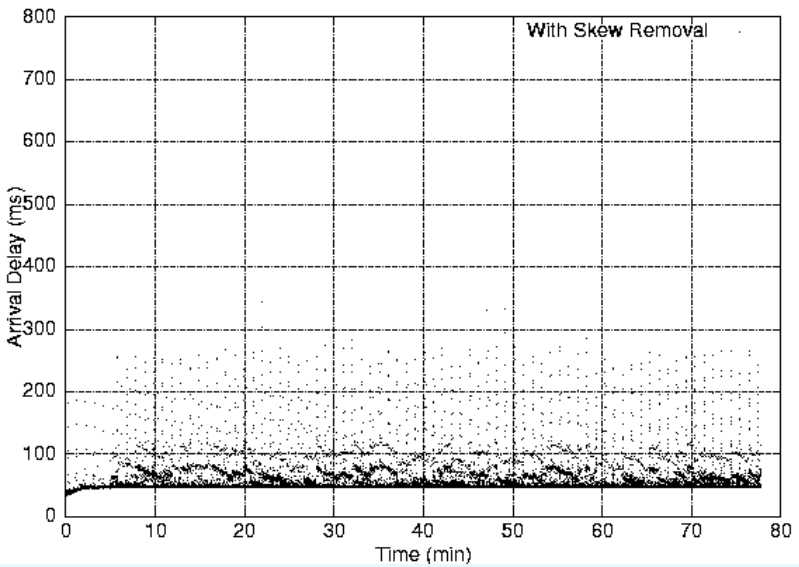
sync

Clock skew estimation and removal

Goal: Remove differences in clock frequencies



Before



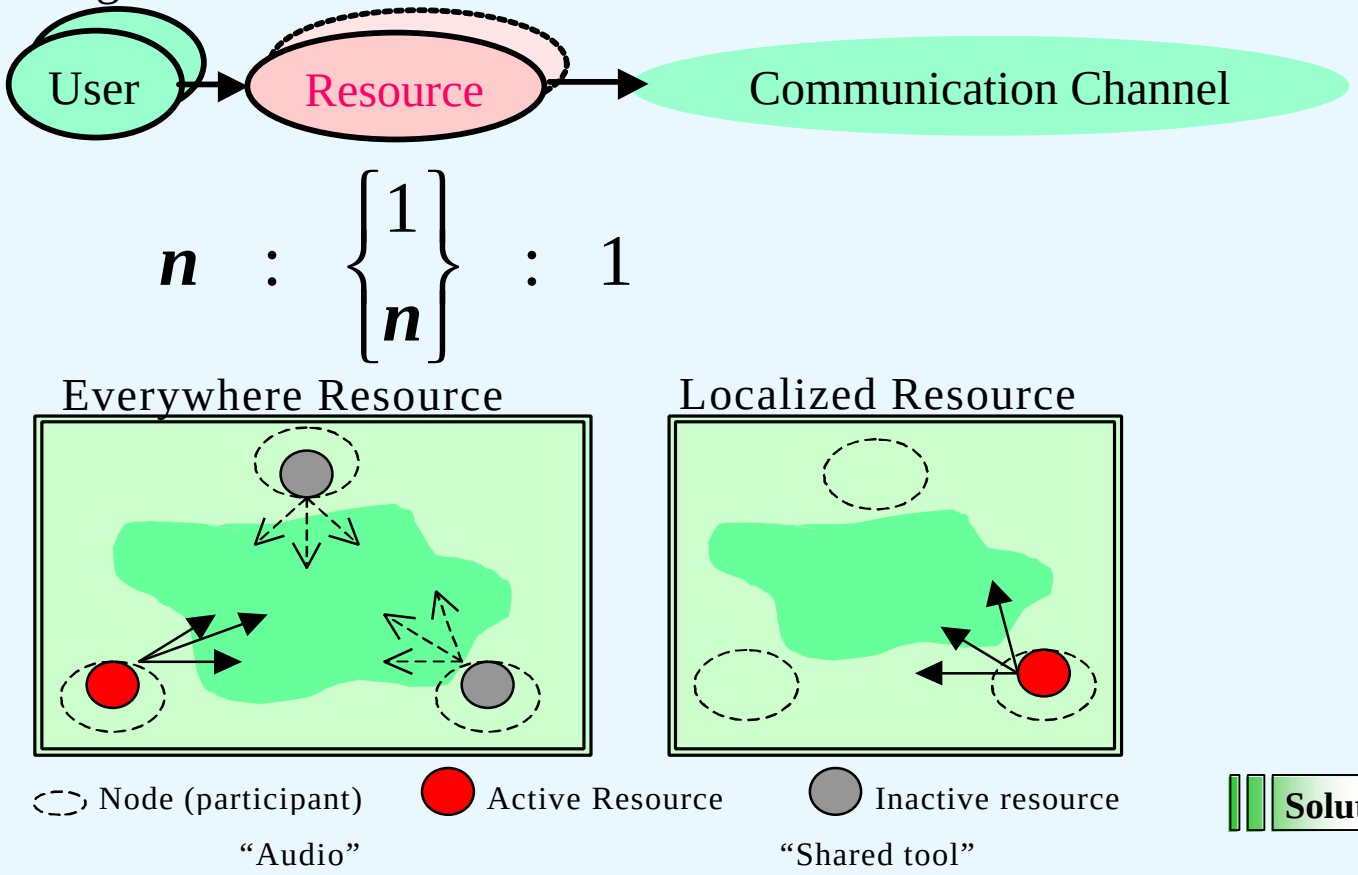
After

The algorithm adjusts a straight line as new packets arrive

sync

Lightweight Framework for Floor Control

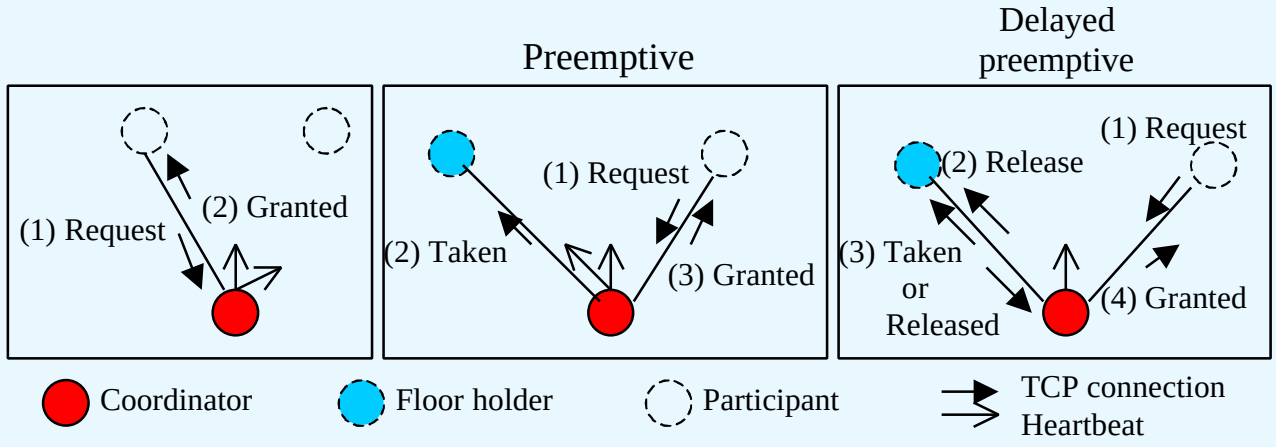
- Problem: How to manage exclusive resources in large-scale multimedia applications?
- We recognize two cases:



Solution

Floor Control (Solution)

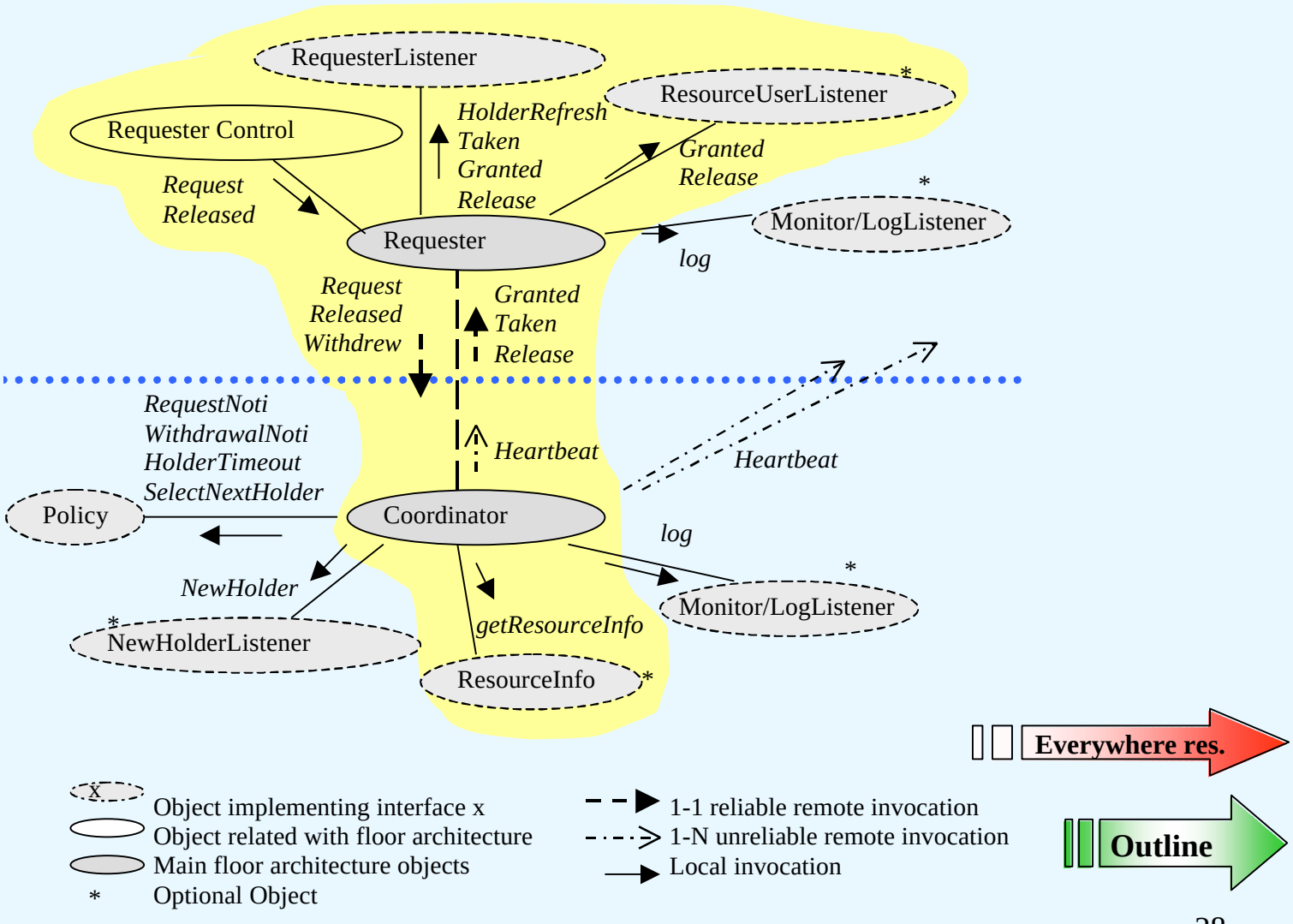
* We propose two protocols for floor control, one per architecture.
* Features: lightweight, scalable, robust



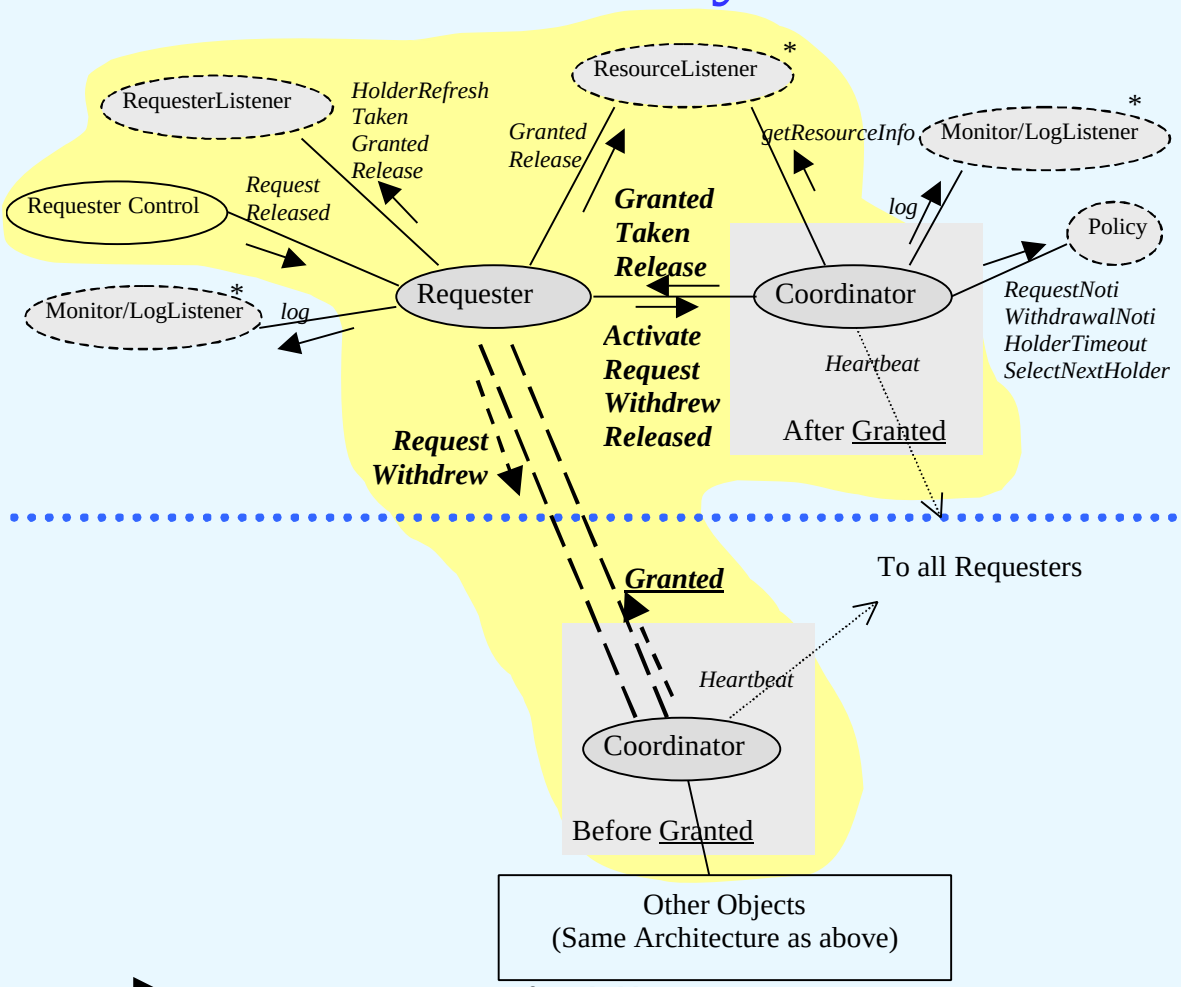
  * The coordinator is stationary for localized resources.
  * The coordinator migrates with floor for everywhere resources

Localized res.

Architecture for localized resources



Architecture for everywhere resources



1-1 **Temporary** connection for reliable remote invocation

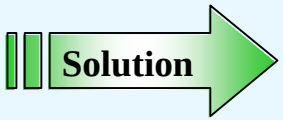
1-N unreliable remote invocation

Local invocation

Outline

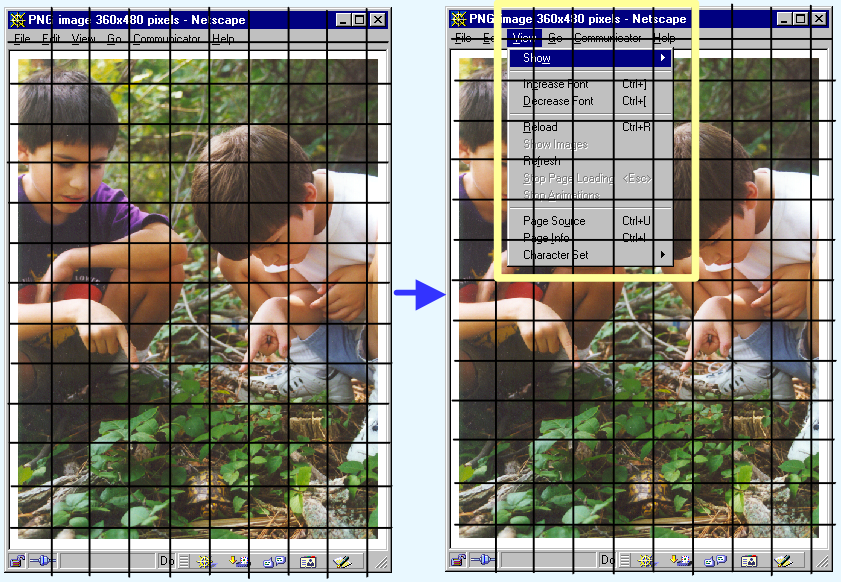
Protocol for Dynamic Image Transmission

- **Problem**: In addition to audio and video, multimedia sessions needs a component to convey the main idea of discussion.
- Traditional solutions:
  - Use video (size limitation & high bandwidth)
  - Shared tools: XTV, co-browsers, VNC,.. (do not scale well)
- Our solution:
  - **Video-like protocol tuned to send dynamic images**

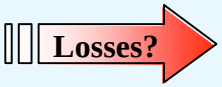


Protocol for Dynamic Image Transmission

- Sender:
  - * *Temporal redundancy removal*
    - » Sample image at regular period
    - » Divide image in tiles
    - » Process only changed tiles
  - * *Spatial redundancy removal*
    - » compress and send changed tiles



- Receiver:
  - » Receive data unit
  - » Decompress tile
  - » Update tile in image



Overcoming losses

- Each tile is retransmitted after a random time
- This also accommodates late comers

Performance Study

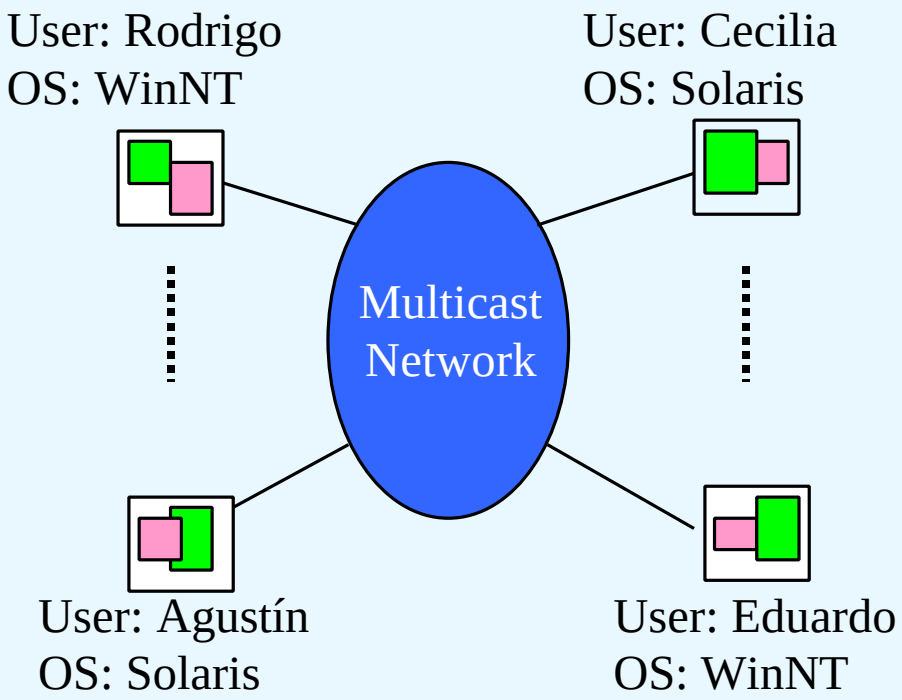
* How to select a tile compression technique? (JPEG, GIF, PNG?)
* Is there a “best” tile size? What does it depend on?
* How often to sample the image?
* How can two tiles be compared efficiently?
* Maximum data transmission rate? What does it depend on?

Outline

Implementation and Experimental Results

- Implementation:
  - Network support implemented
  - Synchronization: implemented and used with real RTP data in off-line analysis
  - Floor control: partially implemented for localized resources
  - Image protocol implemented
- Putting everything together: Odust
  - A prototypical sharing tool built on top of the middleware.
    It uses:
    - Network support, floor control, dynamic image protocol, other application specific modules.

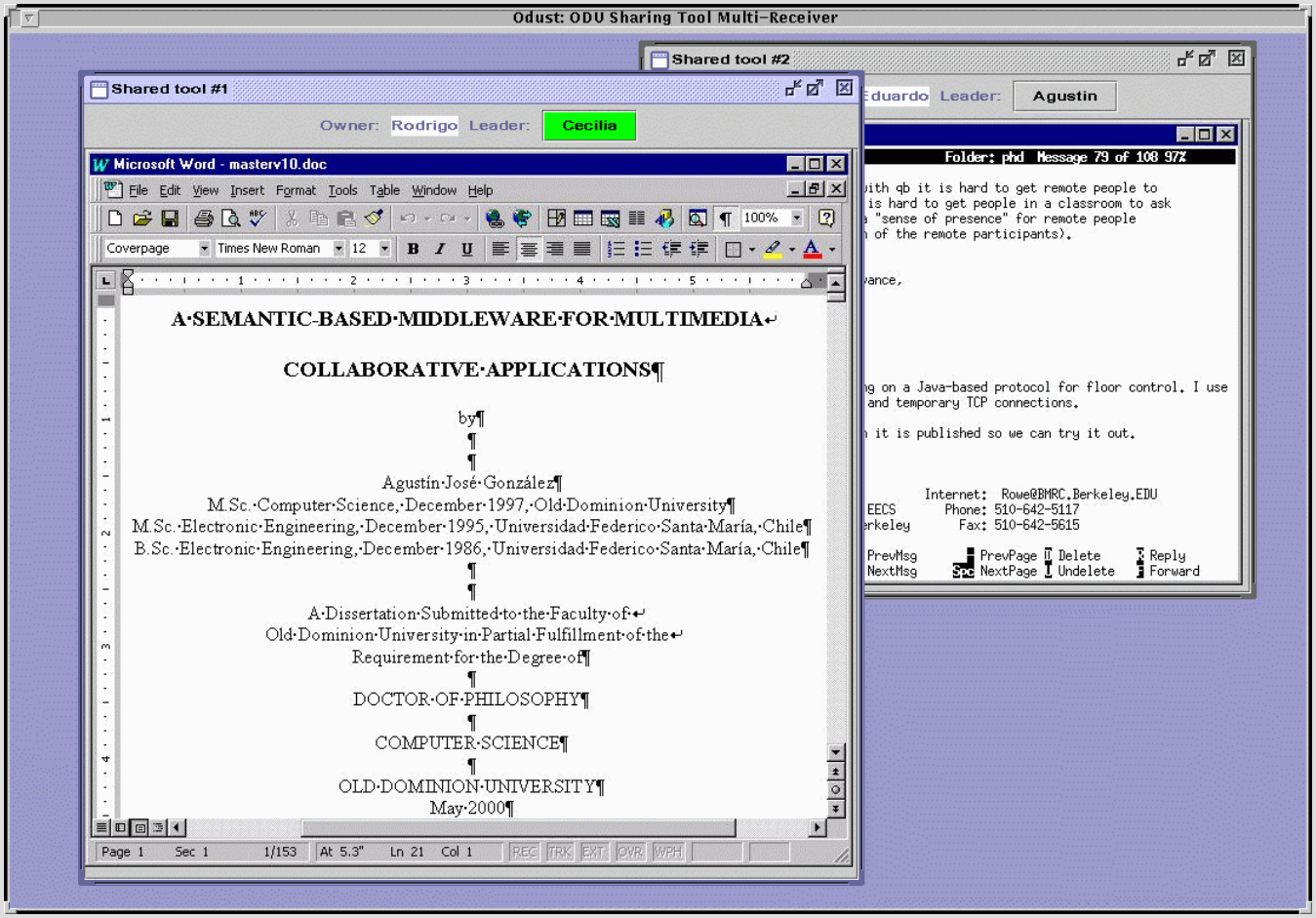
Odust Description

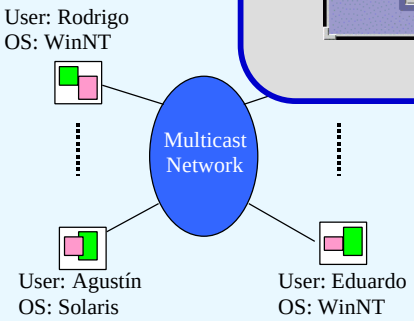


Architecture

Odust Description: Cecilia's view

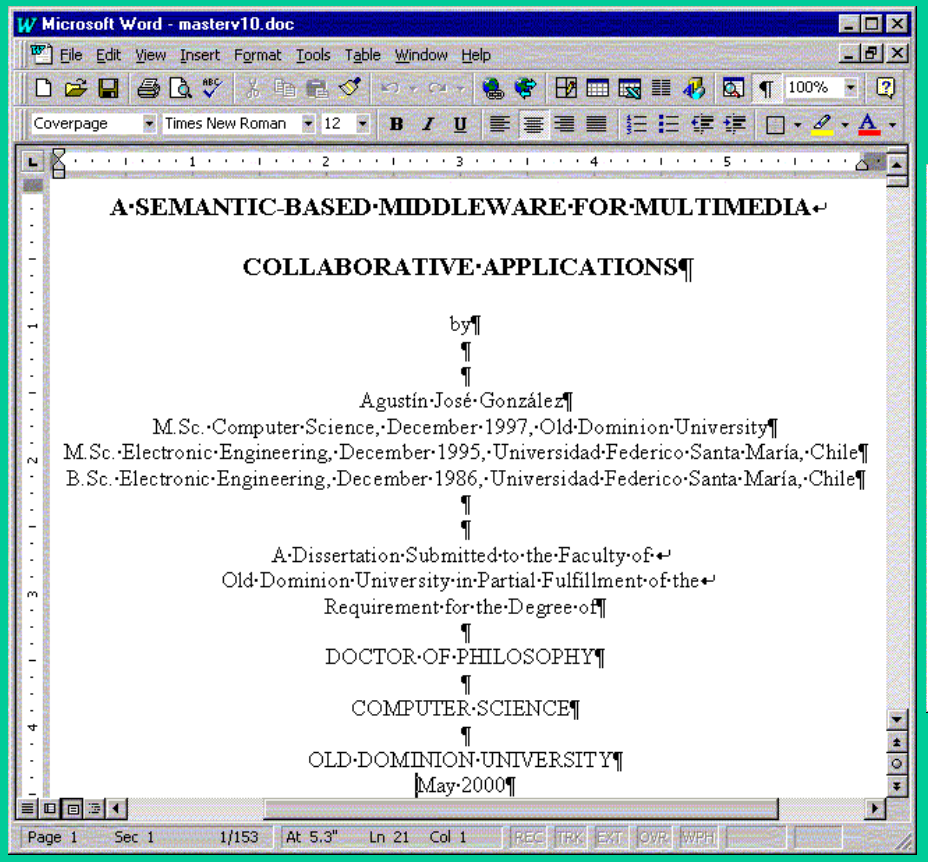
UNIX



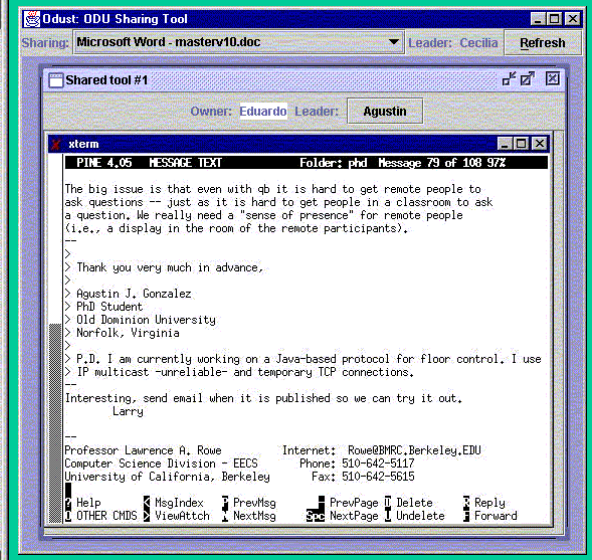


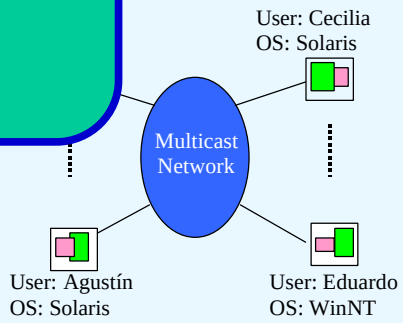
Architecture

Odust Description: Rodrigo's view

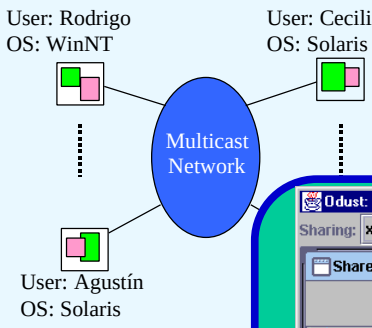


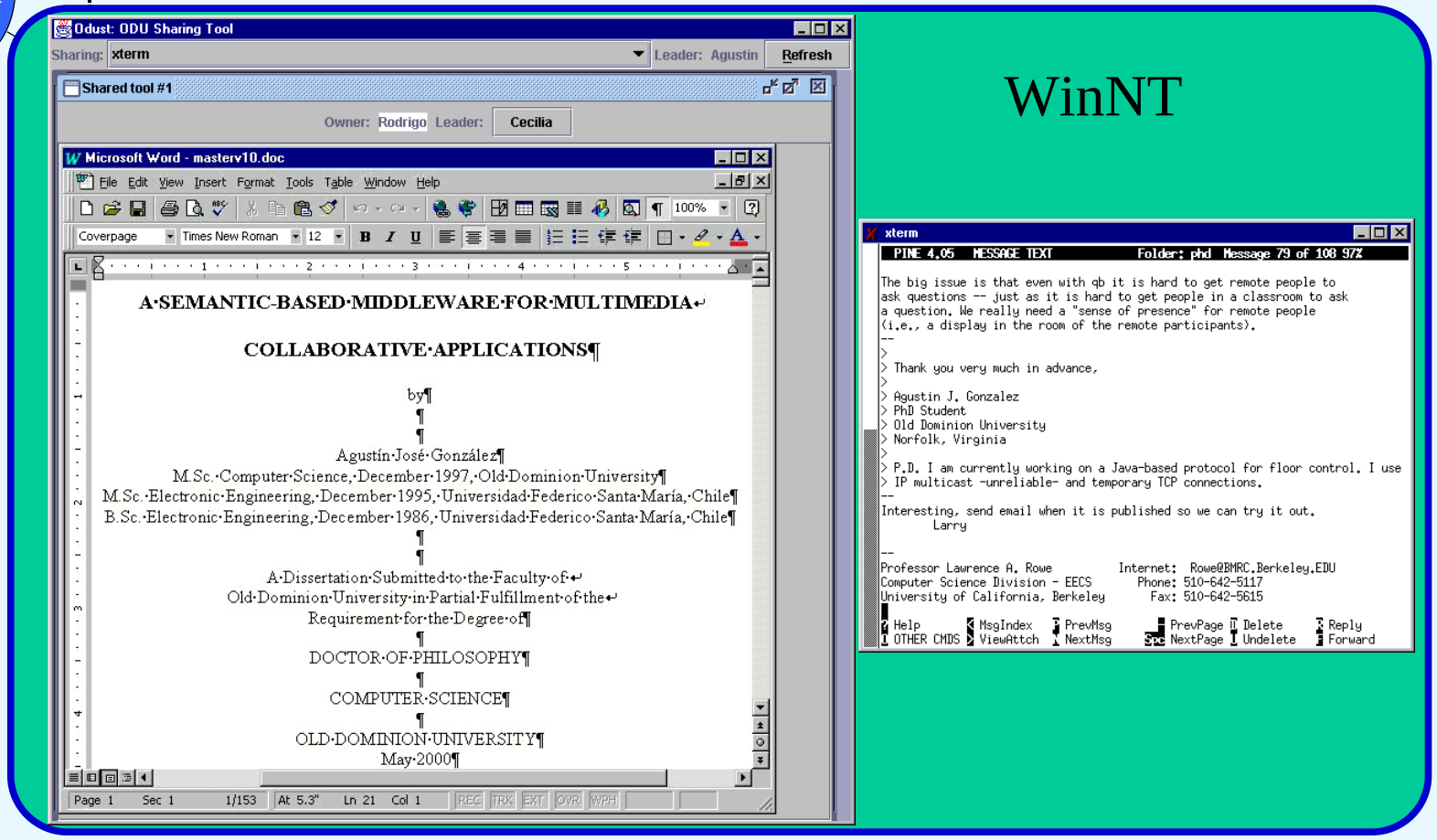
WinNT



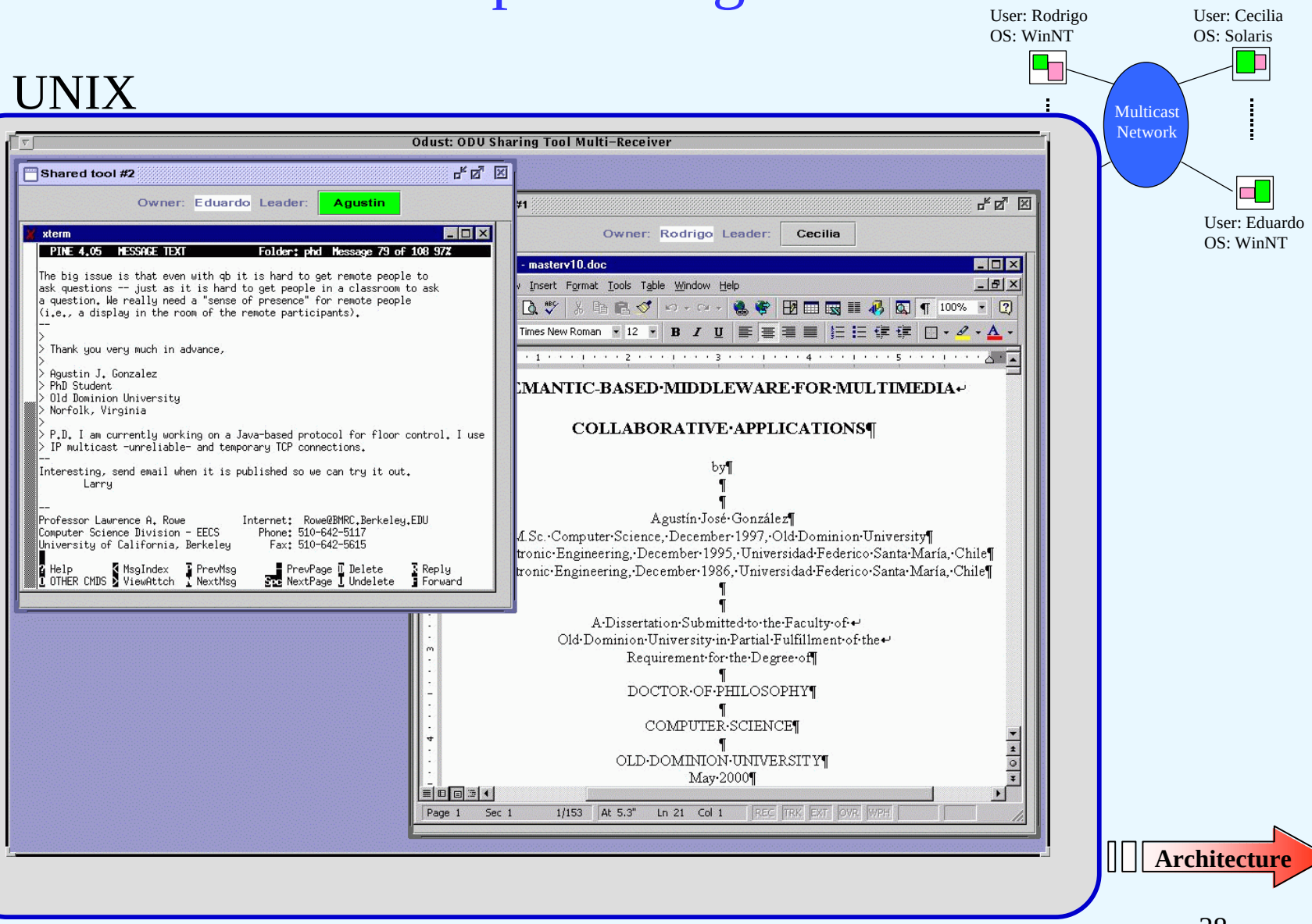


Odust Description: Eduardo's view



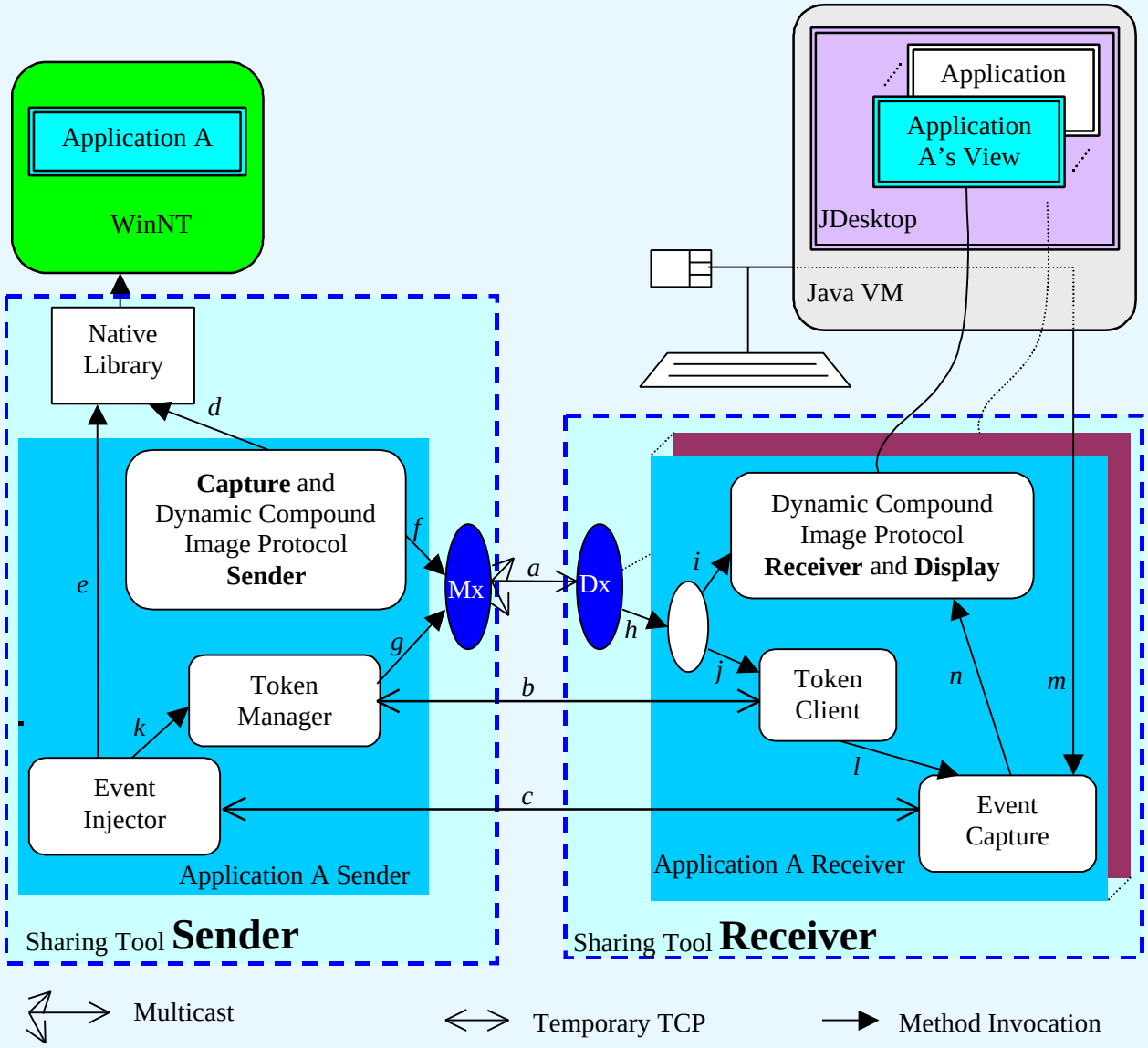


Odust Description: Agustín's view



Architecture

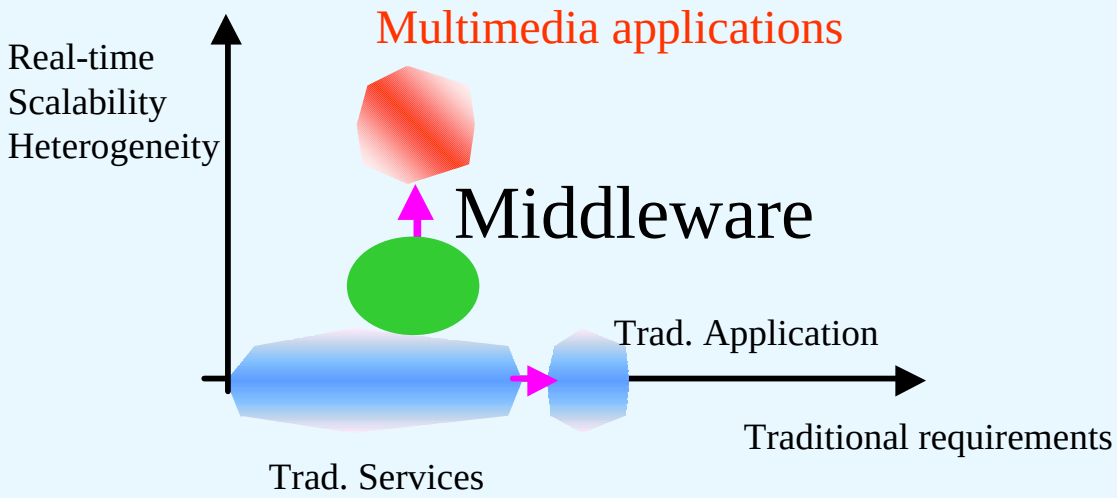
Odust Architecture



Outline

Conclusion

- We observed the convenience of a middleware



- It offers:
  - Multimedia network services
  - Synchronization
  - Floor control
  - Dynamic image transmission
- Future work
  - Add more components
  - Continue implementation
  - Try new ideas (see thesis)

Outline