# LOAD BALANCING IN CONTENT DISTRIBUTION NETWORKS

## - Load Balancing Algorithm for Distributed Cloud Data Centers -

Paul J. Kühn
University of Stuttgart, Germany
Institute of Communication Networks and Computer Engineering (IKR)
E-Mail: paul.j.kuehn@ikr.uni-stuttgart.de
Phone: +49-711-685-68027

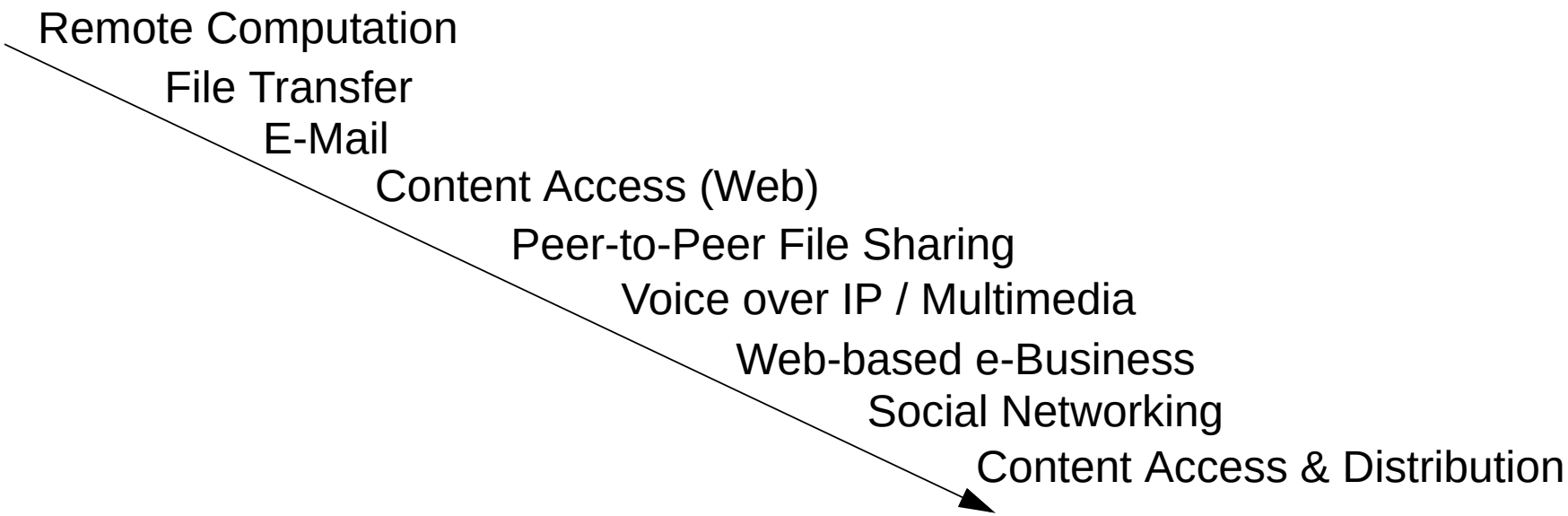
11. Fachtagung des ITG Fachausschusses 5.2 Kommunikationsnetze und -systeme
Future Internet: Architectures, Mobility and Security. Wien, 28. September 2012

# OUTLINE

1. Information Centric Networking
2. Content Distribution and Cloud Computing
3. Managing Content Distribution Networks (CDN)
4. Modeling Algorithms for Load Balancing in CDN
5. Performance Analysis and Results
6. Summary and Outlook

# 1. INFORMATION CENTRIC NETWORKING

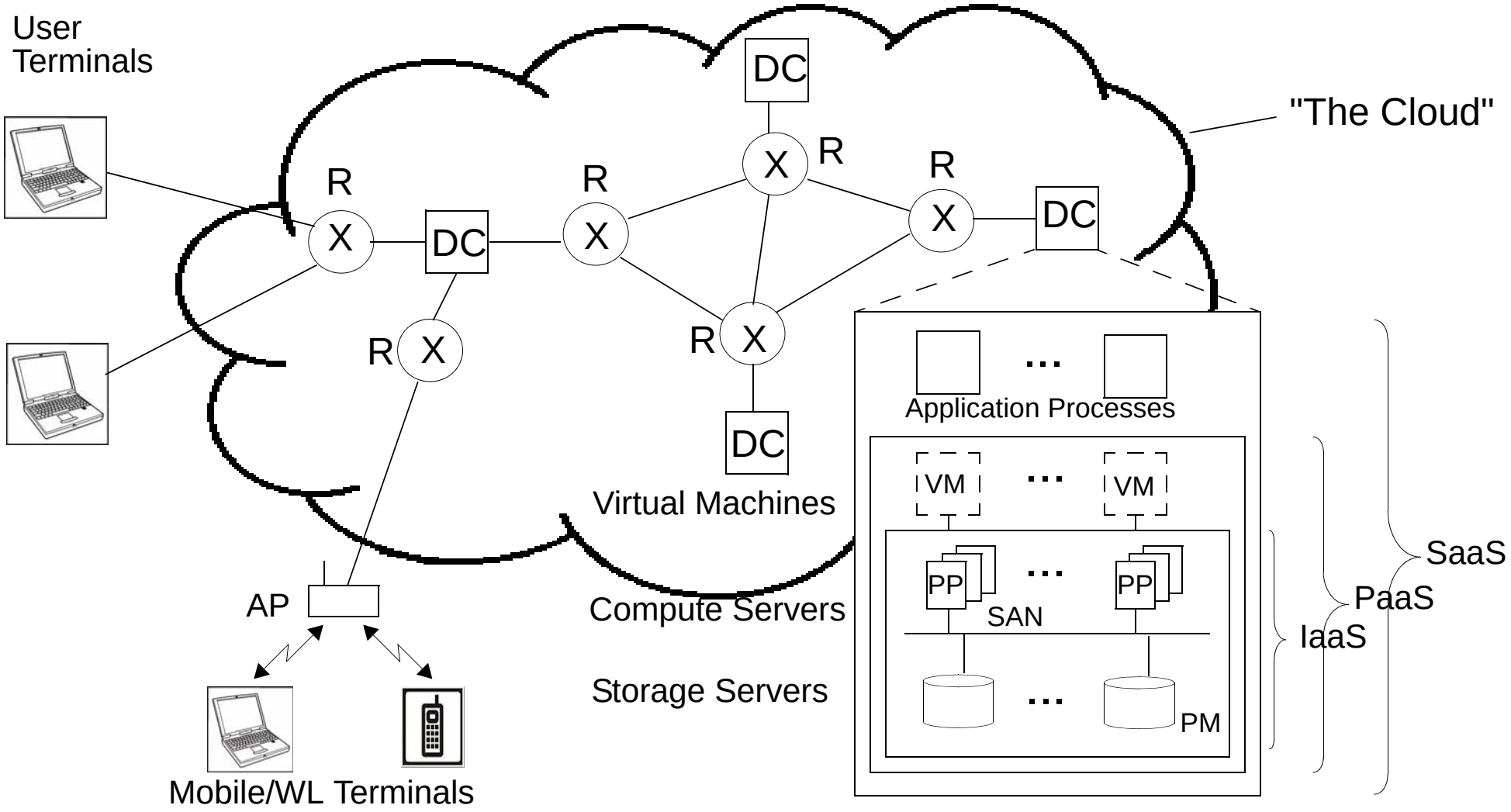
- Major Application Shifts in the Internet




- Paradigm Shifts

| | | |
|---|---|---|
| Transport Network | -----> | Information-Centric Network |
| Fixed Infrastructure | -----> | Wireless and Mobile Infrastructure |
| End-to-End Control | -----> | Network Control |
| Non-Realtime | -----> | Realtime |
| Best Effort Service | -----> | Service-Oriented Network (QoS, QoE, SLA) |

- Current Internet -----> Next Generation / Future Internet

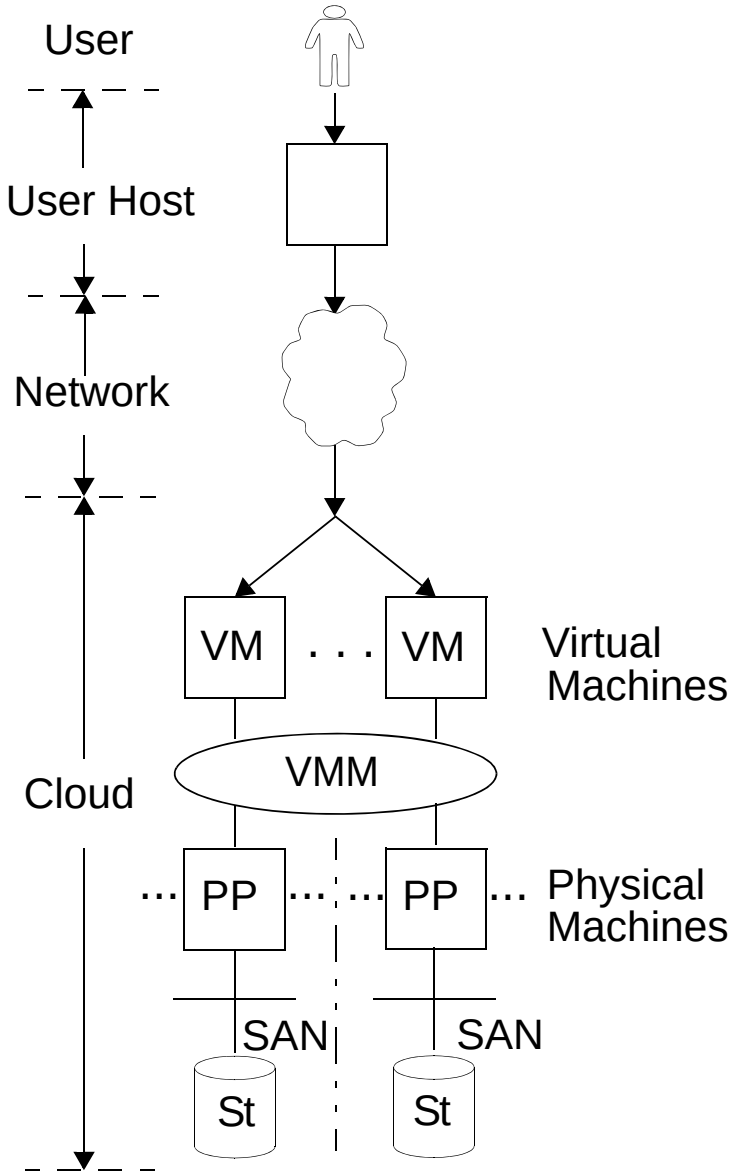


| | |
|---|---|
| CLOUD TYPES: | - Public, Private, Hybrid |
| CLOUD APPLICATIONS: | - Data Retrieval (Web) |
| | - Content Delivery |
| | - Business Processes |
| | - Scientific Grid |
| | - Social Networking |
| CLOUD FUNCTIONS: | - Resource Virtualization and Process Migration |
| | - Resource Sharing |
| INCENTIVES: | - Economics (Outsourcing/Insourcing of IT Services) |
| | - Reliability |
| | - Energy Reduction |

# 2. CONTENT DISTRIBUTION AND CLOUD COMPUTING - RESEARCH ASPECTS

| | |
|---|---|
| CLOUD ARCHITECTURES: | - Process Migration |
| | - Operating Systems, Hypervisor |
| | - Security and Privacy Protection |
| RESOURCE MANAGEMENT: | - Storage Strategies |
| | - Scheduling, Routing |
| | - Admission/Flow/Congestion Control |
| TRAFFIC ENGINEERING: | - Cloud Traffic Volumes/Characteristics |
| | - Traffic Matrix, Load Balancing |
| | - Quality of Service/Experience (QoS/QoE) |
| ECONOMIC ASPECTS: | - Tradeoff between Storage, Processing, and Communication |
| | - Service Level Agreements |
| | - Optimization |

# 3. MANAGING CDNs - VIRTUALIZATION AND VM MIGRATION

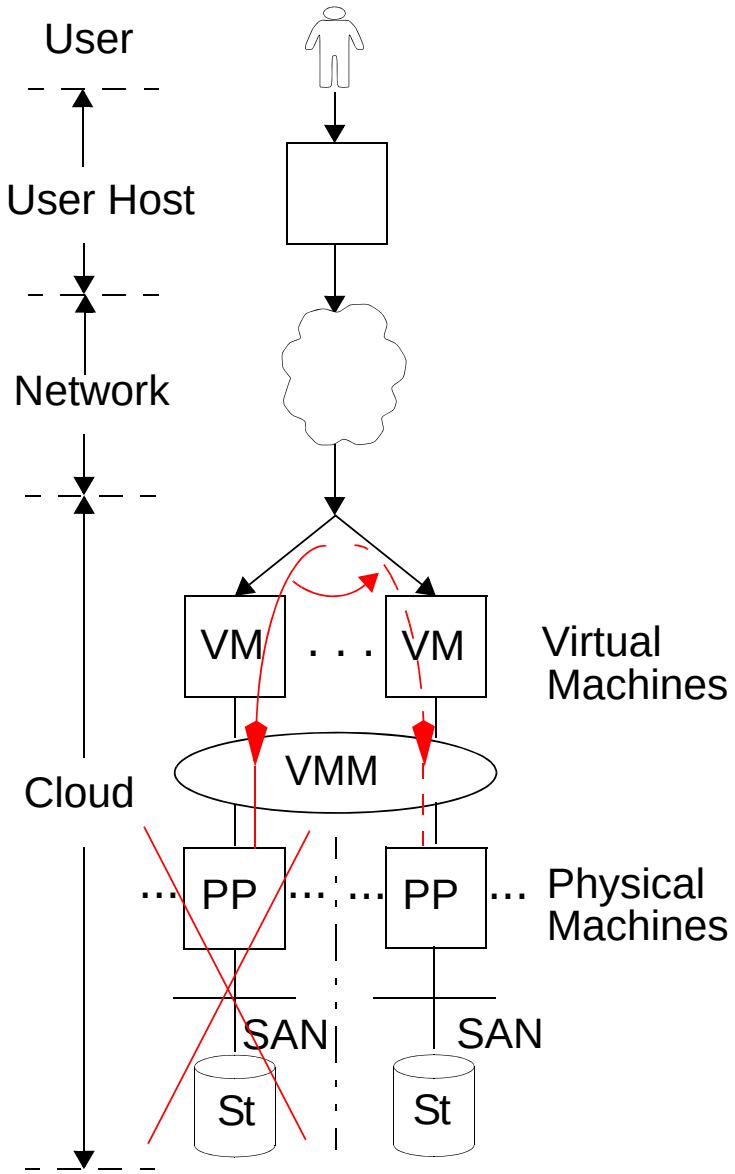



Cloud: Pool of Physical Resources
Interconnected by Network

VM: Virtual Machine
Virtualized View on the Resource Pool

VMM: VM Monitor ("Hypervisor")
Mapping of VM to PM

# 3. MANAGING CDNs - VIRTUALIZATION AND VM MIGRATION




Cloud: Pool of Physical Resources
Interconnected by Network

VM: Virtual Machine
Virtualized View on the Resource Pool

VMM: VM Monitor ("Hypervisor")
Mapping of VM to PM

VM Migration:

- Change of Assignment VM --- PM
- Different Migration Strategies

  "Suspend-and-Copy"

  "Pre-Copy"

  "Post-Copy"

# 3. MANAGING CDNs - DYNAMIC PROVISIONING OF PHYSICAL RESOURCES

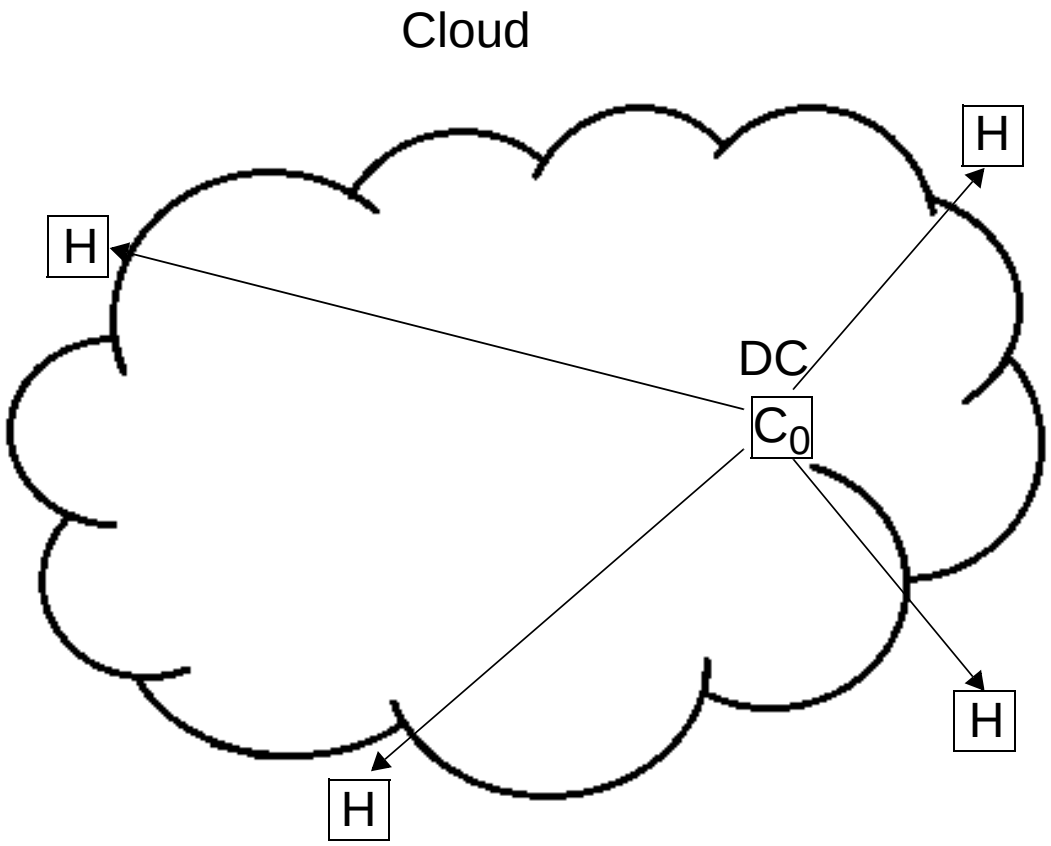
- Incentives

| | | |
|---|---|---|
| Hot Spot Mitigation | ----> | Overload Avoidance |
| Load Balancing | ----> | Economic Capacity Utilization, Energy Saving |
| Server Consolidation | ----> | Avoiding "Sprawling" of Resources |
| Performance/SLA | ----> | Meeting RT Requirements |
| Economics | ----> | Trade-off between Storage Cost -- Communication Cost in Case of Content Storage Replication |

- Content Location: Centralized or Decentralized

- Address Resolution by Publish/Subscribe Mechanism NNC (Network Named Content) Translation NNC ----> IP Address
  (Problem of the Legacy Internet without Identifier/Locator Split!)

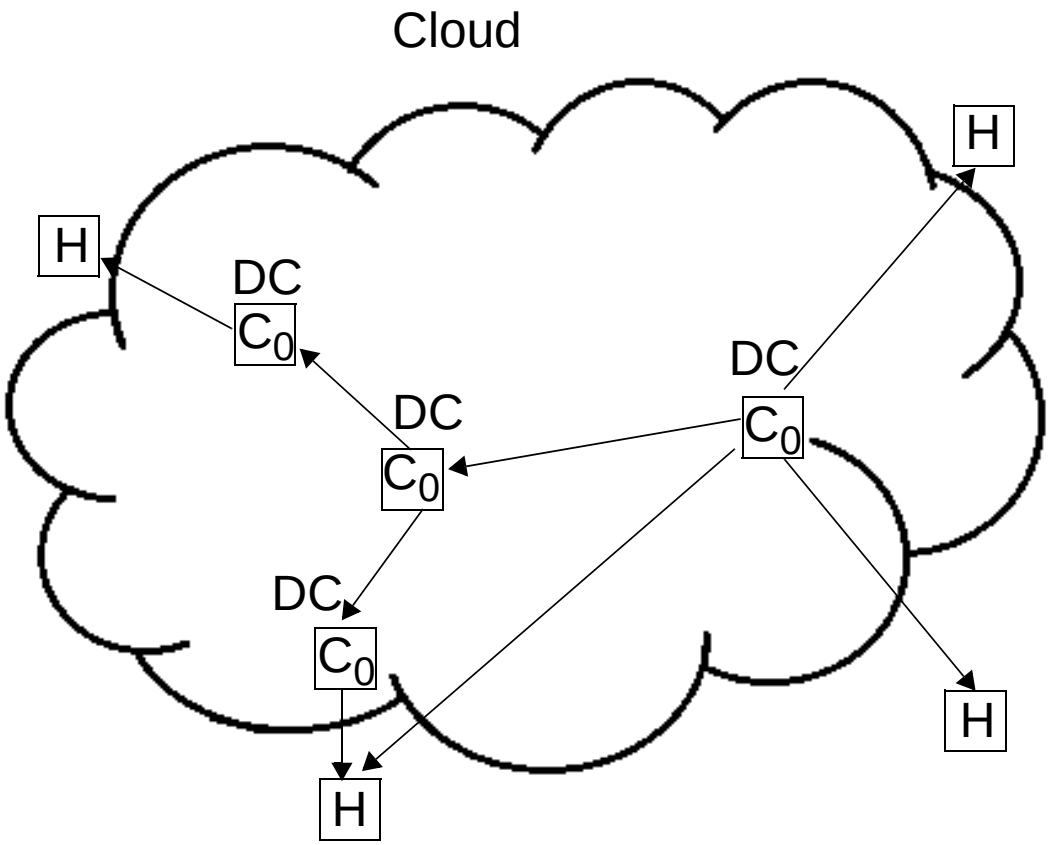
# 3. MANAGING CDNs - CENTRALIZED STORAGE




- Multicast Tree
- Minimum Storage Cost
- Maximum Communication Cost
- Maximum Latency
- High Risk, Reliability

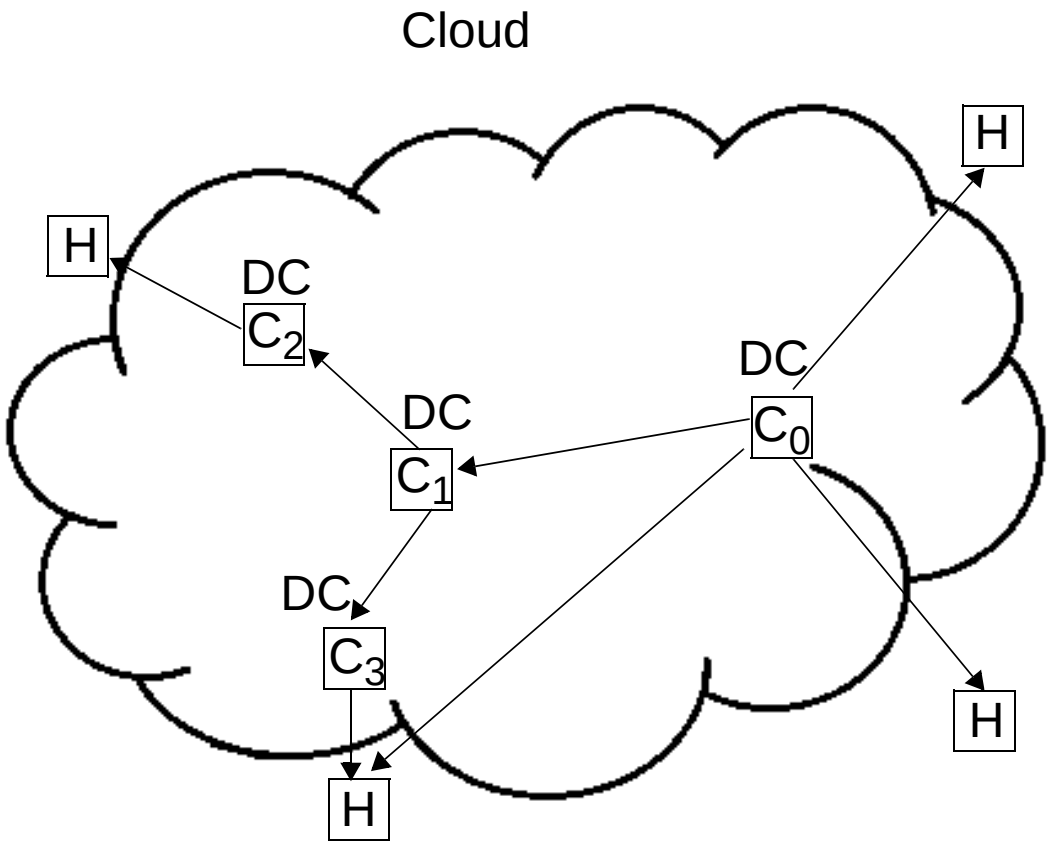
| | |
|---|---|
| H | User Host |
| DC | Data Center |
| $C_0$ | Content |

# 3. MANAGING CDNs - DECENTRALIZED STORAGE BY COMPLETE REPLICATION




- Replication of Full Content $C_0$ by Content Migration
- Higher Storage Cost
  Less Communication Cost
- Short Latency
- Overhead Cost by Replication

# 3. MANAGING CDNs - DECENTRALIZED STORAGE BY PARTIAL REPLICATION




- $C_0$ Full Content
- $C_i$ Partial Content

  $C_i \subseteq C_0$

  $C_2, C_3 \subseteq C_1$
- Dynamic Replication Dependent on Actual Demand
- Replicated Content Storage Management by Caching + LRU Replacement Strategy (Least Recently Used)

Open Questions: Dependence on "Working Set" of Content?
Caching of Content Fragments (Chunks, Packets, whole Objects)?
Amount of Prefetching to Avoid Starvation?
Performance, Energy Demand/Saving?

# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(1)

Modeling Assumptions:

- Cloud with Distributed Data Centers
- NNC Address Resolution by Publish/Subscribe Service
- Multi-Server Model for DC Content Delivery
- Sleep Mode + Activation Delays for Multi-Core Nodes
- Self-Adapting Activation/Deactivation of Core Nodes within each DC
  (state-dependent; can be extended to Measurement- or Forecast-Based Operation)
- Load Balancing Algorithm by State-Multicast

# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(2)

**BASIC IDEAS:**

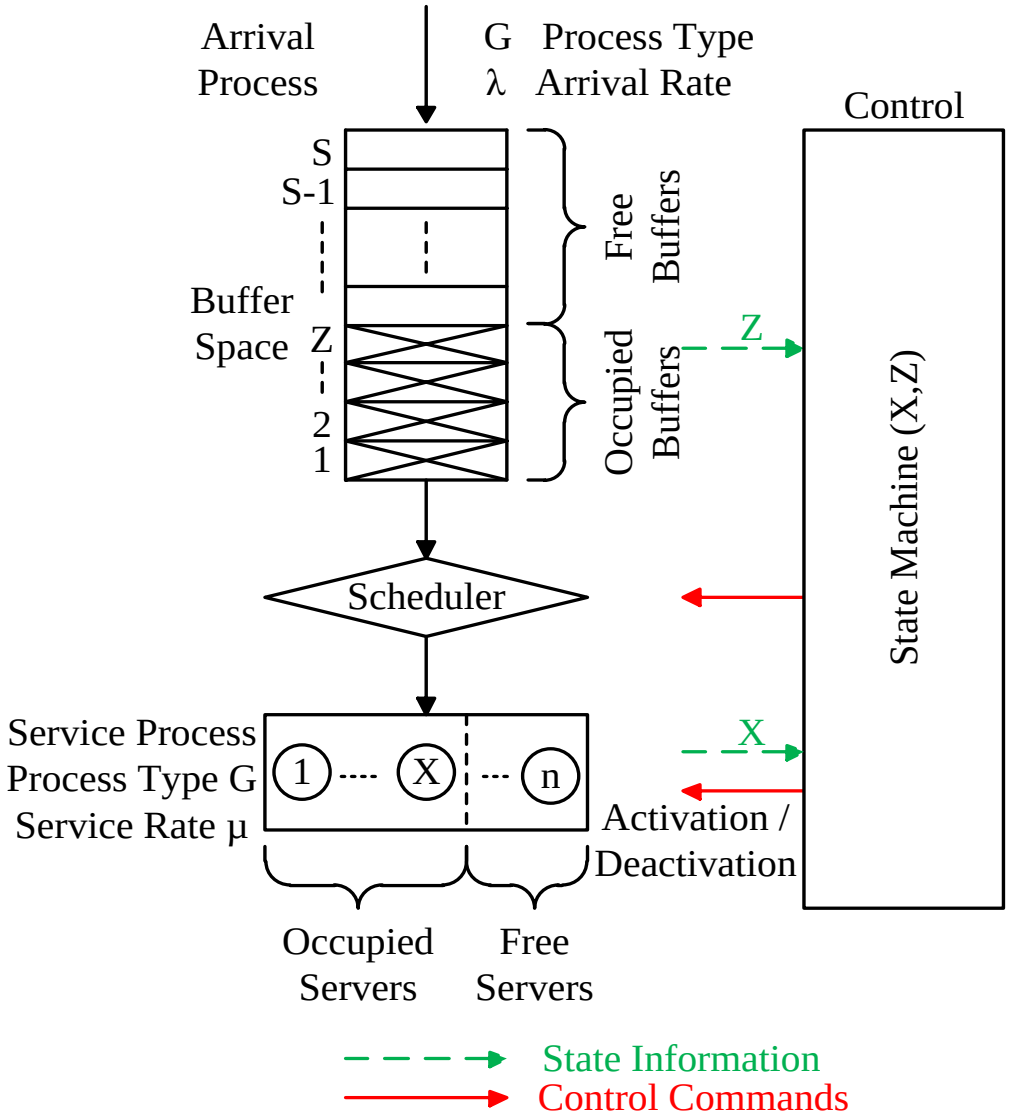
- Self-Adapting Operation of Data Center Resources
  - Local Monitoring of Load Development
  - Local Control of Resource Activation/Deactivation by FSM
- Distributed Load Balancing
  - Resource Utilization Distribution Algorithm
  - Decentralized/Centralized Re-Distribution of Load acc. to Utilization/Distance/Service Type/QoS/Cost/..-Criteria
  - Scheduling of Service Requests by Process Migration to Under-Utilized DCs

**BASIC MODEL:**

- Uniform Services, N Data Centers
- Focus on Processing Resources only
- $(n_i, \rho_i)$ Resource/Utilization Vector of $DC_i$, $i \in [1, N]$

# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(3)

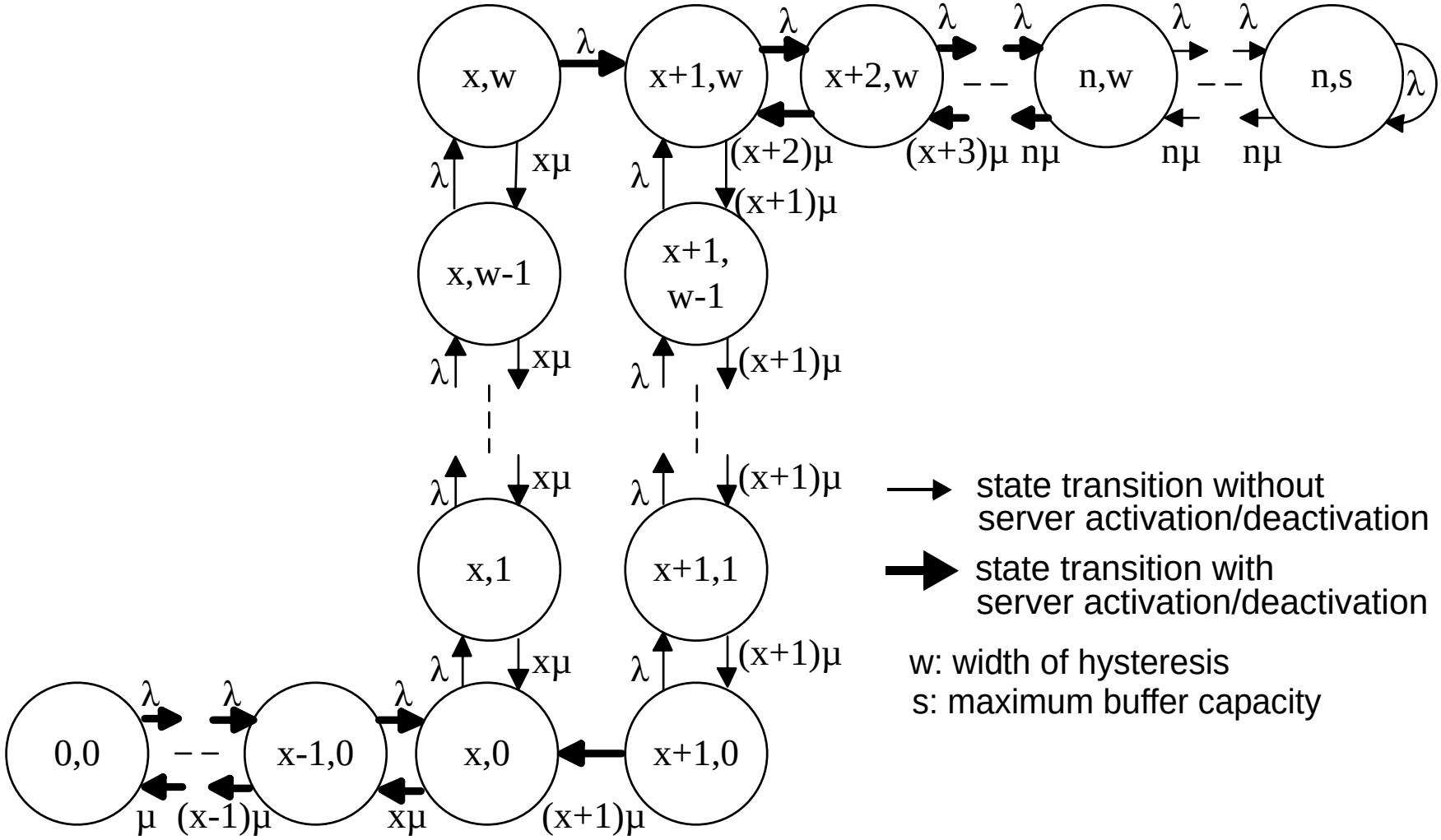
## INDIVIDUAL DC MODEL



# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(4)

## NON-ADAPTING MODEL BY FSM

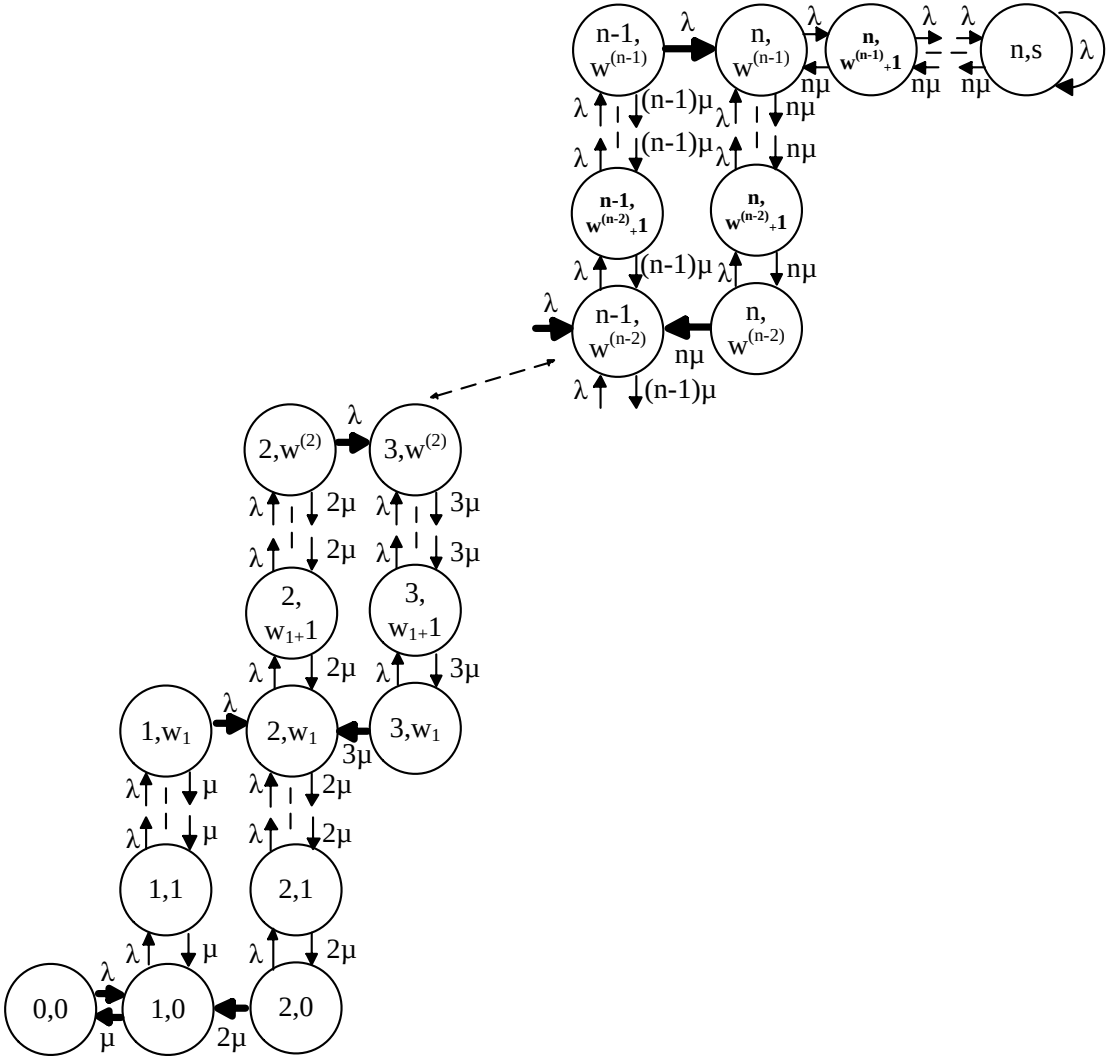
(1) SINGLE HYSTERESIS MODEL



# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(5)

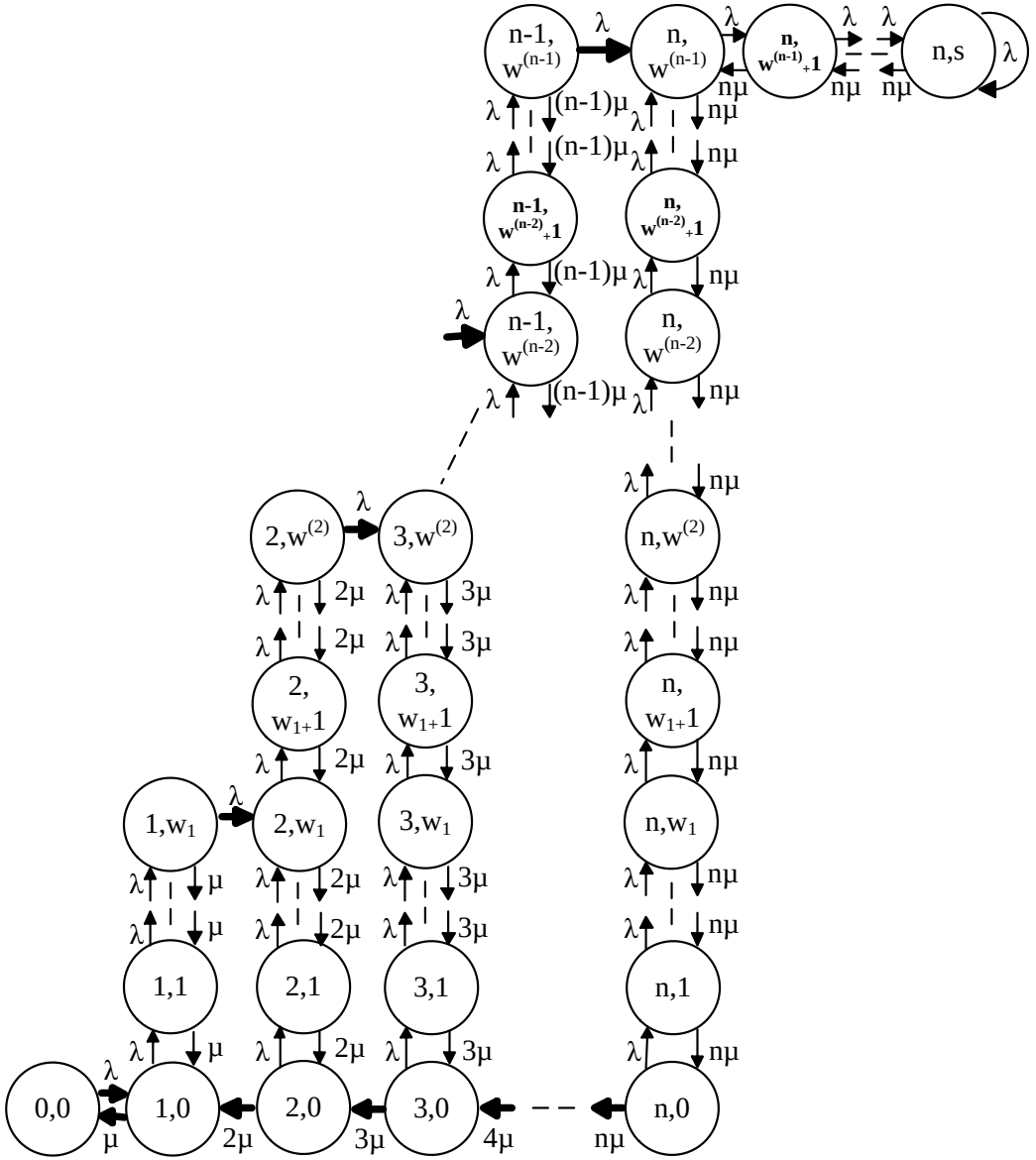
## SELF-ADAPTING MODEL BY FSM

(2) MULTIPLE SERIAL HYSTERESIS MODEL



# 4. MODELING ALGORITHMS FOR LOAD BALANCING IN CDN(6)

## SELF-ADAPTING MODEL BY FSM / (3) MULTIPLE PARALLEL HYSTERESIS MODEL



# 5. PERFORMANCE ANALYSIS AND RESULTS (1)

**MODEL ASSUMPTIONS**

- Load-Dependent Activation / Deactivation of Resources -
- Multiple Parallel Hysteresis Model with Server Activation Overhead
- Server Activation: after Server Booting, Queue Threshold Crossing, Process Migration
- Server Deactivation: only when a Server Becomes Idle or the System Becomes Empty (Server Consolidation)
- Notations:

| | |
|---|---|
| $\lambda$ | Arrival Rate (Requests, Data Units, ...) |
| $\mu$ | Service Rate of a Server |
| $\alpha$ | Activation Rate of a Triggered Server Activation |
| $\rho$ | Utilization Factor ($\rho = \alpha/\mu$) |
| $E[T_W \mid T_W>0]$ | Mean Waiting Times of Delayed Requests |
| $R_A$ | Server Activation/Deactivation Rate |
| W(>t)/W | Compl. DF of Buffered Requests |

# 5. PERFORMANCE ANALYSIS AND RESULTS (2)

**NUMERICAL EVALUATION**

- 1st Choice: Based on the fundamental solutions of Ibe/Keilson by Green's Function

  - **Result:** Numerically too complex

- 2nd Choice: Based on the fundamental solutions of Lui/Golubchik by Stochastic Complement Analysis

  - **Result:** Numerically too complex

- 3rd Choice: New solution by iterative recursions

  - **Result:** Extremely fast and numerically stable
    Extension to DF of delays
    Optimization wrt performance constraints

- Extensions

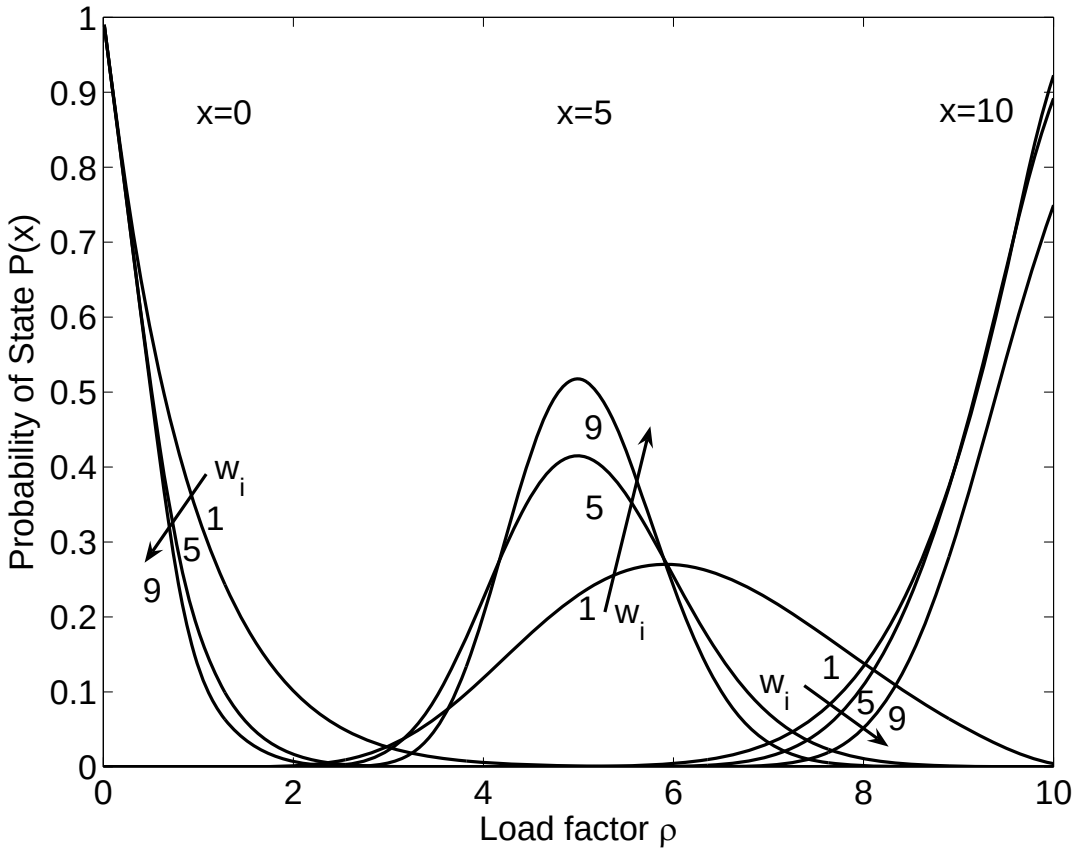
  In all solution methods certain generalizations are possible as

  – bulk arrivals
  – inclusion of activation overhead
  – inclusion of look-ahead activations

# 5. PERFORMANCE ANALYSIS AND RESULTS (3)

## NUMERICAL RESULTS (One DC only)

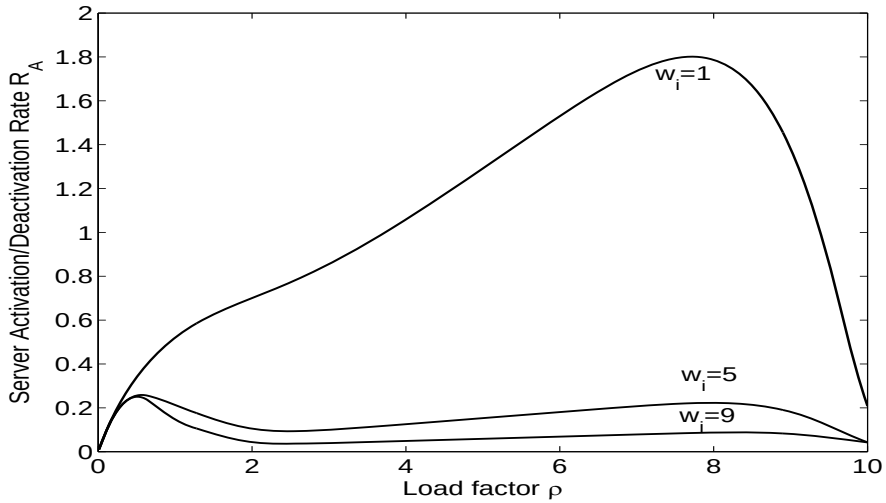
*MULTIPLE SERIAL HYSTERESIS MODEL* **Probabilities of State**



# 5. PERFORMANCE ANALYSIS AND RESULTS (4)

## NUMERICAL RESULTS (One DC only)

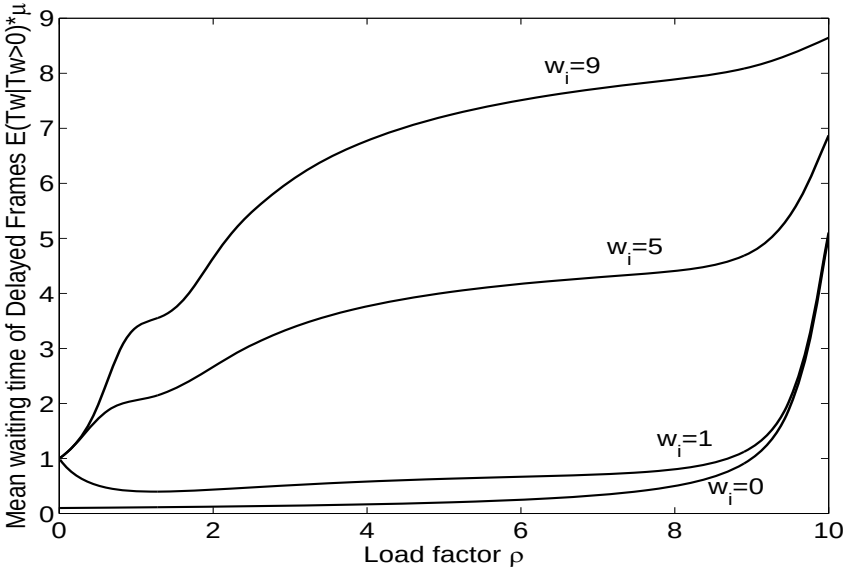
*MULTIPLE SERIAL HYSTERESIS MODEL* **Server Activation / Deactivation Rate**



# 5. PERFORMANCE ANALYSIS AND RESULTS (5)

## NUMERICAL RESULTS (One DC only)

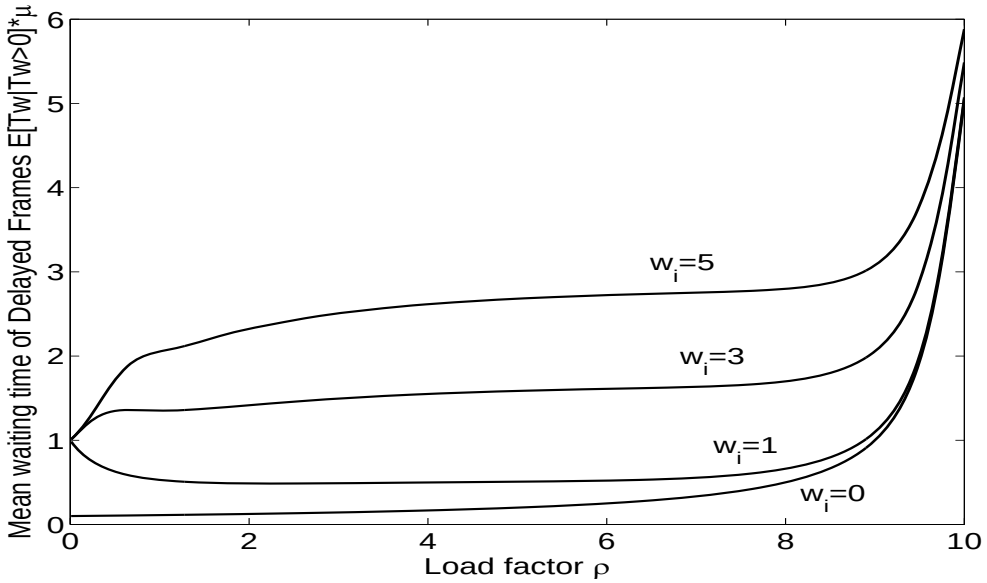
*MULTIPLE SERIAL HYSTERESIS MODEL* **Mean Waiting Time of Delayed Requests**



# 5. PERFORMANCE ANALYSIS AND RESULTS (6)

## NUMERICAL RESULTS (One DC only)

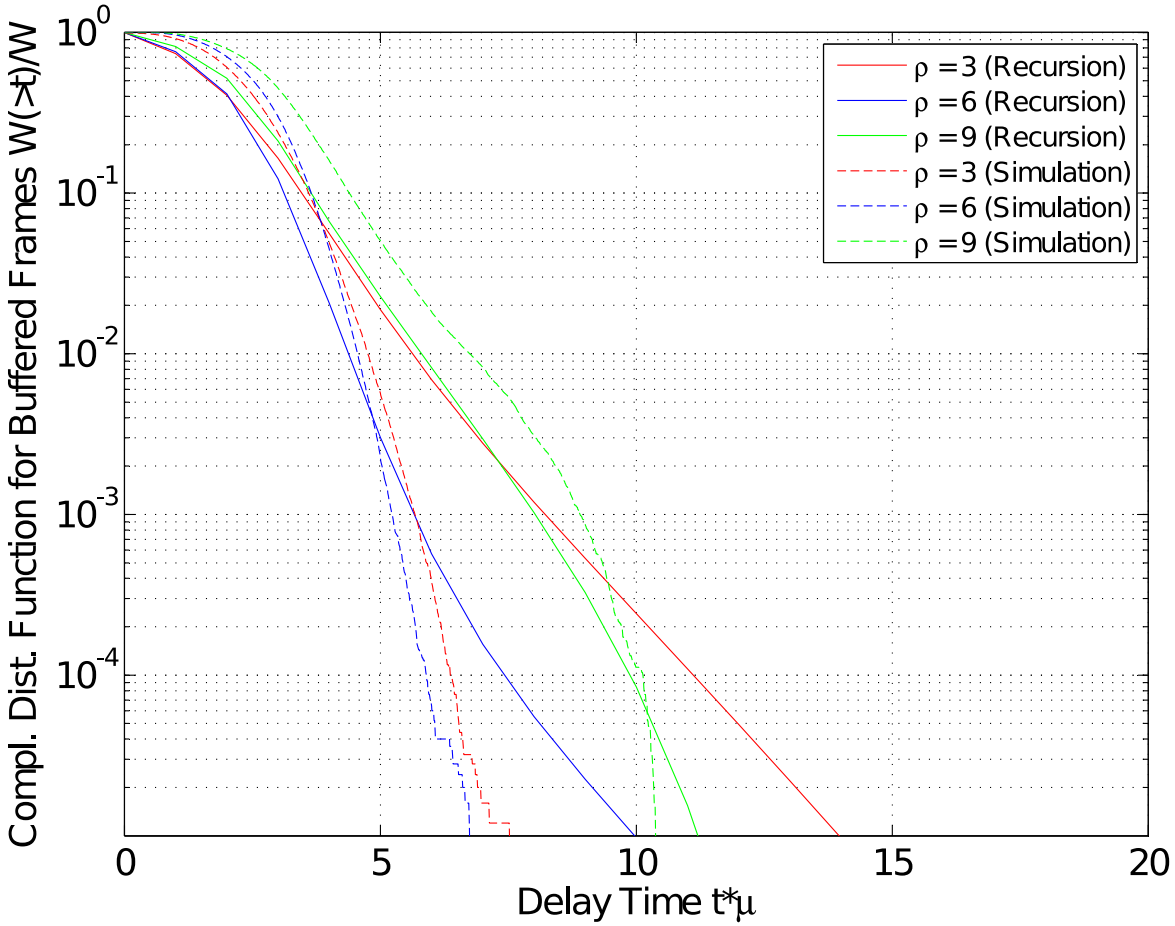
*MULTIPLE PARALLEL HYSTERESIS MODEL* **Mean Waiting Time of Delayed Requests**



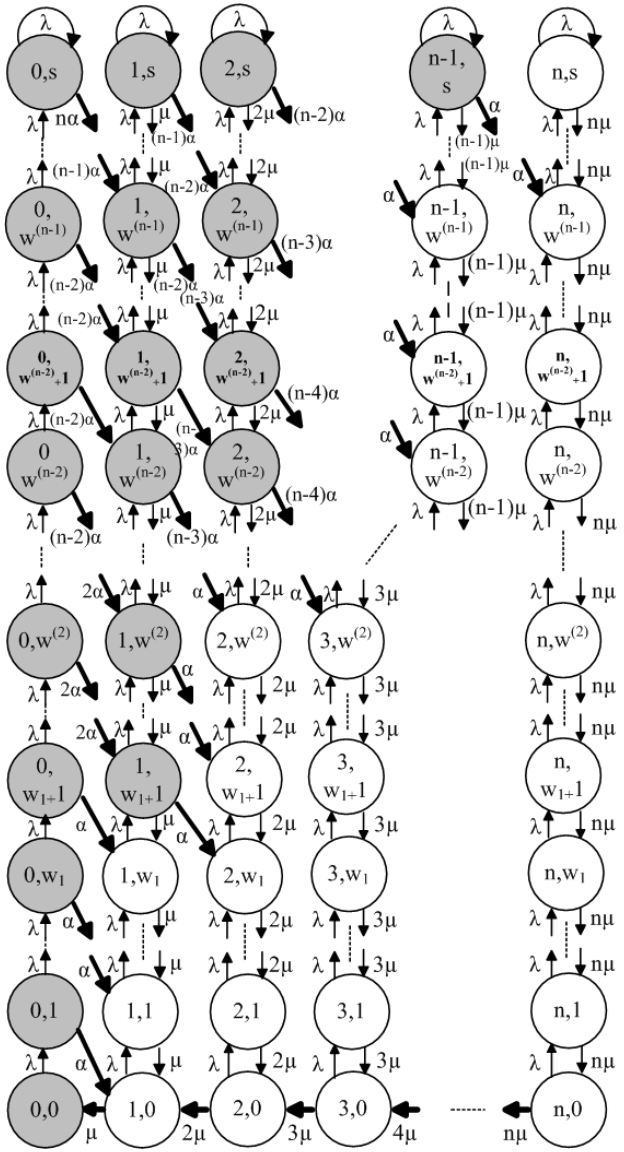
# 5. PERFORMANCE ANALYSIS AND RESULTS (7)

## NUMERICAL RESULTS (One DC only)

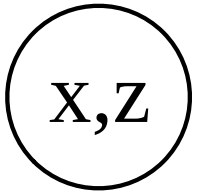
*MULTIPLE PARALLEL HYSTERESIS MODEL* **Compl. DF of Buffered Requests**



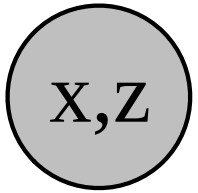
# 5. PERFORMANCE ANALYSIS AND RESULTS (8)



- Multiple Parallel Hystereses, Multi-Server Queuing System with/without Activation Overhead




without Activation Overhead




with Activation Overhead

# 5. PERFORMANCE ANALYSIS AND RESULTS (9)

**NUMERICAL RESULTS (One DC only):** *Probability State Distributions*

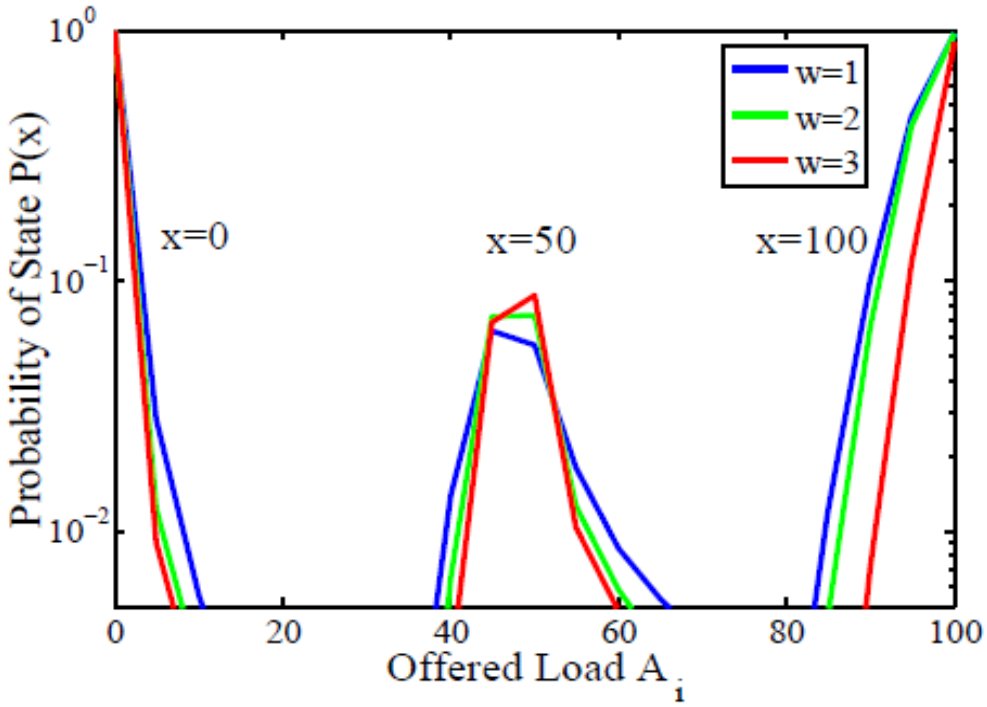



Figure: Probability of 'x' active servers vs. offered load
n = 100, s = 300, $\alpha$ = 1, variable w

# 5. PERFORMANCE ANALYSIS AND RESULTS (10)

**NUMERICAL RESULTS (One DC only):** *Server Activation Rate*

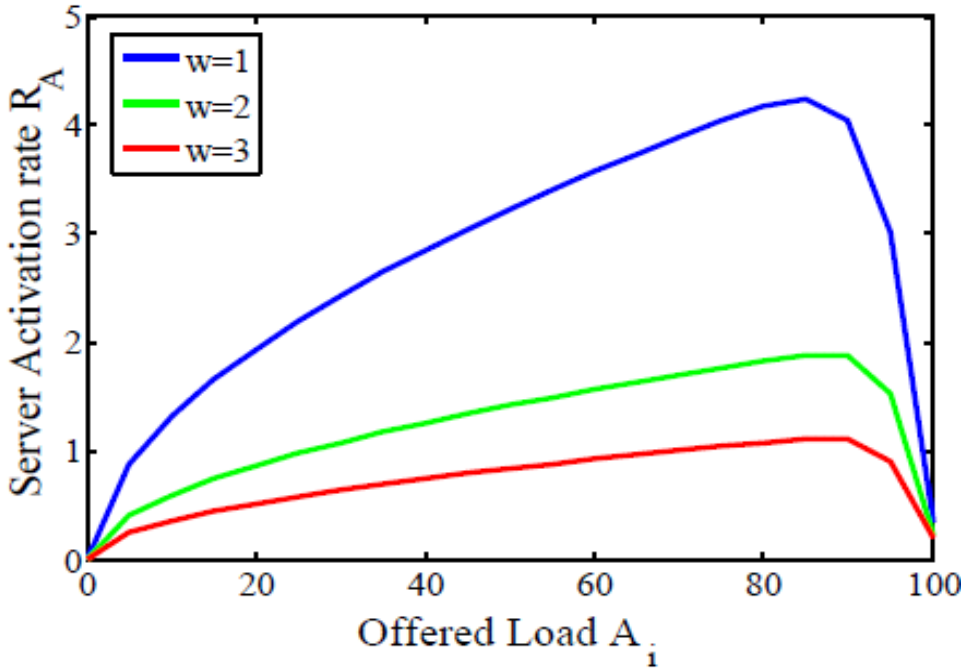



Figure: Server activation rate vs. offered load

n = 100, s = 300, $\alpha$ = 1, variable w

# 5. PERFORMANCE ANALYSIS AND RESULTS (11)

**NUMERICAL RESULTS (One DC only):** *Mean Delay*

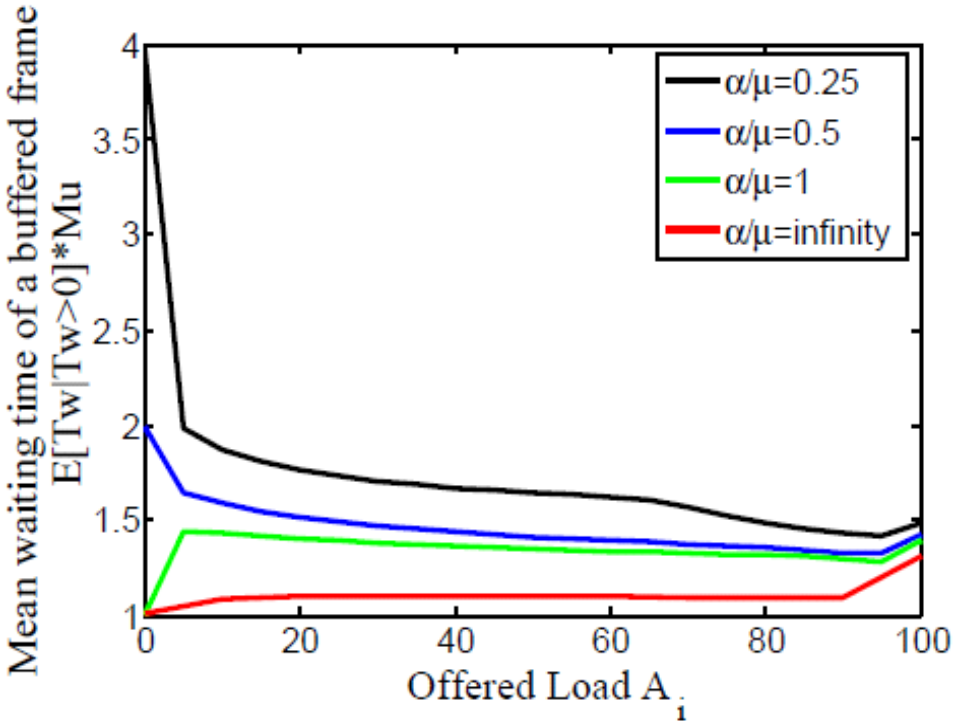



Figure: Mean delay of delayed frames vs. offered load

n = 100, s = 200, w = 2, variable $\alpha/\mu$

# 5. PERFORMANCE ANALYSIS AND RESULTS (12)

## LOAD BALANCING ALGORITHM (1)

- Assumptions
  - N data centers are involved in the load balancing process
  - Each data center has $n_i$ servers
  - Each data center has offered load $A_i = \lambda_i/\mu_i$

# 5. PERFORMANCE ANALYSIS AND RESULTS (13)

## LOAD BALANCING ALGORITHM (2)

- Algorithm Steps
  1. Determine the maximum load that could be handled by each data center
     $A_{(max,i)} = [\text{function } (n_i) \mid t_w < t_{SLA}]$
  2. Determine the load margin $\Delta A(i) = A_i - A_{(max,i)}$
     If $\Delta A(i) > 0$: Data center i is overloaded and the extra load $\Delta A(i)$ needs to be shifted to another data center.
     If $\Delta A(i) \leq 0$: Data center i can still handle extra load equal to $\Delta A(i)$ without affecting its performance.
  3. For DCs whose $\Delta A(i) > 0$, shift this amount of load to the nearest DC who can accommodate this load shift, fully or partially.
  4. Repeat the above steps until no more load shifting is necessary.

# 6. SUMMARY AND OUTLOOK

- Internet Paradigm Shift: Information Transport ----> Information Centric Network
- Cloud Server Virtualization allows for Flexible Content Distribution and Access
- Network Named Content vs. Network Caching
- Models for Self-Adapting DC Server Activation/Deactivation
- Trade-off between Power Reduction and Performance
- Algorithm for Load Balancing and Server Consolidation

**Outlook**

- Realistic Cloud Application Classes
- Refined Models for DC Architectures and Operations
- Cost Optimization