Software System Testing

Speaker: Jerry Gao Ph.D.

Computer Engineering Department
San Jose State University

email: jerry.gao@sjsu.edu
URL: http://www.engr.sjsu.edu/gaojerry

Presentation Outline

- Introduction
  - ❑ Performance Testing, Process, and Tools
  - ❑ Performance Evaluation and Models
  - ❑ Performance Evaluation and Metrics
- Performance Evaluation Models:
  - ❑ Event-Based Function Scenario Model for System Performance Evaluation
  - ❑ Event-Based Transition Model for System Performance Evaluation
- Performance Metrics
  - Performance, throughput, availability, reliability, scalability, and utilization
- Supporting Environment

Performance Testing

Performance testing – refers to test activities on checking system performance

The major objectives of performance testing:

- To confirm and validate the specified system performance requirements.
- To check the current product capacity to answer the questions from customers and marketing people.
- To identify performance issues and performance degradation in a given system

Performance Testing – Focuses

- System process speed (Max./Min./Average)
  - system processes, tasks, transactions, responses.
  - data retrieval, data loading
- System throughput (Max./Min./Average)
  - loads, messages, tasks, processes
- System latency (Max./Min./Average)
  - Message/event, task/process
- System utilization (Max./Min./Average)
  - Network, server/client machines.
- System availability (component-level/system-level)
  - component/system, services/functions
  - system network, computer hardware/software

Performance Testing - Focuses

- System reliability (component-level/system-level)
  - component/system, services/functions
  - system network, computer hardware/software
- System scalability (component-level/system-level)
  - load/speed/throughput boundary
  - improvements on process speed, throughput
- System successes/failures rates for
  - communications, transactions, connections
  - call processing, recovery, …
- Domain-specific/application-specific
  - agent performance
  - real-time report generation speed
  - workflow performance

Performance Testing - Test Process

- ❑ Understand system and identify performance requirements
- ❑ Identify performance test objectives and focuses
- ❑ Define performance test strategy:
  - o Define/select performance evaluation models
  - o Define/select performance test criteria
  - o Define and identify performance test metrics.
- ❑ Identify the needs of performance test tools and define performance test environment
- ❑ Write performance test plan
- ❑ Develop performance test tools and support environment
- ❑ Set up the target system and performance test beds
- ❑ Design performance test cases and test suite
- ❑ Performance test execution and data collection
- ❑ Performance analysis and reporting

Performance Test – Tools

Performance test tools can be classified into:

- ❑ Simulators and data generators:
  - o Message-based or table-based simulators
  - o State-based simulators
  - o Model-based data generators, such as
    - o Pattern-based data generators
    - o Random data generators
- ❑ Performance data collectors and tracking tools
    - o Performance tracking tools
- ❑ Performance evaluation and analysis tool
  - o Performance metric computation
  - o Model-based performance evaluation tool
- ❑ Performance monitors
  - o For example, sniffer, Microsoft performance monitor
  - o External third-party tools
- ❑ Performance report generators

Performance Evaluation

What is performance evaluation?
→ Using a well-defined approach to study, analyze, and measure the performance of a given system.

The basic tasks and scope:

- ❑ Collect system performance data
- ❑ Define system performance metrics
- ❑ Model system performance
- ❑ Measure, analyze, estimate system performance
- ❑ Present and report system performance

Performance Evaluation – Objectives and Needs

The major objectives:

- ❑ Understand product capacity
- ❑ Discover system performance issues
- ❑ Measure and evaluate system performance
- ❑ Estimate and predict system performance

The basic needs are:

- ❑ Well-defined performance metrics
- ❑ Well-defined performance evaluation models
- ❑ Performance evaluation tools and supporting environment

Performance Evaluation - Approaches

- ❑ Performance testing: (during production)
  - o measure and analyze the system performance based on performance test data and results
- ❑ Performance simulation: (pre-production)
  - o study and estimate system performance using a simulation approach
- ❑ Performance measurement at the customer site: (post-production)
  - o measure and evaluation system performance during system operations

Performance Evaluation - Models

What is a performance evaluation model?

→A well-defined formal model which depicts different prospects of system performance of a system.

Why do we need performance evaluation models?

→To present the system performance properties

→To provide a guideline for engineers to find the strategy on performance evaluation.

→To set up a foundation to define performance metrics.

→To identify the needs of the target performance environment.

Performance Evaluation - Models

Type of performance evaluation models:

- Queuing model
- Scenario-based evaluation model
- Architecture-based evaluation model
  - Component-based evaluation model
- Process-based evaluation model
- Transaction-based evaluation model
- Transition-based models
  - State-based evaluation model
- Domain-oriented evaluation model

Performance Evaluation - Models for Portal V5

Two performance evaluation models are defined:

- ❑Event-Based Function Scenario Model for System Performance Evaluation
- ❑Event-Based Transition Model for System Performance Evaluation

Portal V5 systems can be viewed as a component-based distributed system over the Internet. It accepts, processes, and supports the following types of event messages.

- ❑ System-caller interaction event messages
- ❑ System-agent interaction event messages
- ❑ Call-oriented event interactions between components

Event-Based Function Scenario Model

Model definition: An event-based function scenario model is an event-based scenario diagram with tracked time stamps.

We use ***G(N, E, T, O)*** to represent a functional scenario diagram.

- ***N*** is a set of component nodes.
- ***E*** represents a set of direct links. Each link ***e*** is a direct edge ***(C_r, C_d),*** which represents an event that is sent by component ***C_r*** and received by the component ***C_d***.
- ***T*** is a set of tracked time stamps for links. Each link has a pair of time stamps ***(t_i, t_o),*** where ***t_i*** indicates the incoming time stamp of event ***e*** in ***C_r***, and where ***t_o*** indicates the outgoing time stamp of event ***E_p*** in $\mathbf{C_d}$.
- ***O*** is a set of pairs ***(E_p, order),*** each of them represents the order of for an event occurred in the scenario.

An event-based functional scenario S_q is a sequence of interaction events between components in a system. ***S_q*** can be represented as a path in a functional scenario diagram. Thus, ***S_q*** can be denoted as ***E_{i1}, …., E_{im}***, where ***E_{i1}*** is the starting event of ***S_q***, and ***E_{im}*** is the ending event of ***S_q***.

Event-Based Function Scenario Model

Applications and benefits:

To provide a sound and theoretic foundation to help performance evaluation in the following aspects:

- ❑Assist and enforce performance test engineers to understand the functional scenarios, system structures and behaviors.
- ❑Provide a fundamental base to develop a systematic performance solution.
- ❑Provide a guideline to develop a performance test environment.

Model Limitation:

- ❑Each component receives and processes incoming/outgoing events in a sequential way although these events can be processed in a concurrent way inside the component.
- ❑The sequence of events in a functional scenario is fixed according to the system behaviors and implementations.
- ❑Each scenario has only one starting event and ending event.
- ❑Each outgoing event must depend on at least one incoming event.

Event-Based Function Scenario Model: An Example

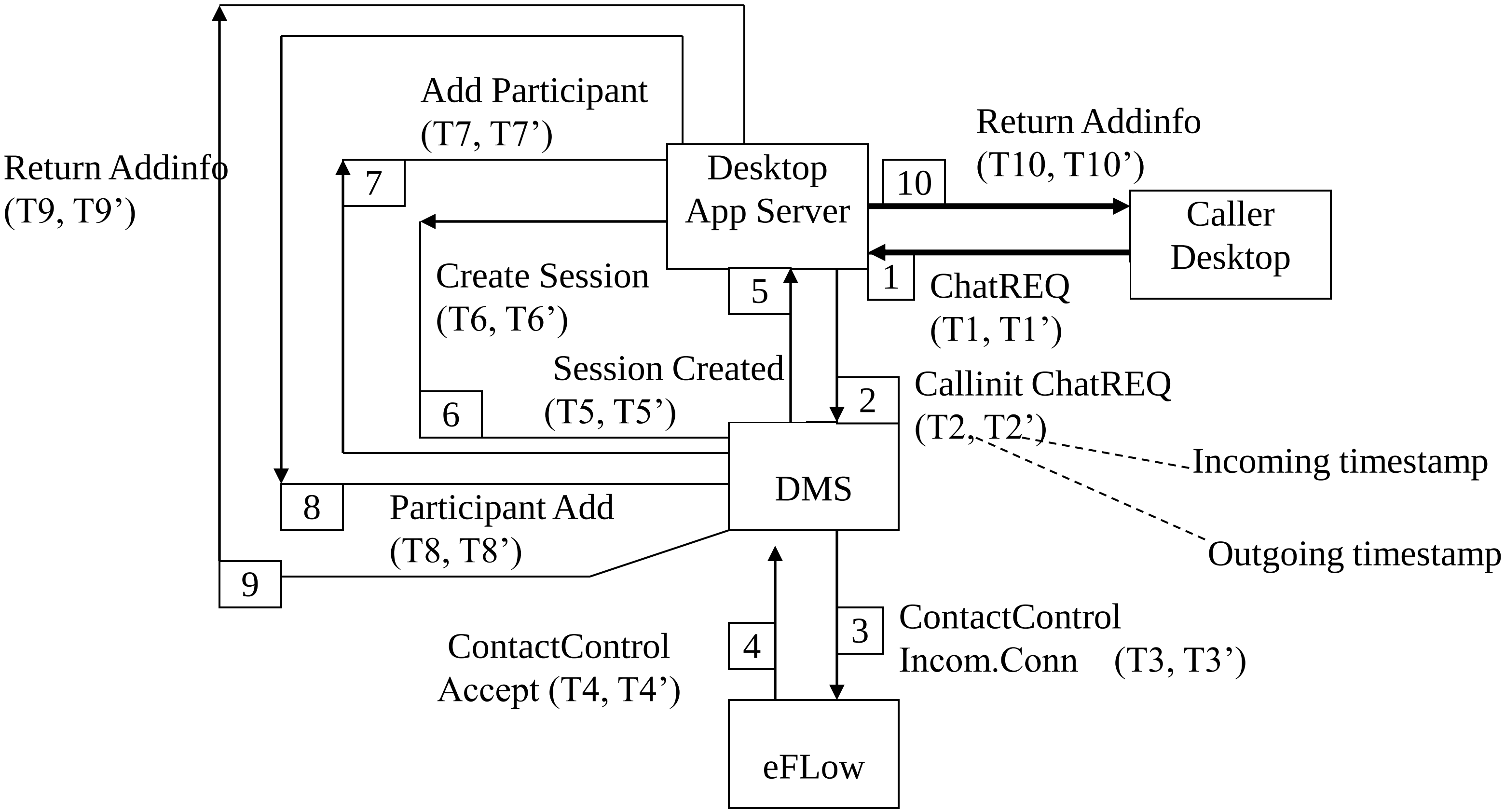


Figure 2. An Event-Based Functional Scenario for TextChat Sequence #1

Event-Based Transition Model

Model definition: An event-based system evaluation model is an event-based transition diagram with a set of tracked time stamps.

We use ***D(S, L)*** to represent an event-based transition diagram,

- ❑***S*** is a set of event transition states. Each transition state node represents a component's transition state from an incoming event to an outgoing event.
- ❑***L*** represents a set of direct links between state nodes. Each link l_i ***in L*** is a direct edge (S_r, S_d), which indicates an event (either an incoming/outgoing event) between two transitions.
- ❑Each link (say l_i) has a pair of count vectors (I_{ct}, O_{ct}). I_{ct} = ($I_{ct1,\ldots,}$ I_{ctm}) is a count vector, each element indicates the total number of tracked incoming events in a test time frame for the link. O_{ct} = ($O_{ct1,\ldots,}$ O_{ctm}) is a count vector, each element indicates total number of tracked outgoing events in a test time frame for the link.

Event-Based Transition Model

Major Applications:

The event-based performance model can be used to measure system performance at the component-level and the system level in the following areas:

- ❑ Function and service-oriented throughput for different types of media requests.
- ❑ Function and service-oriented reliability for different types of media requests.
- ❑ Function and service-oriented availability for different types of media requests.
- ❑ System scalability and performance improvement.

The Major Benefits:

→To provide a sound base and correct theoretic foundation for developing a systematic performance evaluation solution for XXXXX Portal V.5 products.

→To help and enforce performance test engineers to understand the behaviors of a system.

Event-Based Transition Model: An Example

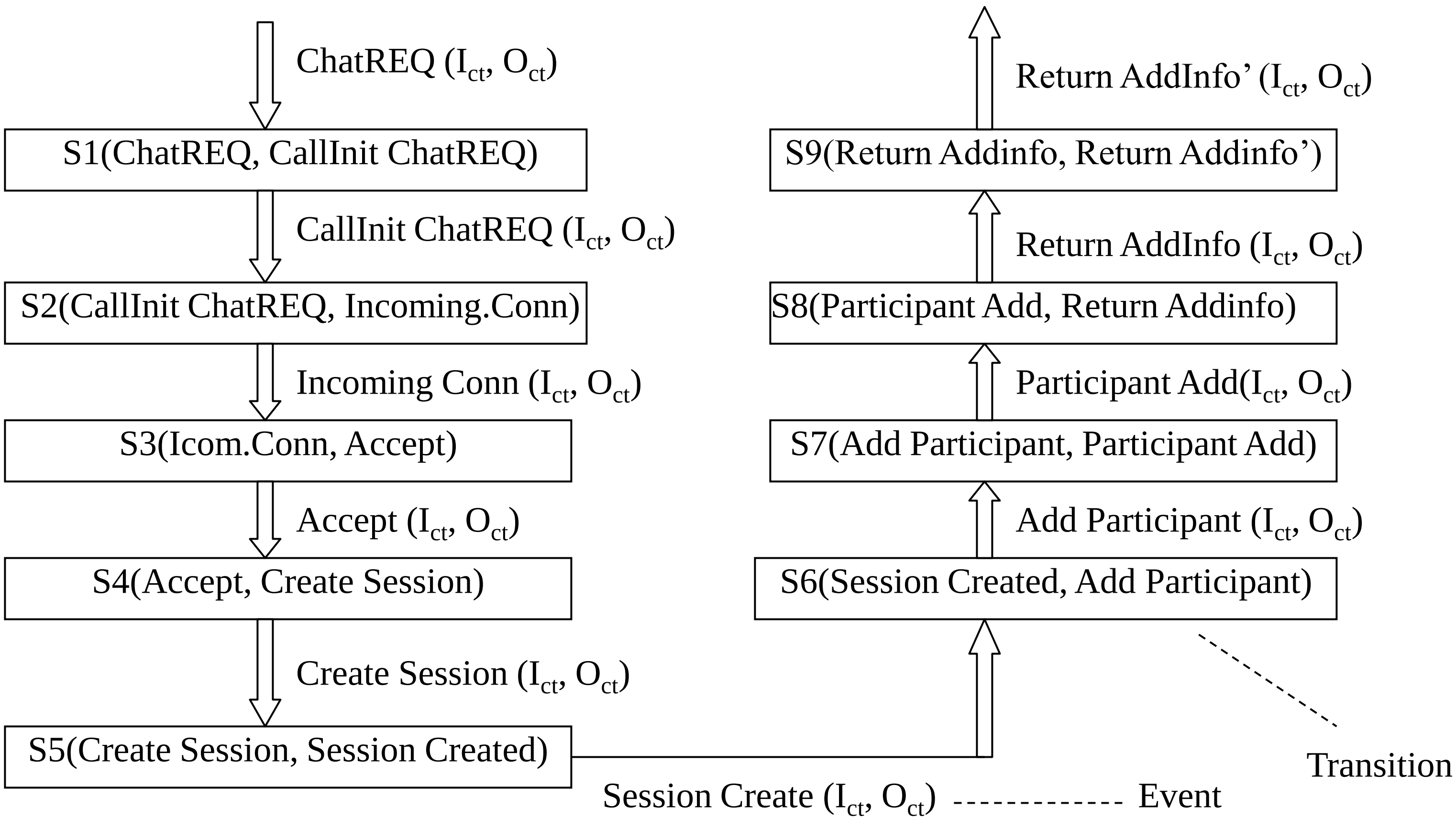


Figure 3(b). An Event-Based Transition Diagram for TexChat Sequence #1

Performance Evaluation - Metrics

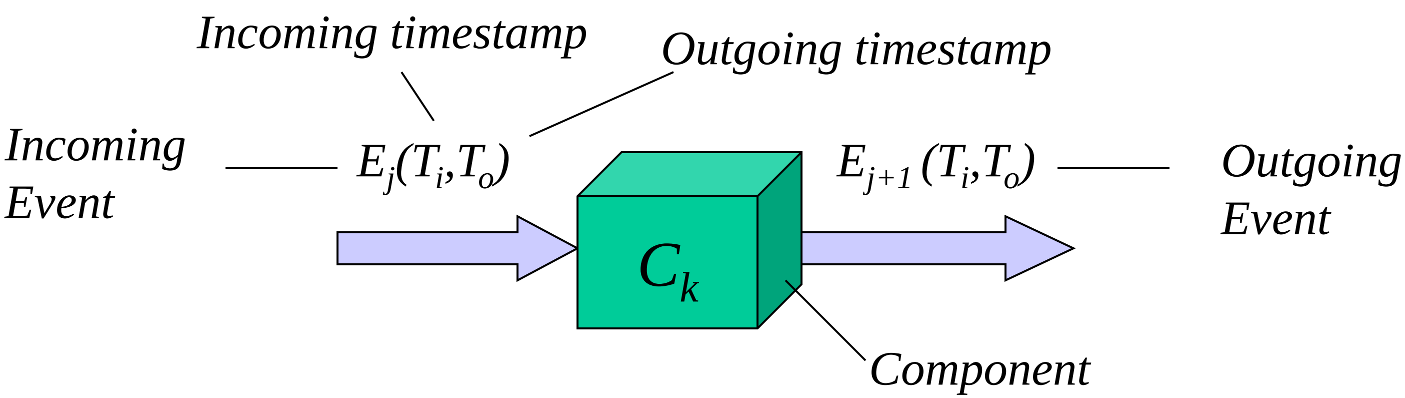
- ❑Performance metrics
  - oCall request process time
  - oEvent interaction latency
- ❑Throughput metrics (component and system level)
  - oCall processing throughput
  - oCall load throughput rate
- ❑Availability metrics (component and system level)
- ❑Reliability metrics (component and system level)
- ❑Scalability metrics
- ❑Utilization metrics

Performance Metrics

Common used performance metrics: (for components/systems)
- ❑ Functional or process speed metrics
- ❑ User response time metric
- ❑ Communication speed metric
- ❑ Transaction speed metric
- ❑ Latency metric

Performance metrics for Portal V.5 products:
- ❑ Call process time metric
- ❑ Call process speed metric
- ❑ Event latency metric

Performance Metric – Call Process Time



Component Process Time Metric for a call request *reqi*:

Process-Time$_{Sq}$ ($C_{k,}$, reqi)
= Ej+1's outgoing timestamp – Ej's incoming timestamp

Performance Metric – Call Process Time

Component Process Time Metric for call requests Req(media type):

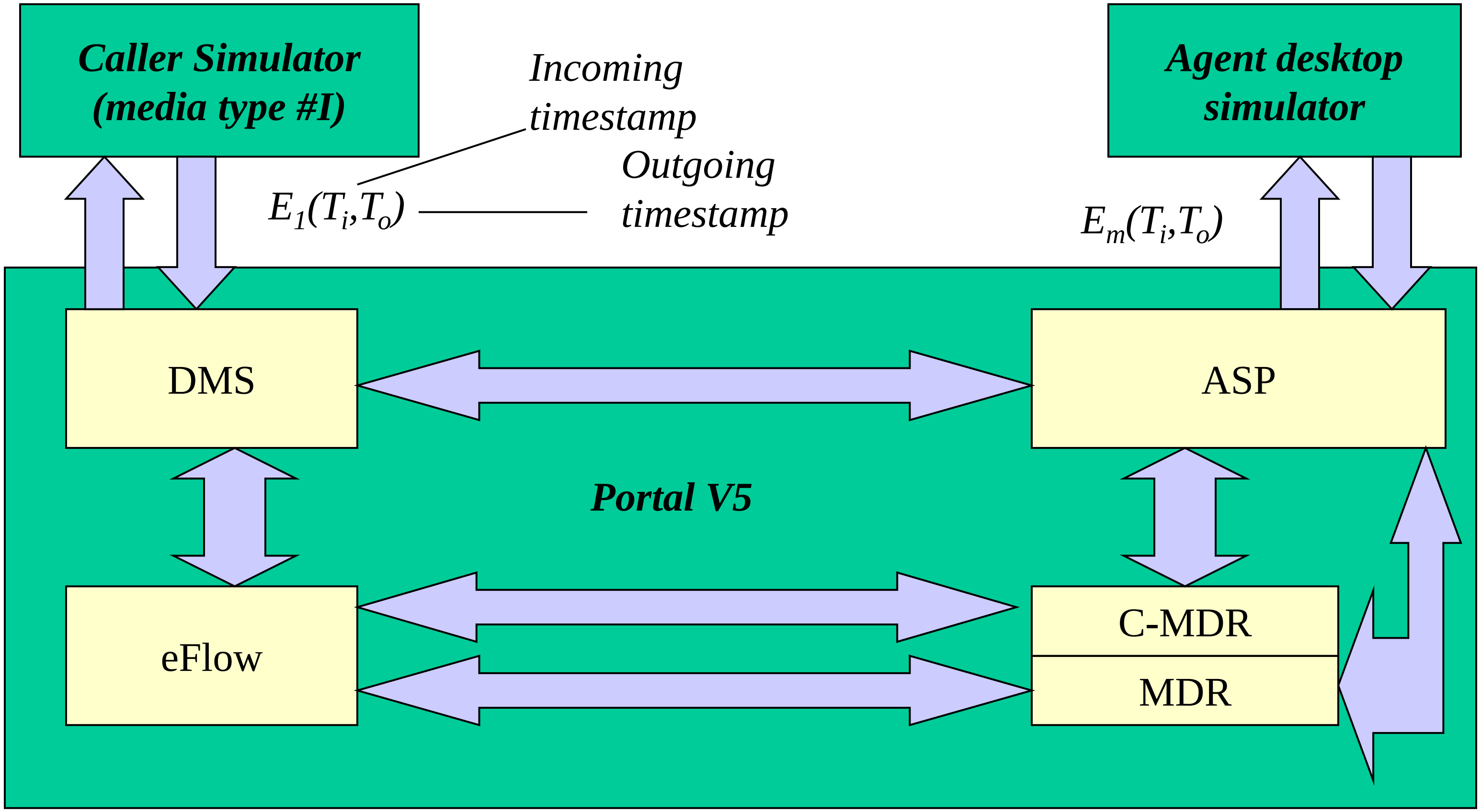
$$Req(media\ type) = \{ req_1, \ldots, req_n \}$$

$$Max\text{-}Process\text{-}Time_{Sq}\ (C_{k,}, Req(media\ type))$$
$$= Max_i \{ Process\text{-}Time_{Sq}\ (C_{k,}, req_i) \} \qquad (i = 1,\ldots n)$$

$$Min\text{-}Process\text{-}Time_{Sq}\ (C_{k,}, Req(media\ type))$$
$$= Min_i \{ Process\text{-}Time_{Sq}\ (C_{k,}, req_i) \} \qquad (i = 1,\ldots n)$$

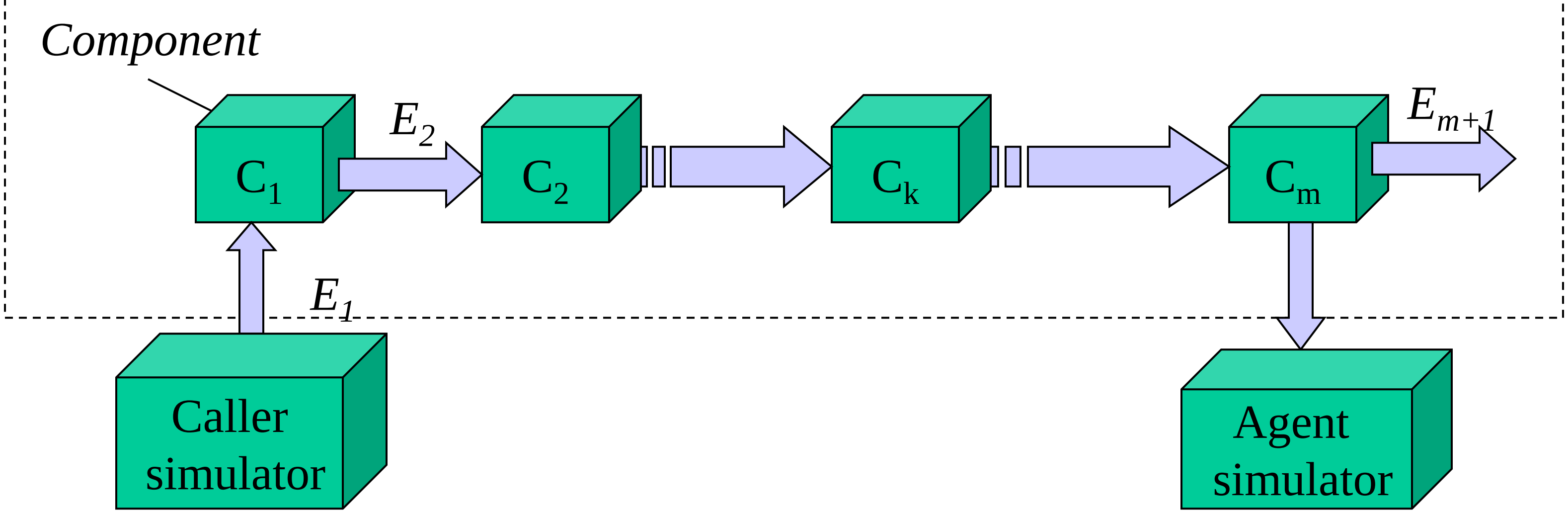
$$Avg\text{-}Process\text{-}Time_{Sq}\ (C_{k,}, Req(media\ type))$$
$$= [\Sigma_i\ Process\text{-}Time_{Sq}\ (C_{k,}, req_i)] / n \qquad (i = 1,\ldots n)$$

Performance Metric – Call Process Time



Performance Metric – Call Process Time

Event-Based Functional Scenario Path ***S*** *for a special type of media requests:*



System Call Process Time Metric for a call request *reqi*:

Process-Time(S, reqi)
= E_{m+1}'s outgoing timestamp – E_1's incoming timestamp

Performance Metric –Call Process Time

System Call Process Time Metric for a specific media load.

Req(media type) = { req_1, …., req_n }

Max-Process-Time (S, Req(media type))
= Max_i { Process-Time (S, req_i) } ***(i = 1,…n)***

Min-Process-Time(S, Req(media type))
= Min_i { Process-Time (S, req_i) } ***(i = 1,…n)***

Avg-Process-Time (S, Req(media type))
= [Σ_i Process-Time (S, req_i)] / n ***(i = 1,…n)***

Performance Metric – Latency

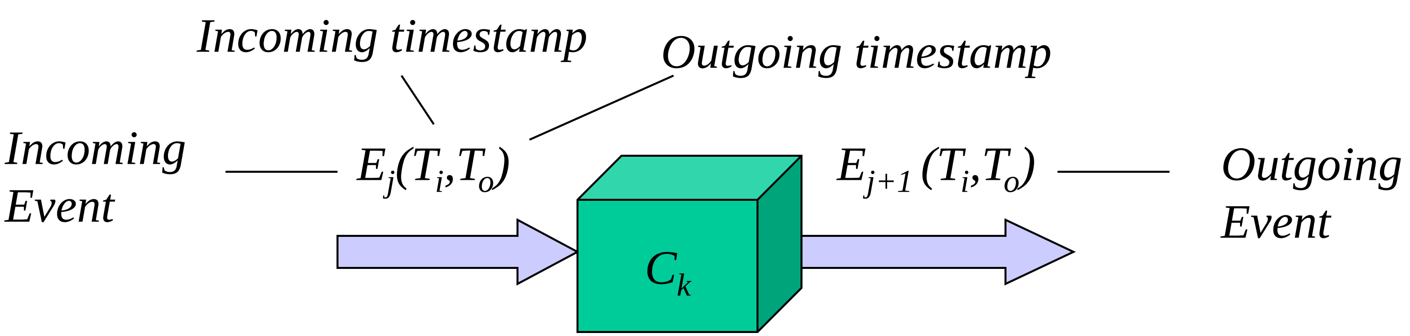
Latency metrics are used to measure the delays of:

- Messages, transactions, tasks, …..

We can define an event message latency metric for Portal V.5 to measure delay of event messages between components.

Event-Latency$_{Sq}$ (E_i, req$_j$)
= E_j's incoming timestamp – E_j's outgoing timestamp = T_i - T_o

We can compute the Max, Min, and Average of Latency.



Performance Metrics –Call Process Speed

Definition: The ***system call process speed*** for a special type of media requests ***Req(media-type)***. in a given performance test period T_p refers to the ratio of the total number of processed requests by the system to the test time $|T_p|$.

System Call Process Speed for a specific media load in a given test time T_p.

$$Req(media\ type) = \{ req_1, \ldots, req_n \}$$

$$System\text{-}Process\text{-}Speed_{Tp}(Req(media\ type))$$
$$= Total\ number\ of\ processed\ requests\ /\ |T_p|$$

Performance Metric – Call Process Speed

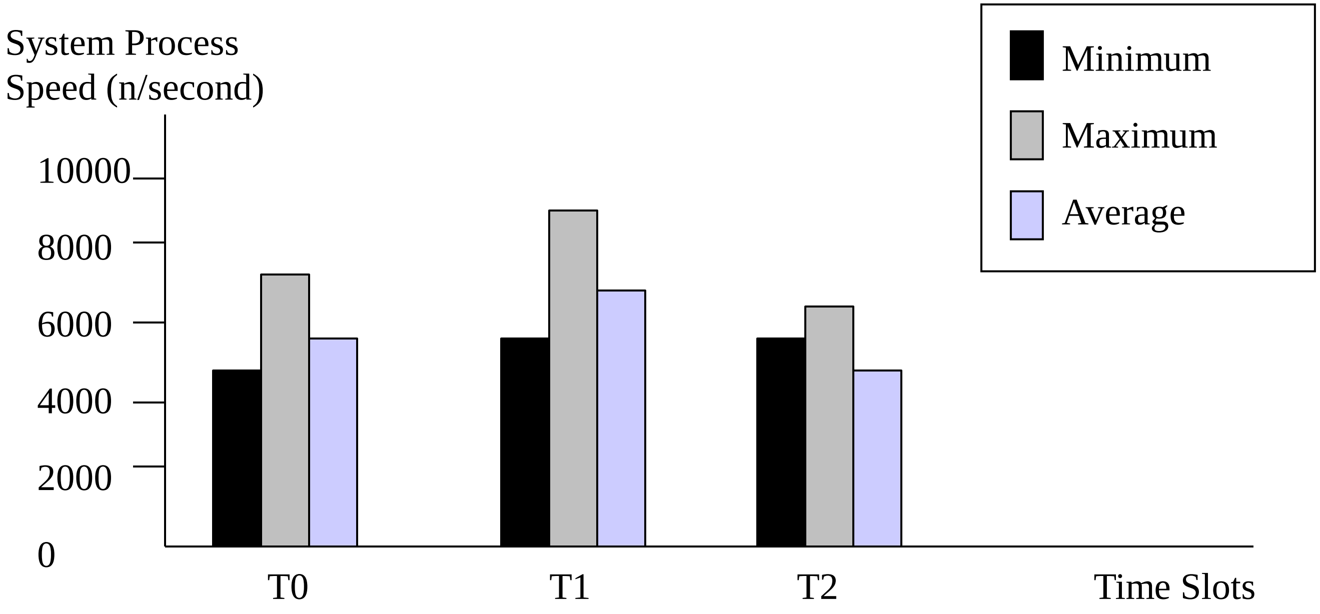


Figure 8(b). System Process Speed Measurement

Throughput Metrics

Objective: To measure the call processing capacity of a system.

The four types of throughput metrics are defined:

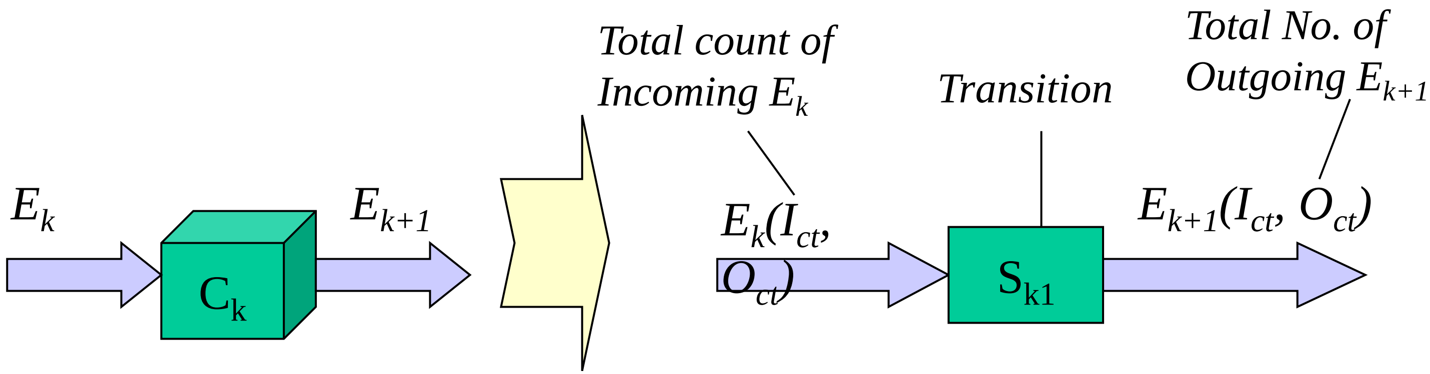
- ❑System processing throughput for a special type of media requests
- ❑System processing throughput for all types of media requests
- ❑System-load throughput rate for a special type of media requests
- ❑System-load throughput rate for all types of media requests

Concerns:

- ❑ ***Maximum of the throughput***
- ❑ ***Minimum of the throughput***
- ❑ ***Average of the throughput***

Throughput Metric
- Component Process Throughput

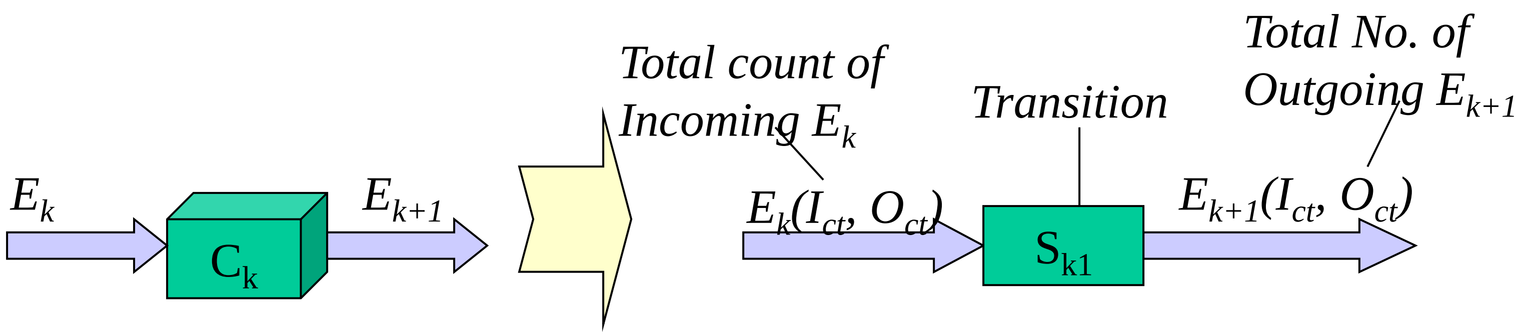
Definition: In a given functional scenario, the ***component-process throughput*** for a special type of media requests ***Req(media-type)*** in a given performance test period T_p refers to the total number of its outgoing events which are generated by processing its incoming events for these media requests.



System-Process-Throughput $_{Tp}$ ***(Req(media type))***
= total number of outgoing E_k

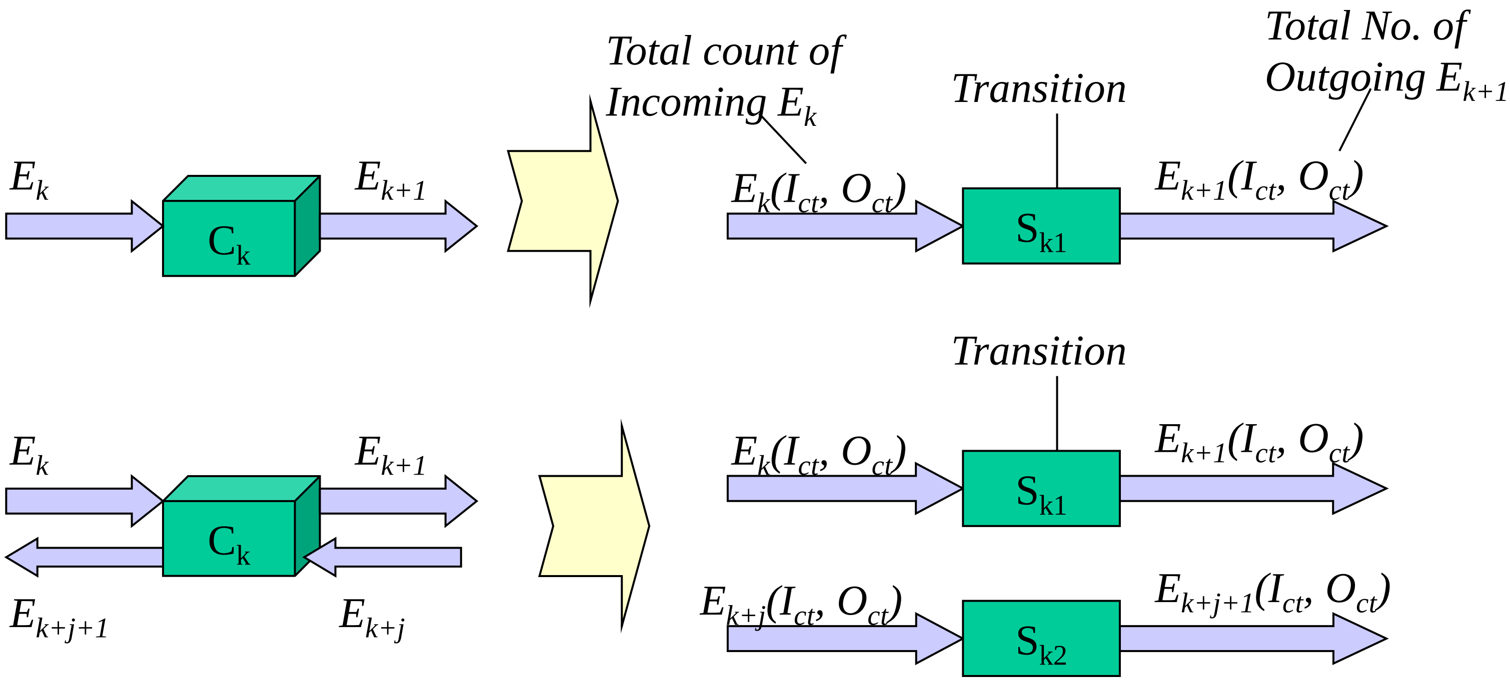
Throughput Metric
- Component Process Throughput Rate

Definition: In a given functional scenario, the ***component-process throughput rate*** for a special type of media requests ***Req(media-type)*** in a given performance test period $\boldsymbol{T_p}$ refers to the ratio of the total number of its outgoing events (which are generated by processing its incoming events for these media requests) to the total number of its incoming events.



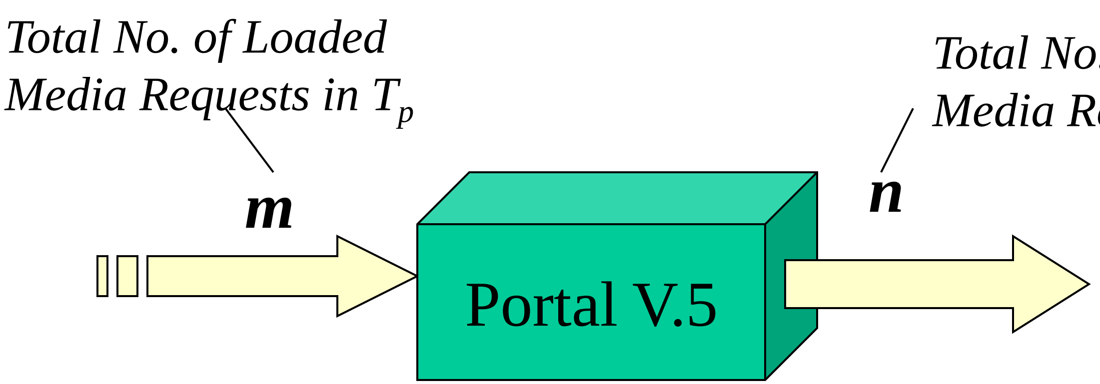
System-Process-Throughput-Rate $_{Tp}$ ***(Req(media type))***
= total number of outgoing E_{k+1} ***/ total number of incoming*** E_k

Throughput Metric
- Component Process Throughput



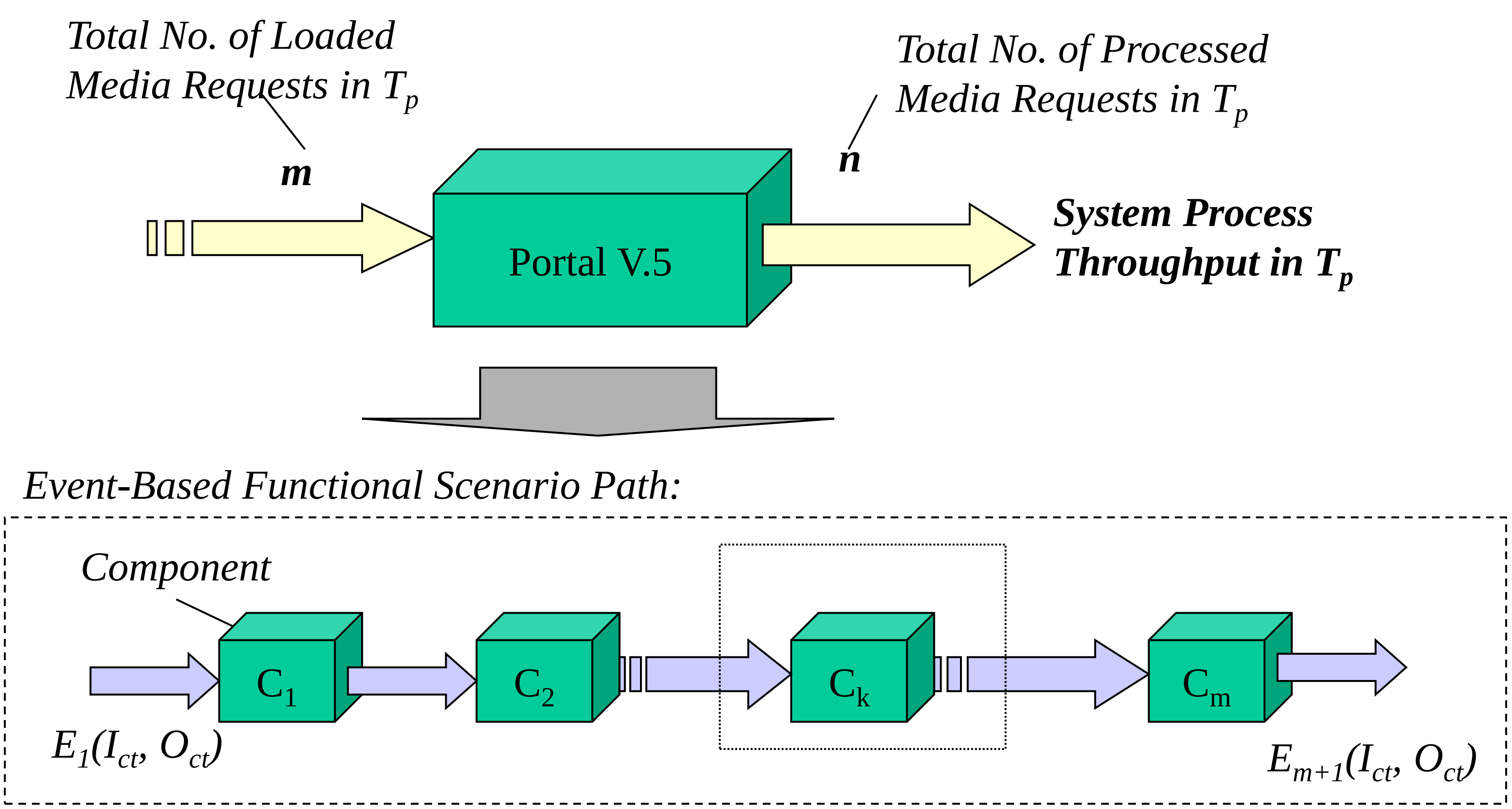
Throughput Metric - System Process Throughput

Definition: The ***system-process throughput*** for a special type of media requests ***Req(media-type)*** in a given performance test period T_p refers to the total number of processed requests in the incoming request set ***Req(media-type)***.



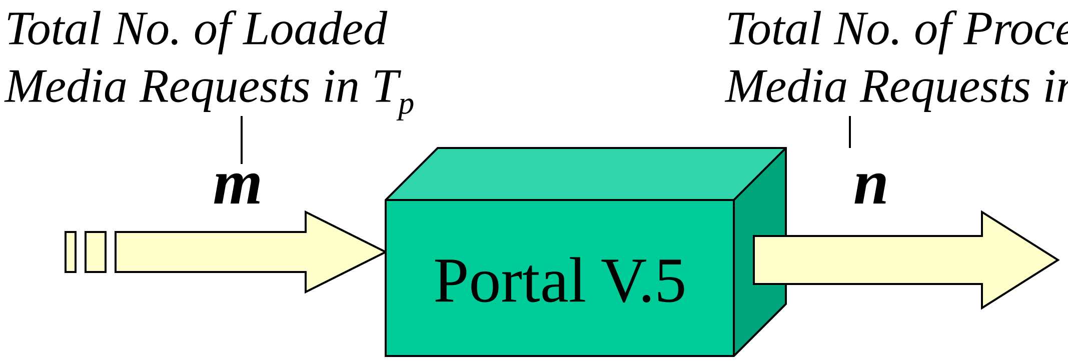
System Process Throughput in T_p = n

Throughput Metric - System Process Throughput



Throughput Metric – System Throughput Rate

Definition: The ***system-load throughput rate*** for a special type of media requests ***Req(media-type)*** in a given performance test load T_p refers to the ratio of the total number of processed requests to the total number of incoming requests in ***Req(media-type)***.



System Load Throughput Rate = n/m

Throughput Metric - Throughput

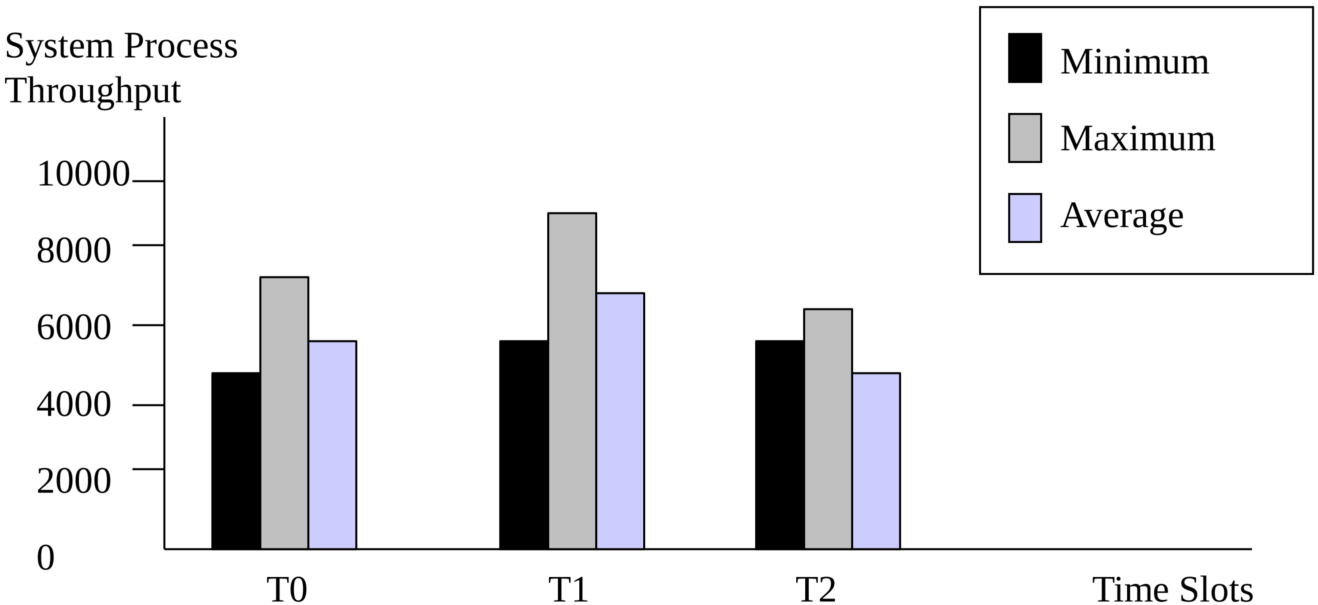


Figure 8(a). Throughput Measurement

Throughput Metric –Throughput Rate

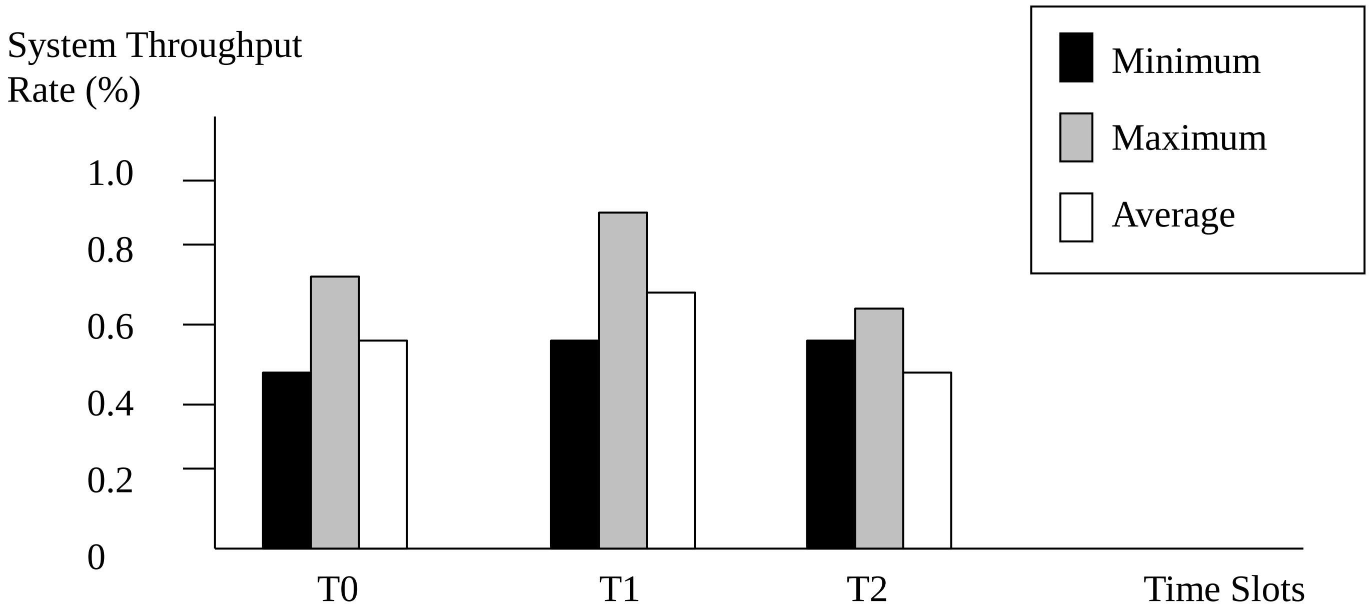


Figure 8(b). Throughput Rate Measurement

Availability Metrics

Objectives:

Availability metrics are defined to evaluate the availability of components and systems in providing their specified functions and services to system users and customers.

Several availability metrics are defined here:

- ❑Component-level
  - ❖Component Availability Metrics
  - ❖Component Service Availability Metrics
- ❑System-level
  - ❖System Availability Metric
  - ❖System-Service Availability Metric

Availability Metric - Component Availability

Definition: The *component availability* of a component C_k in a system during a time period T refers to the ratio of the total available time of the component to the total time T, including both available and unavailable time.

Component Availability Metric:

$$\textit{Component-Availability}_T(C_k) = \textit{available-time}(C_k) / (\textit{available-time}(C_k) + \textit{unavailable-time}(C_k)) = \textit{available-time}(C_k) / |T|$$

Where $\textit{available-time}(C_k)$ represents the available time of C_k during T_p, and $\textit{unavailable-time}(C_k)$ includes the time when the component can not provide the specified functions and services.

Application: to measure the availability of components.

Availability Metrics – HA Component Availability

For a HA component with a cluster of redundant components,
how to compute the available time or unavailable time of a HA component?

The first approach is to use the single failure criterion:

> "Under the single service failure criterion, any failure of its component service in a HA component is a service failure of the HA component, hence the unavailable time of a HA component refers to the time slots when at least one of its components is un-available."

Available-time (C) = Union $_k$ (unavailable-time (Ck)) (k = 1, …n)
Available-time (C) = |T| – unavailable-time (C)
Component-Availability $_T$(C) = available-time (C) / |T|

Availability Metrics – HA Component Availability

For a HA component with a cluster of redundant components,
how to compute the available time or unavailable time of a HA component?

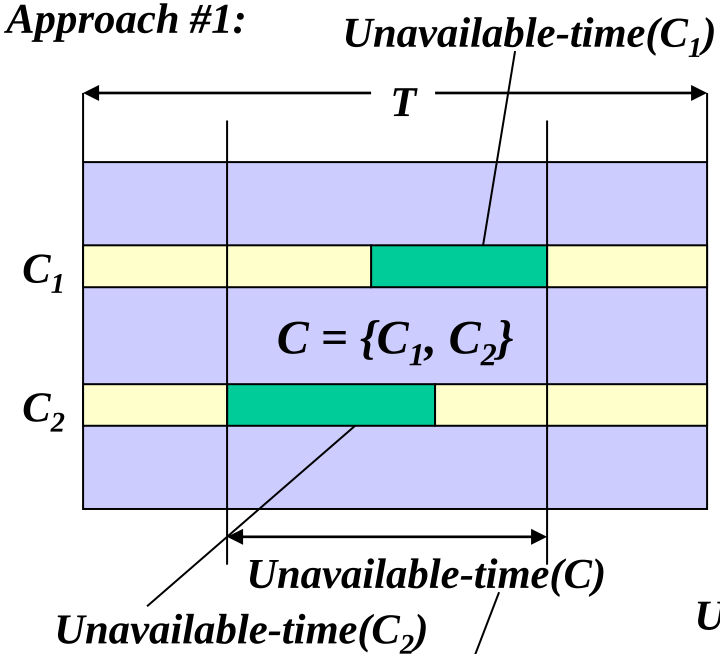
The other approach is to use the reliable service criterion:

> "Under the availability service criterion, when a HA component is up, at least one of its component is up and provides the available functions and services. Hence, the uptime of a HA component refers to the time slots where at least one of its components is up, and supports the specified functions and services."

Available-time (C) = Union $_k$ ***(available-time (Ck)) k = 1, …n)***
Component-availability $_T$***(C) = available-time (C) / |T|***

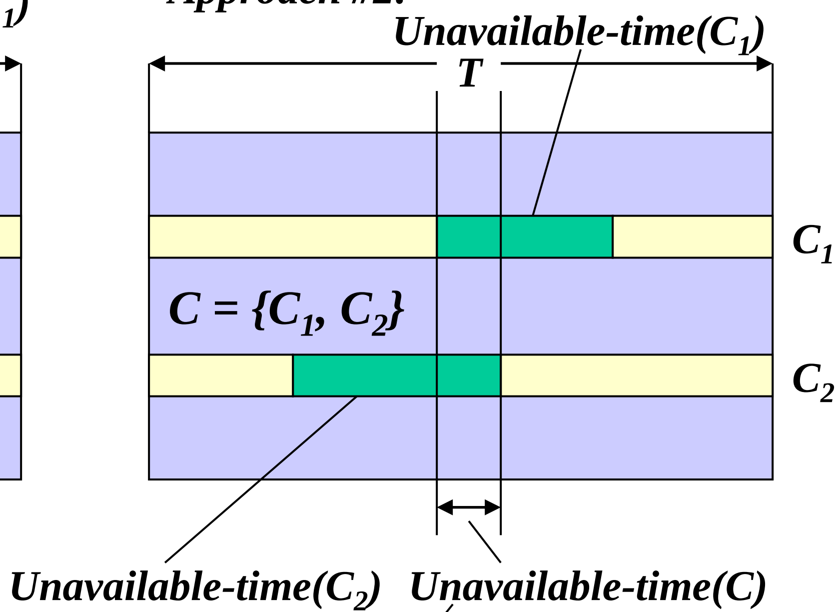
Comparison of Two Approaches

Approach #1:



Unavailable-time(C)
= unavailable-time(C_1) U unavailable-time(C_2)

Approach #2:



Unavailable-time(C)
= unavailable-time(C_1) intersect unavailable-time(C_2)

Availability Metrics - System Availability

Definition: For a system ***S***, its system availability during a given test period T_p refers to the ratio of its total available time to provide all specified function sets for its users in all given media types to the total test time $|T_p|$.

Let use ***System-Availability*** $_{Tp}$ ***(S)*** to denote the system availability for system ***S***. $S = \{C_1, \ldots, C_m\}$ consists of M components. To help engineers evaluate system availability, we define the following metric by considering the single failure criterion:

System-Availability $_{Tp}$ ***(S)***
= available-time(S) / (available-time(S) +unavailable-time(S))
= available-time(S) / $|T_p|$

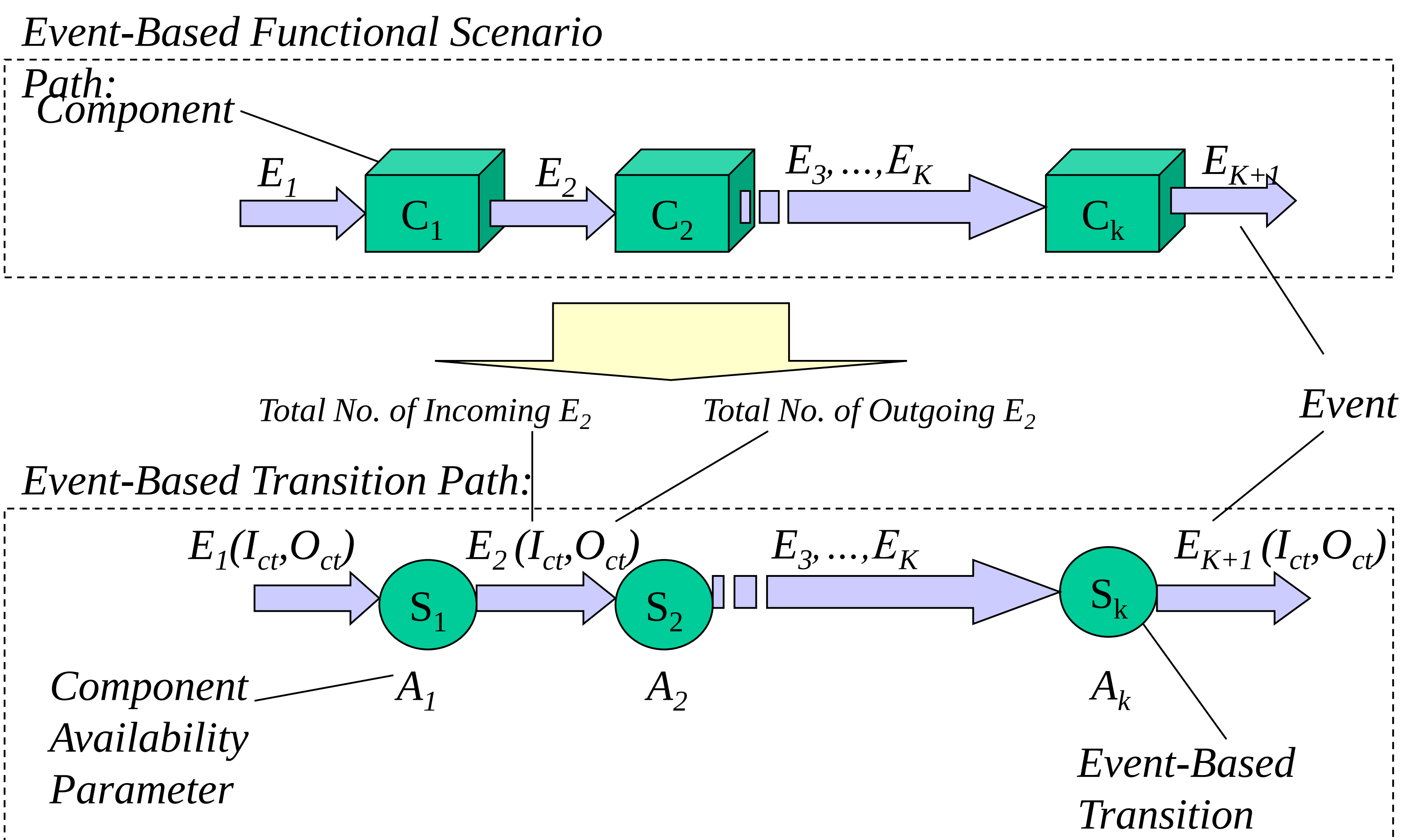
available-time(S) = Min $_k$ ***{ available-time*** (C_k) ***}*** $k = 1,..M$.

Where ***available-time (S)*** refers to the system available time in T_p, and ***un-available-time (S)*** represents the system un-available-time in T_P.

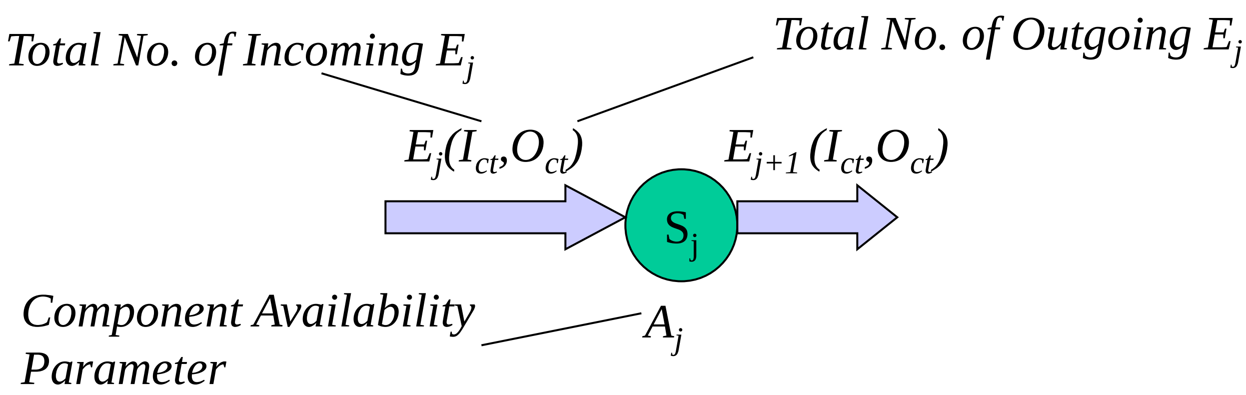
Availability Metric - System Service Availability

Definition: For a system ***S***, its ***system availability*** of function services during a given test period T_p to process user requests in a given media type ***Req(media type)*** is a function, denoted as ***System-Service-Availability(Req(media type),*** T_p***)***, which represents the probability of the system that is able to deliver the specified functional services to users in T_p.

System Service Availability Model



Availability Metric - System Service Availability Metric



System Service Availability Metric:

$$\textit{System-Service-Availability}(Req(\text{media type}), T_k) = A_1 * A_2 * \ldots * A_m$$

$$A_j = \begin{cases} 0 \text{ if there is no incoming event } E_j \text{ during test time T. Otherwise,} \\ \text{total number of outgoing } E_{j+1} \text{ / total number of incoming } E_j. \end{cases}$$

Availability Metrics

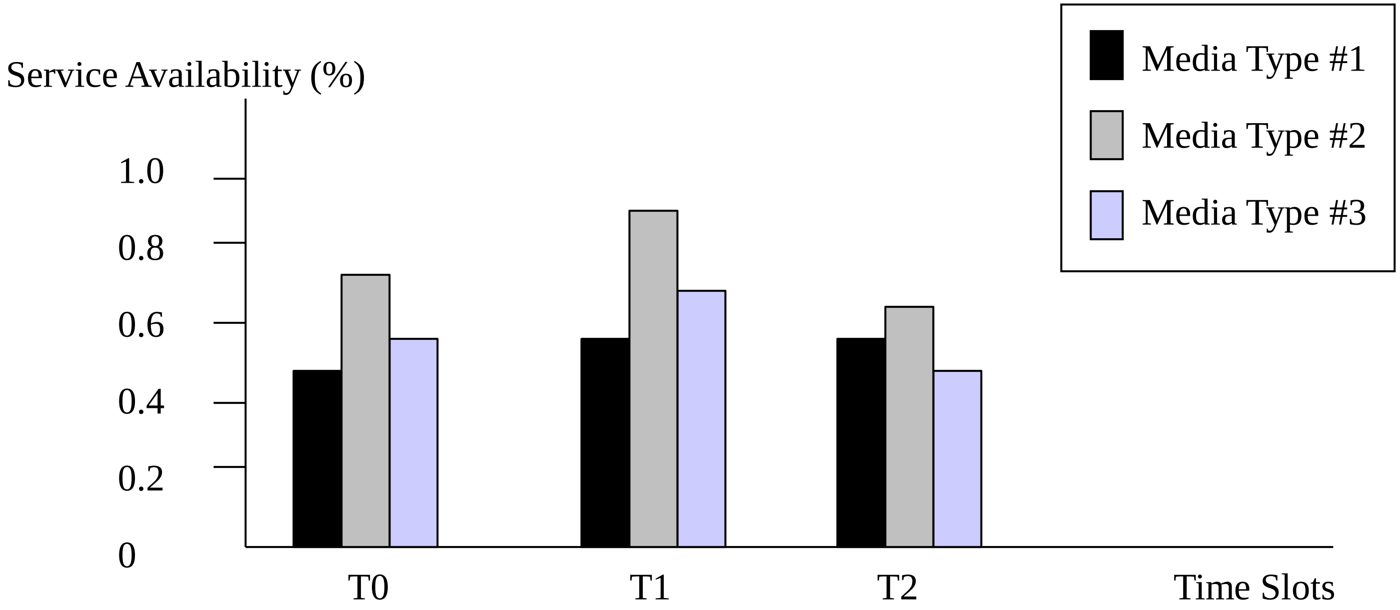


Figure 15. System Service Availability Measurement

Reliability Metrics

Objectives:

Reliability metrics are defined to evaluate the system reliability in a given time to support the reliable functions and services to system users.

Several availability metrics are defined here:

- ❑Component-level
  - ❖Component Reliability Metric
  - ❖Component Service Reliability Metric
- ❑System-level
  - ❖System Reliability Metric
  - ❖System-Service Reliability Metric

Reliability Metrics - Component Reliability

Definition: The *component reliability* of a component C_k in a system during a time period T refers to the ratio of the total uptime of the component to the total time, including both uptime and downtime.

Component Reliability Metric:

Component-Reliability $_T$ ***(C$_k$)***
= up-time(C$_k$) / (up-time(C$_k$) +down-time(C$_k$))
= up-time(C$_k$) / |T|

Where ***up-time(C$_k$)*** represents the up time of C_k during T_p, and ***down-time(C$_k$)*** includes the down time and recovery time of C_k.

Application: to measure the reliability of components.

Reliability Metrics – HA Component Reliability

For a HA component with a cluster of redundant components, how to compute the uptime or downtime of a HA component?

The first approach is to use the single failure criterion:

> "Under the single failure criterion, any failure of its component in a HA component is a failure of the HA component, hence the downtime of a HA component refers to the time slots when at least one of its components is down."

downtime (C) = Union $_k$ (downtime(Ck)) (k = 1, …n)
uptime (C) = |T| – downtime(C)
Component-Reliability $_T$(C) = uptime (C) / |T|

Reliability Metrics – HA Component Reliability

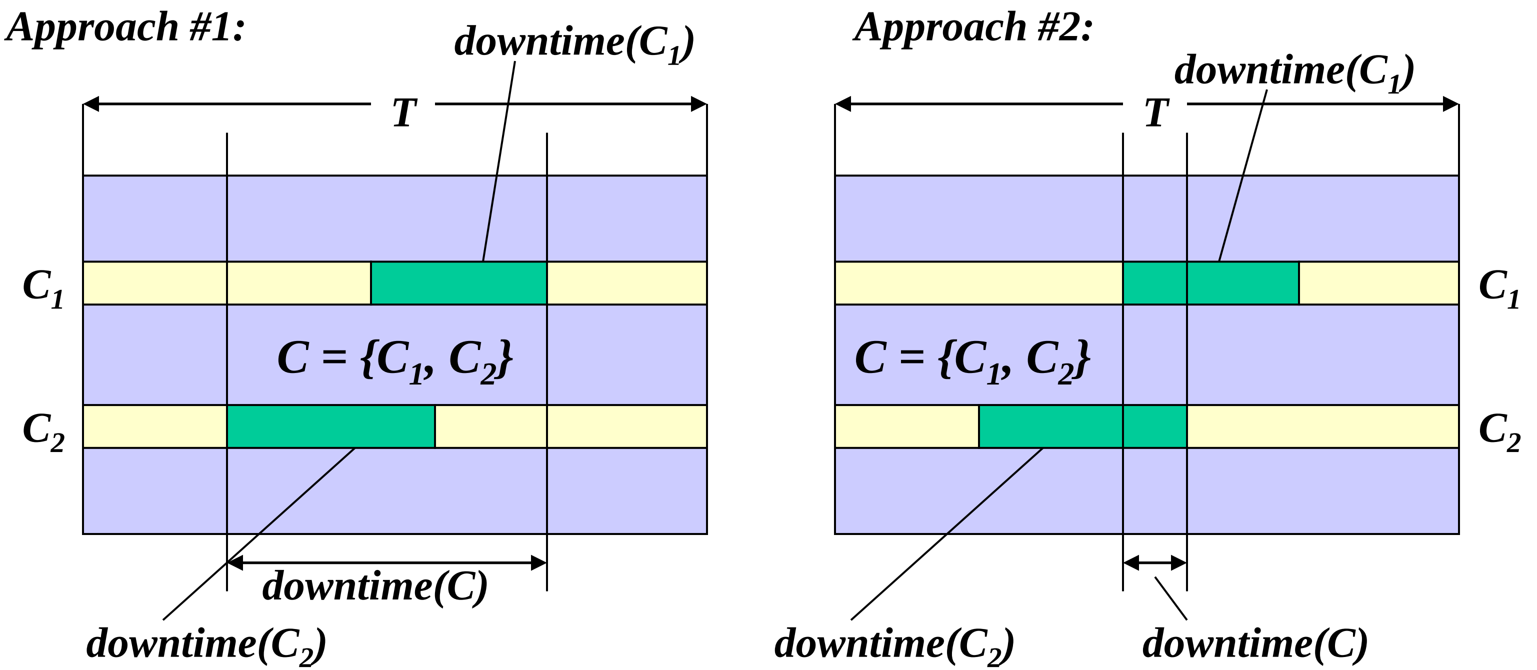
For a HA component with a cluster of redundant components,
how to compute the uptime or downtime of a HA component?

The other approach is to use the reliable service criterion:

> "Under the reliable service criterion, when a HA component is up, at least one of its component is up and provides the reliable functions and services. Hence, the uptime of a HA component refers to the time slots where at least one of its components is up, and supports the specified functions and services."

uptime (C) = Union $_k$ ***(uptime(Ck))*** ***(k = 1, …n)***
Component-Reliability $_T$***(C) = uptime (C) / |T|***

Comparison of Two Approaches



downtime(C)
= downtime(C_1) U downtime(C_2)

downtime(C)
= downtime(C_1) intersect downtime(C_2)

Reliability Metrics - System Reliability

Definition: For a system ***S*** with a single failure criterion, its system reliability during a given test period T_p refers to the ratio of its total time to support all specified function sets for its users through all given media types to the total test time $|T_p|$.

Let use ***System-Reliability*** $_{Tp}$ ***(S)*** to denote the system reliability for system ***S***. ***S = {C_1,…., C_m}*** consists of M components. To help engineers evaluate system reliability, we define the following metric by applying single failure criterion:

System-Reliability $_{Tp}$ ***(S)***
= uptime(S) / (downtime(S) +uptime(S)) = uptime(S) / |T_p|
Uptime(S) = Min $_k$ ***{ uptime(C_k) }*** ***k = 1,…M.***

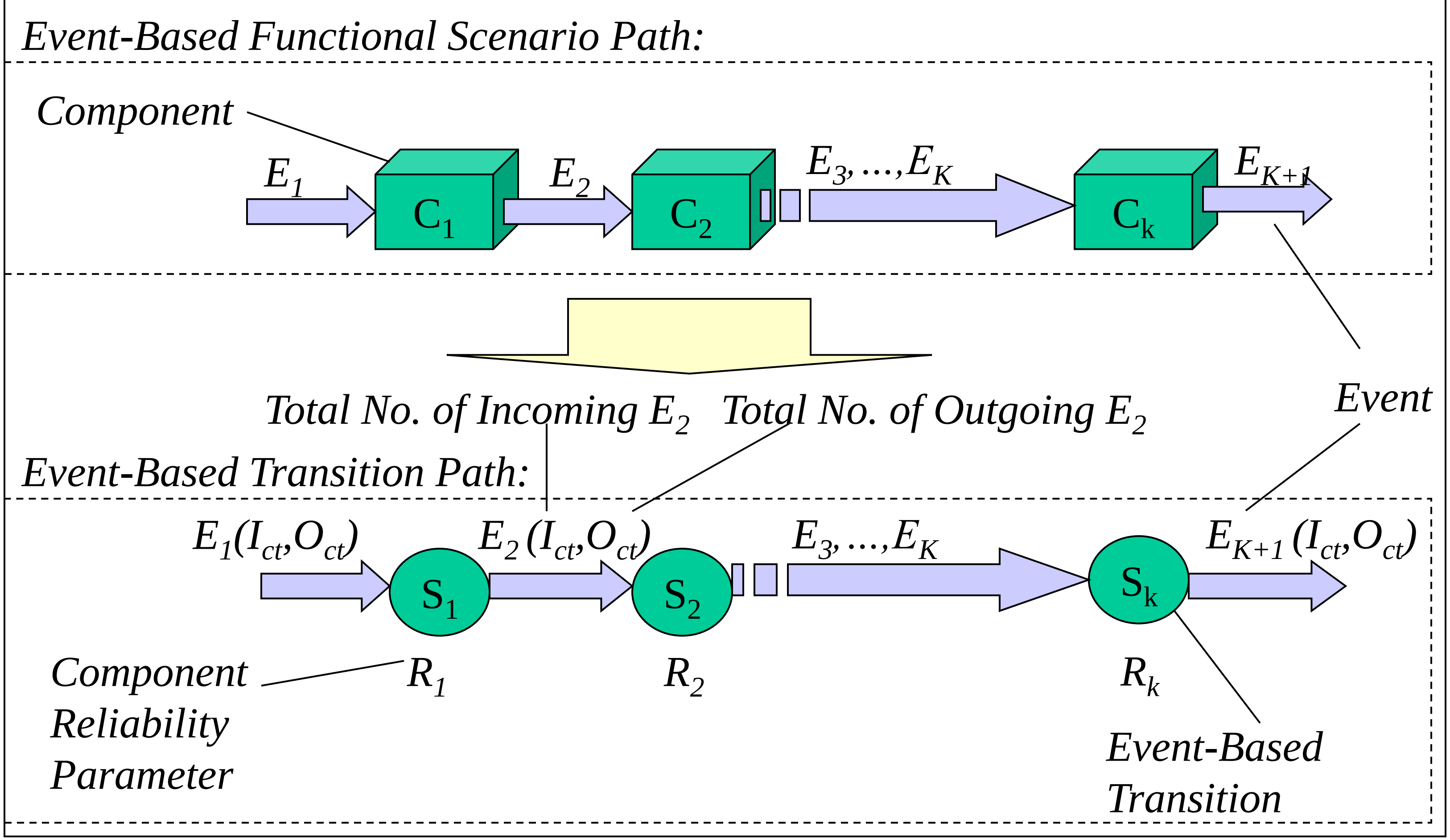
Where ***uptime(S)*** refers to the system uptime in T_p, and ***downtime(S)*** represents the system downtime in T_P.

Reliability Metrics - System Service Reliability

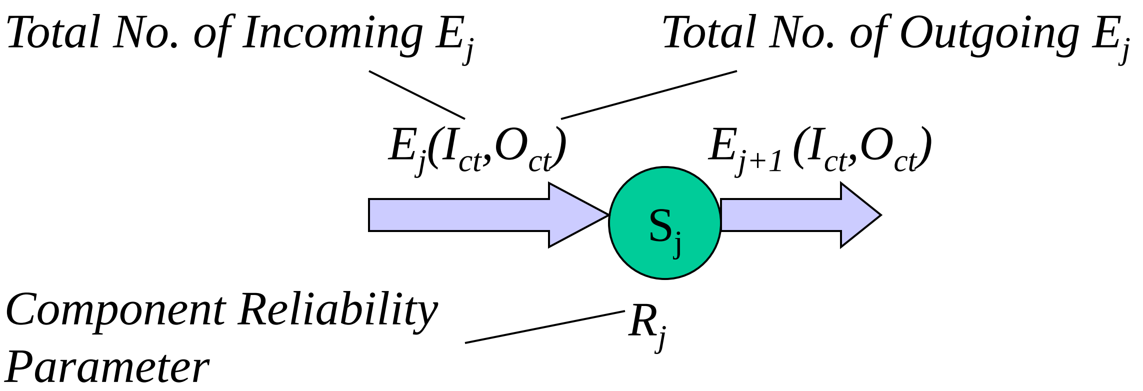
Definition: For a system ***S***, its ***system reliability*** of function services during a given test period T_p to process user requests in a given media type ***Req(media type)*** is a function, denoted as ***System-Service-Reliability(Req(media type),*** T_p***)***, which represents the probability of the system that is reliable enough to deliver the specified functional services to users in T_p.

Application: evaluate the system reliability on providing the specified functional services to process the requests from users using a given media type.

System Service Reliability Model



Reliability Metrics - System Service Reliability Metric



System Service Reliability Metric:

System-Service-Reliability(Req(media type),T_k)

$$= R_{1*} R_2 * \ldots * R_m$$

R_j = { **0 if there is no incoming event Ej during test time T. Otherwise,**

total number of outgoing E_{j+1} / total number of incoming E_j.

Reliability Metrics - System Service Reliability

Now let's examine this metric in the following special cases:

- If there is R_j with 0 value, then ***System-Service-Reliability(Req(media type),T_k)*** must be 0**.** This indicates that at least one component on the path ***P*** is not available during the time period of ***T***.
- If all R_j ***(j=1,.., m)*** with value of 1, then ***System-Service-Reliability(Req(media type),T_k)*** = 1.

The system service reliability during the test period T_p can be evaluated below.

System-Service-Reliability(Req(media type),T_p)

= Σ_k System-Service-Relilability(Req(media type), T_k)

(k =1,... z)

$= \Sigma_k (\Pi_j R_j) / z$ ***(j = 1,…, m, k =1,…, z)***

System Service Reliability

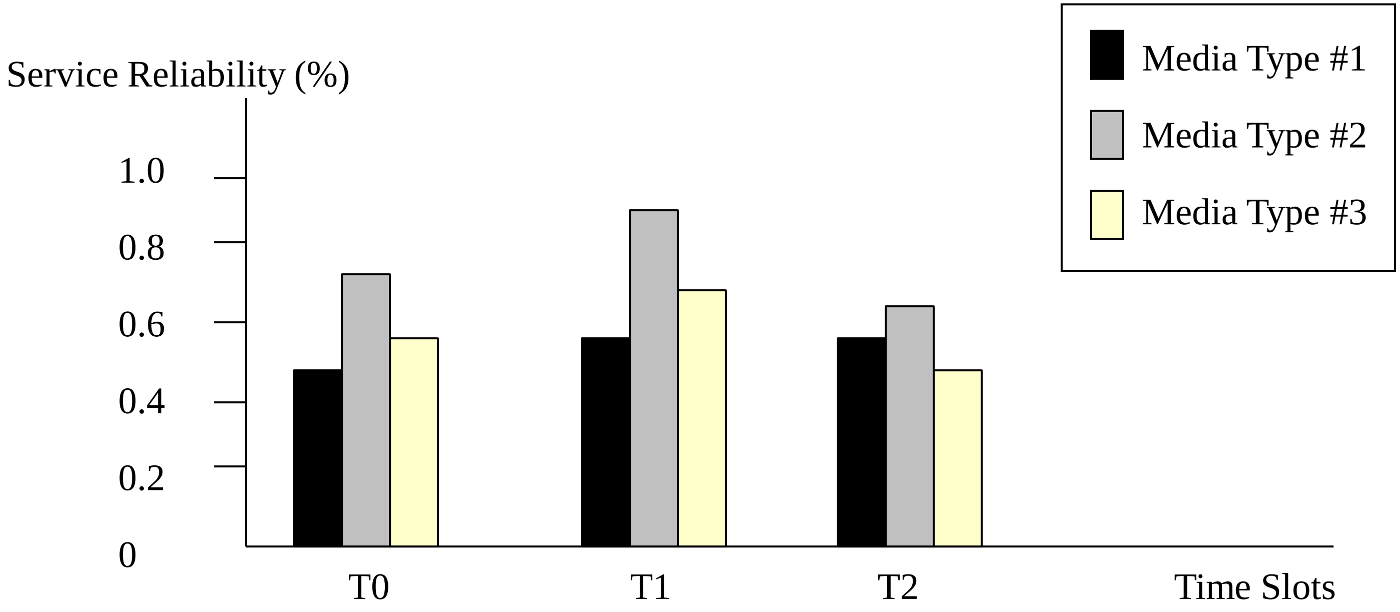


Figure 15. System Service Reliability Measurement

Scalability Metrics

The major applications of scalability metrics:

- To check the boundary and threshold of a system and its components.
- To check the speed-up of a system and its components when more hardware machines and application servers are added into a system.
- To evaluate the throughput improvement of a system (or its components) when more hardware machines and application servers are added into a system.

The scalability metrics defined here:

- Speed-up
- Throughput Improvement

Load Boundary and Threshold

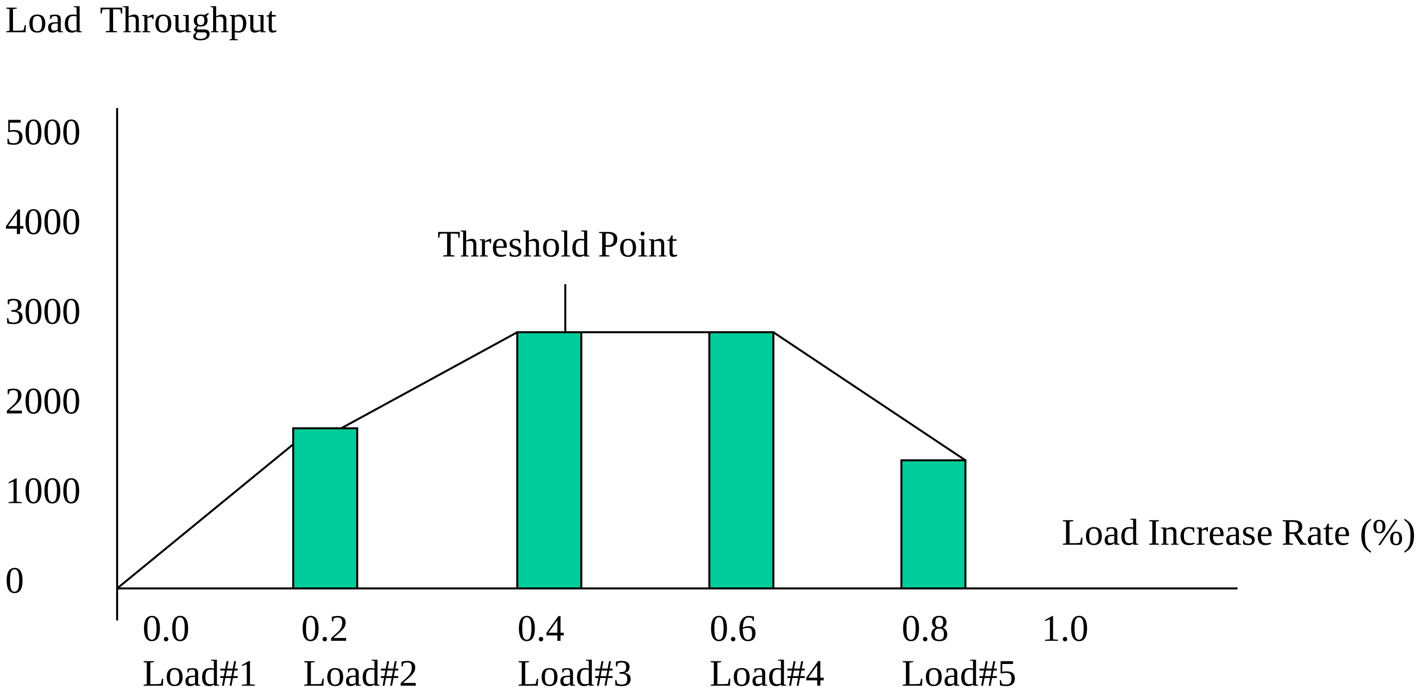


Figure 16. Load Boundary and Threshold

Throughput Improvement

Definition: The ***throughput improvement*** for a system (or a component cluster) for a fixed load ***Req(S)*** in a given time T_p, denoted as ***Throughput-Improvement(A,B)*** refers to the throughput rate increase from configuration setting ***A*** to configuration setting ***B***. The detailed metric can be defined below:

Throughput-Improvement (A,B)
= (system load throughput under B
– system load throughput under A)/|Req(S)|

Application: to measure the improvement of system throughput

Throughput Improvement

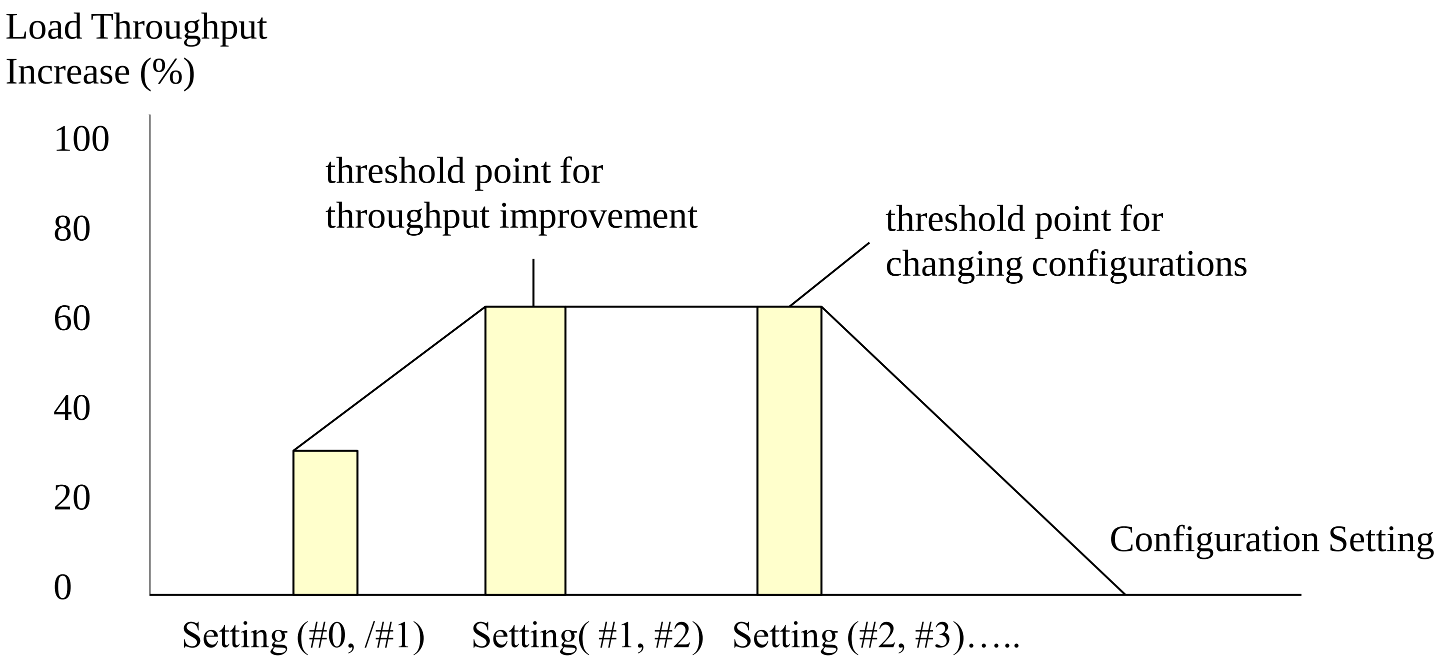


Figure 17. Throughput Improvement

Speed-Up

Definition: The process ***speed-up*** for a system (or a component cluster) under a fixed load ***Req(S)*** in a given time T_p, denoted as ***Speed-up(A,B),*** refers to the process speed increase from configuration setting ***A*** to configuration setting ***B***. The detailed metric can be defined below:

Speed-up (A,B)
= (process speed under B – process speed under A)/ process speed under A

Application: to measure the improvement of the system process speed

Speed-Up

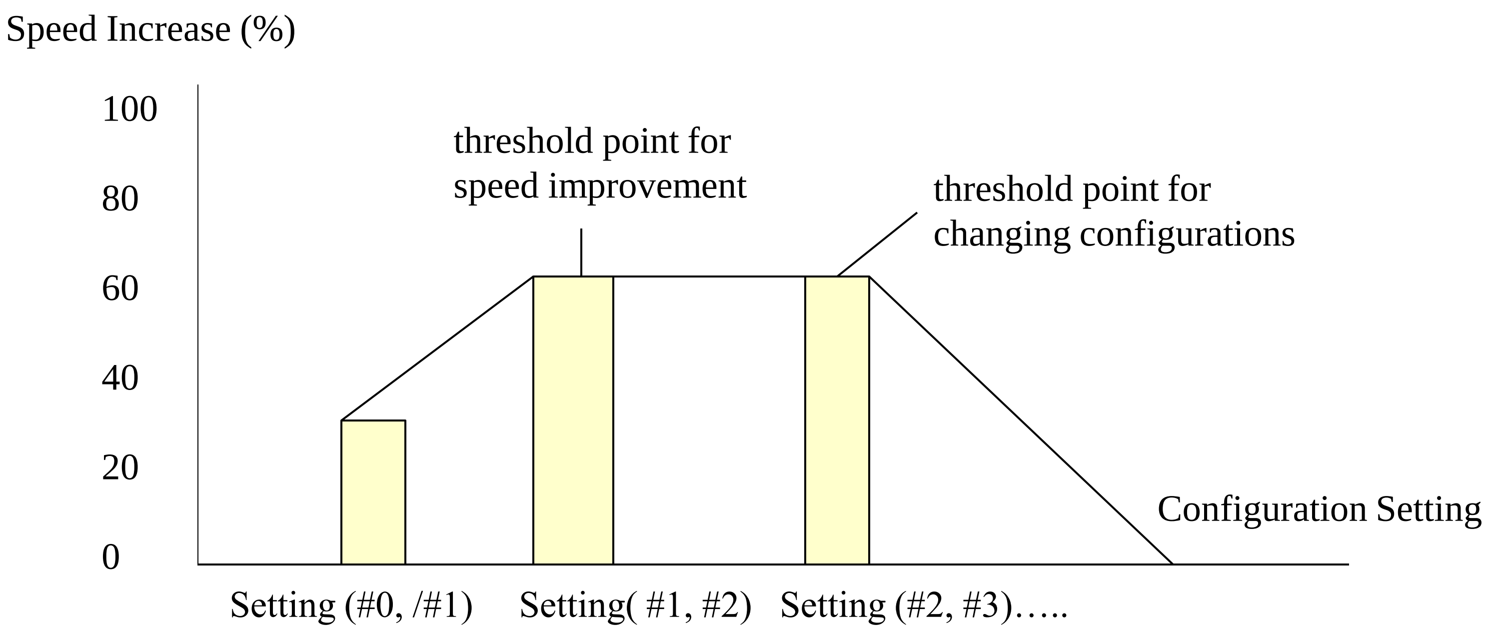


Figure 18. Speed-up

Utilization Metrics

The major applications of utilization metrics:

- **Network utilization:**
  To check the network utilization of different protocols at a specific node on a given network.
  Example: HTTP. H.323, TCP/IP, ASAI, SMTP/POP, ..

- **Server machine utilization:**
  To check the utilization of system resource of each server machine, including CPU, RAM, Cache, and disk.
  Example:DMS, MDR, eFlow, ASP servers and clusters

- **Client machine utilization:**
  To check the utilization of system resources of a client machine, including CPU, RAM, Cache, and disk.
  Example: Agent and caller machine.

Utilization Metrics

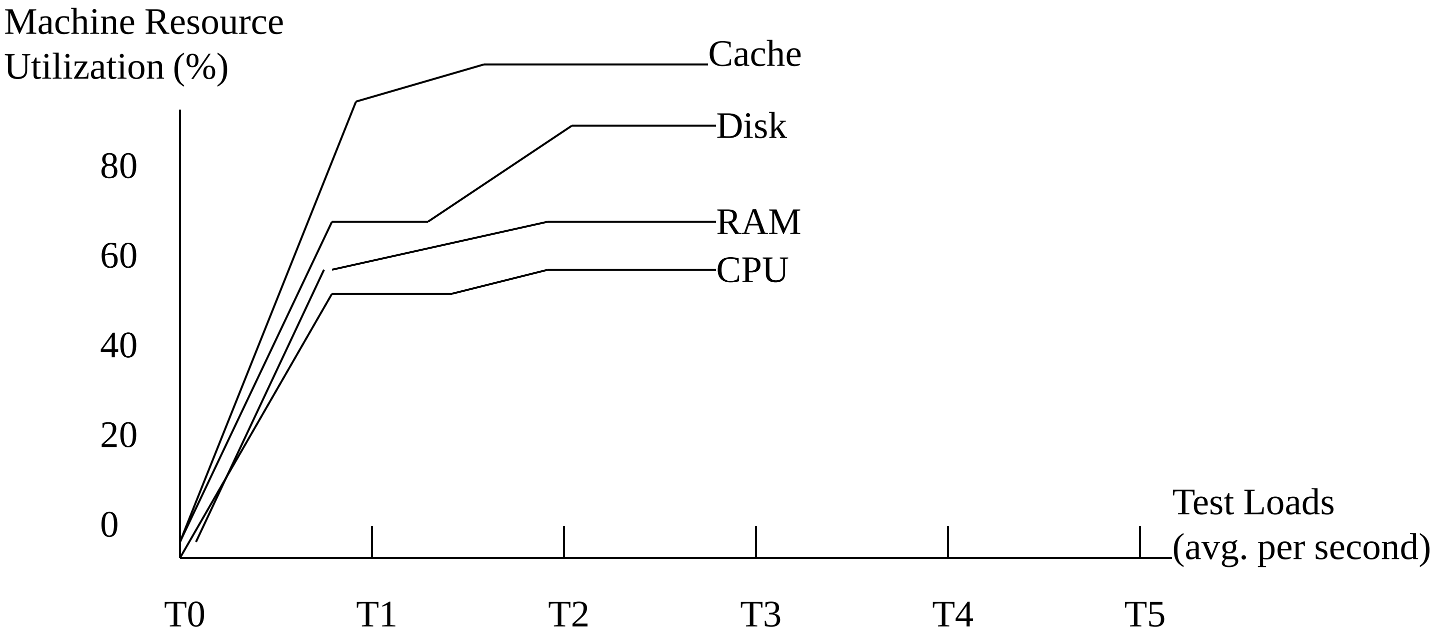


Figure 20. Utilization of Single Client/Server Machine

Utilization Metrics

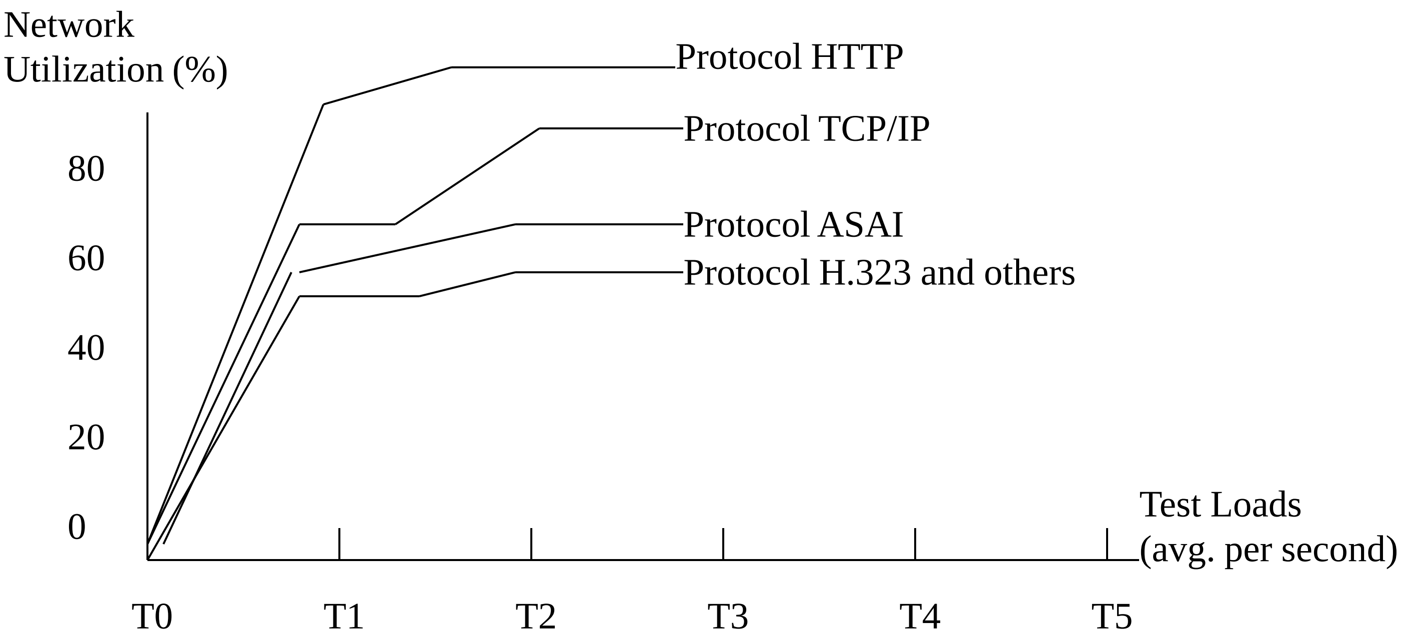


Figure 19. Network Utilization

Supporting Environment for Performance Evaluation

The major applications of utilization metrics:

- **Basic Components**
  To check the network utilization of different protocols at a specific node on a given network.
  Example: HTTP. H.323, TCP/IP, ASAI, SMTP/POP, ..

- **System Infrastructure:**
  To check the utilization of system resource of each server machine, including CPU, RAM, Cache, and disk.
  Example:DMS, MDR, eFlow, ASP servers and clusters

- **Construction Step:**
  To check the utilization of system resources of a client machine, including CPU, RAM, Cache, and disk.
  Example: Agent and caller machine.

References

Jerry Gao and Tim Lui, “XXXXX Performance Test Environment for Portal V.5 Products”, July, 2000.

Jerry Gao “Performance Test Strategy and Test Metrics for XXXX Portal V.5 Products”, August, 2000.

Jerry Gao, “A Set of Performance Notes”. July, 2000.

Jerry Gao, “Performance Testing and Measurement for Distributed Component-Based Systems: A Systematic Approach”, in March 2001.

A Collection of Technical Research Papers on Performance Evaluation and Metrics.