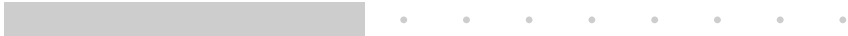




## Performance and Scaling in e-commerce Systems



A. Buchmann  
DVS, Fachbereich Informatik  
T.U. Darmstadt  
buchmann@informatik.tu-darmstadt.de



## Goals

- To learn how to design reliable, highly available, performant and scalable e-commerce systems
- Understand the underlying
  - Models
  - Technologies
  - Architectures
  - Performance models and metrics
  - Capacity planning and forecasting methods
- Apply the principles learned to case studies



## Motivation

- Performance is everything!
  - Your competition is only one mouse-click away
  - Empirical knowledge/belief: if page is not downloaded within 8 sec. user tends to abandon page
- Availability is important!
  - Unavailability of site is front-page news
  - If site is unavailable effect may be felt in stock-price (e-Bay)
  - Denial of service attacks cost millions
  - Security affects both performance and availability and should be considered from the beginning



## Motivation 2

- Reliability is critical!
  - orders and payments should never be lost
  - commercial interactions must be transactional
  - commercial transactions must be reproducible/auditable
- Many e-commerce systems must include existing (legacy) systems
  - .com systems are often built from scratch
  - e-commerce solutions of existing companies must interact with existing commercial systems



:

### Motivation 3

- Ability to evolve is essential
  - usage and growth patterns are difficult to predict
  - systems must be designed as modular as possible
  - hardware and software expansions take time!
- Up-front sizing is necessary but difficult
  - up-front sizing determines initial investment
  - oversized system drives costs up
  - undersized system may not perform or may not keep up with growth
  - potential bottlenecks must be identified and eliminated

. . . . .5

:

### The Need for Quantification

- Critical quality of service questions cannot be answered by qualitative reasoning alone
- Bottlenecks appear in often unexpected places
- Bottlenecks can only be attacked if quantified
- Investments must be justified by hard facts
- Lack of quantification caused many of the .com's problems
- The bla-bla days are over!

. . . . .6

:

### SPE - Software Performance Engineering

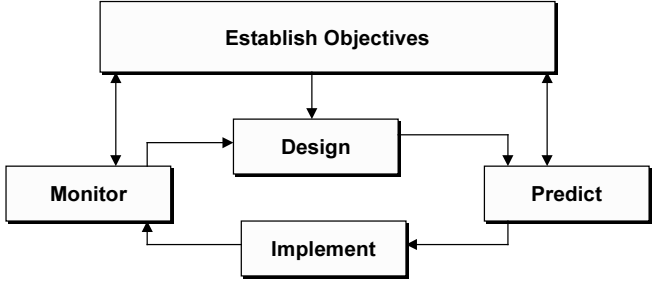
- Workgroup Client/Server Computing is different from Enterprise Client/Server Computing
  - single platform vs. multiplatform
  - high speed LAN vs. WAN or multinetworks
  - reasonably small workloads vs. high workloads
- Distribution of functionality and data exacerbates any performance problem
- Performance is integral part of quality in mission-critical systems ==> design for performance!

. . . . .7

:

### Scope of SPE

- Design and predict performance for initial sizing
- Monitor running system to detect (potential) bottlenecks and take corrective action (tuning/redesign)



. . . . .8

:

## Scope of SPE: design and initial sizing

- Design new systems for performance
  - Define acceptable performance levels
    - unless we have an objective, we can never have a performance problem **Service Level Agreement**
  - Design software architecture considering resource limitations and performance objectives
    - logical layering - logical layering mistakes cannot be easily resolved at the physical level
    - balance the needs of users, applications and organization
    - instrument systems to monitor performance
  - Perform initial sizing based on approximate performance models

.....9

:

## Service Level Agreements

- The parties
  - organizations, people representing orgs., date
  - reliability of service
  - acceptable limitations
- Prologue
  - intent, assumptions that form context, revision process, commitments
  - how service level is measured
  - how compliance is scored
  - how service level is reported
- Service details
  - service to be provided
  - timeliness reqs. of service
  - volume of demand for service over time
  - accuracy requirements
  - availability of service
  - Financial terms
    - compensation for providing s.
    - refunds/discounts for lower s.
    - additional compensation good s.
    - method of resolving disputes
  - Duration
    - renegotiation and termination

.....10

:

## Scope of SPE: monitoring and tuning

- Monitor ongoing software performance
  - monitor during standard operation to have a baseline to compare against, look for trends
  - begin monitoring during a crisis compounds problems
  - three level monitoring strategy (continuous exception monitoring, regular performance tracking, occasional performance audit)
  - make sure loads, resource requirements and performance can be measured
  - monitoring components can't reveal response t. problems
- Identify biggest bottleneck and fix it - repeat until done
- Today's peak is tomorrow's norm

.....11

:

## The Cost of Performance Fixes

	Bus. Req's.	System Req's.	Logical design	HW/SW design	HW/SW environ.	Component perf.	System perf.	
Planning	x							
Analysis	x	x						1 000 - 5 000
Schematic design	x	x	x					5 000 - 25 000
Technical design	x	x	x	x				25 000 - 100 000
Construction	x	x	x	x	x			100 000 - 500 000
Deployment	x	x	x	x	x	x		500 000 - 1 000 000
Production	x	x	x	x	x	x	x	???????????????

.....12

:

## Perception of Response Time

- Response time is subjective:
  - inordinate influence of longest response time observed
  - perceived average response time is 90th percentile of response time distribution
  - minimize response time variation first (not average)
  - users are seldom aware of improvements of less than 20% in average response time
  - avoid response time variations in new applications (if necessary, build in initial delays)
  - generate partial responses fast
  - use asynchronous tasks to reduce perceived resp. time

. . . . .13

:

## Checklist of Potential Performance Factors

- Average/peak transaction volumes
- number of customers
- application logic
- use of resources (processor, storage, net)
- business growth
- application architectures
- partitioning application between client & server
- partitioning application across servers
- Database size
- table size
- database design
- normalization/multitable joins
- database locking
- indexing, clustering, partitioning
- logical vs. physical I/Os
- buffering or caching
- query result set sizes
- copy management
- DBMS-controlled replication

. . . . .14

:

## Performance Factors (cont.)

- Data refresh volumes
- DBMS and OS platforms
- DBMS tuning choices
- OS tuning options
- middleware
- access to legacy systems via gateways
- transaction monitors
- message queuing middleware
- DBMS/HW compatibility
- HW platform (parallelism, processor speed, multiprocessors)
- LAN performance
- WAN performance
- impact of other workloads
- impact of security
  - firewalls
  - authentication and third party software
- buildup of pages (applets, XML, ads, multimedia)

. . . . .15

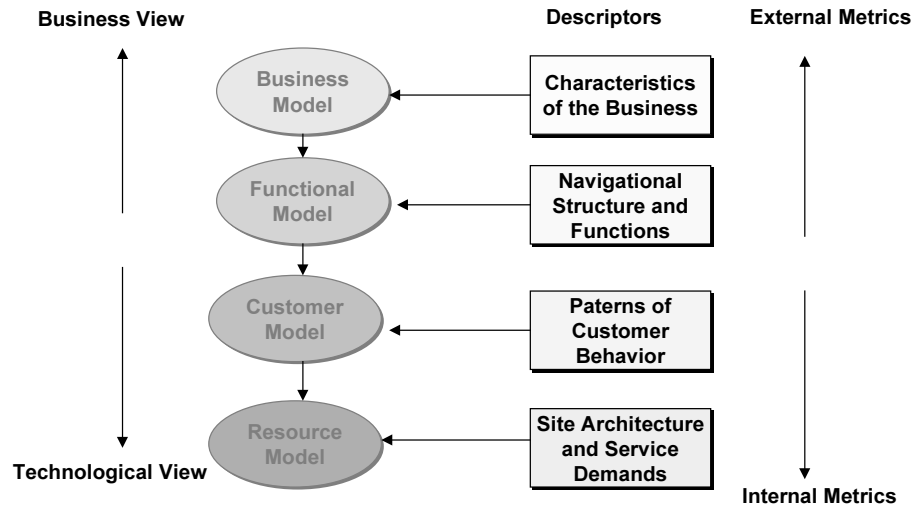
:

## Framework for Quantitative Analysis in E-commerce

- Framework proposed by Menascé and Almeida
- Four modeling levels:
  - e-business model
  - functional model
  - customer behavior model
  - IT resource model
- Models of different level of detail can be provided, especially for customer behavior, workload and individual IT resources (network, web-server, authentication server, TP monitor, application server, DB-server, etc.)

. . . . .16

## Reference Models for Electronic Business



.17

## E-Business and E-Commerce

- E-business is the more comprehensive term
- E-business subsumes e-commerce
- E-commerce originally used to refer to commercial transactions carried out over a network (generally assuming use of the WWW on the Internet)
- Many typical e-commerce applications also valid on intranets (e.g. info dissemination, logistics, supply chain management, e-publishing, etc.)
- Because of strong user variability e-commerce applications on the Internet more interesting from performance and scalability perspective

.18

## E-Business/E-Commerce Models

- Classification based on the parties involved (X-to-Y)
  - Business to Consumer (B2C)
  - Business to Business (B2B)
  - Business to Administration (B2A or B2G)
  - Consumer to Administration (C2A or C2G)
  - Consumer to Consumer (C2C)
  - Business to Business to Consumer (B2B2C or B2I2C) based on Intermediaries
- M-commerce follows above basic models with additional constraints (mobile users/devices, limited bandwidth and device capabilities, security, etc.)

.19

## B2C

- Interaction between business and consumer is characterized by
  - large volume of customers
  - high variability in access patterns (daily/seasonal peaks)
  - small volume of transactions per customer
  - spontaneous user reactions (abandoned shopping carts, impatience with slow performance, etc.)
  - mostly catalog access, browsing, selection, payment
  - order fulfillment may be immediate (e.g. SW downloads), deferred by seller directly (e.g. HW, tickets) or via third party (e.g. books, tickets)

.20

:

## B2B

- Business to Business interactions are characterized by:
  - longer standing business relationship/partnership
  - B2B usually part of bigger integration process (virtual enterprises, supply chain management)
  - integration of business processes more important than direct sales
  - negotiation is often part of B2B process
  - even in the case of direct sales, business gets different treatment (e.g. credit line, volume discounts, etc.)

.....<sup>21</sup>

:

## B2A and C2A

- Interactions with administration/government are characterized by
  - attempt of governments to be more responsive to user needs (opening hours)
  - need to compensate for staff cutbacks and rising costs
  - staged efforts: information only, payments, submission of whole transaction (e.g. tax declarations w. refunds and notification by e-mail)
  - procurement channel for government: requests for proposal, submission of bids, auctions

.....<sup>22</sup>

:

## C2C

- Consumer to consumer interactions are characterized by
  - prevalent business model are auctions, music and other exchanges on the rise
  - large volumes of users and transactions
  - high percentage of volume is due to monitoring of auctions (polling)
  - high sensitivity to down-time/availability of site
  - auction site usually only provides the platform (catalog, bidding mechanisms)
  - fulfillment/payment done directly by seller/buyer (exception: escrow service)

.....<sup>23</sup>

:

## B2B2C and Marketplaces

- Mediated interactions are characterized by:
  - increasing volume of offers (chaos) creates opportunities for brokers/mediators
  - point-and-click paradigm not appropriate for intermediaries (need automated processing)
  - emphasis on information integration (e.g. ontologies)
  - traditional service providers are reentering the market via reintermediation (e.g. travel agencies --> direct sales by airlines --> on-line travel agencies)
  - portals and marketplaces as first point of contact, business transaction and fulfillment realized by seller

.....<sup>24</sup>

:

### Questions to Ask about Business Models

- What is the purpose of the business (mission)?
- What are the business goals?
- What is the revenue model?
- What are measurable objectives?
- How is the product delivered (order fulfillment)?
- Is the system open or restricted to some group?
- What is the potential size of the market?
- What is the expected growth rate?
- What are critical success factors?
- What are the quantitative descriptors?

.....25

:

### Application Patterns

- Application patterns are abstract descriptions of application scenarios
- Based on abstract classes of requirements
- A set of generic application patterns were developed for benchmarking object monitors (Boucher/Katz)
- Since object monitors are often used as platforms for e-commerce applications, these patterns can be used to characterize e-commerce applications

.....26

:







### Properties of Application Patterns

- number of concurrent users
- length of user session
- user population change rate (sign-in/sign-off variance)
- number of data sources
- inquiries to update ratio
- average length of chain of a distributed call
- connectivity/invocation requirements (synchronous vs. asynchronous)
- requirement for maintaining state of changes
- requirement for high availability/fault tolerance
- use of application services

.....27

:

### Sample Generic Patterns of Popular Applications

- Patterns are named based on properties and benefits they are to bring to the customers
  - Customer Service Direct 
  - Customer Service Indirect 
  - Customer On-Call 
  - Customer with a Vote 
  - Indecisive Customer 
  - Customer with Cash 
- Note: patterns are incomplete w.r.t. list of requirements, should be adapted to specific case

.....28

:

### Customer Service Direct

- User drives the application
- Main objective: rapid response to multiple customer requirements
  - product purchase
  - access to account information
  - home banking (used here for requirement spec.)
- Large number of customers
- Unpredictable peak loads
- Need for scalability and performance
- Multiple short transactions with updates
- Caution: don't underestimate long interactions

.....<sup>29</sup>

:

### Requirements Customer Service Direct (1)

Feature	Business Reqs.	Technology Implication
# concurrent users	large > 10 <sup>3</sup>	Multiple users accessing same object Multithreading or process pools req.
kind of interactivity	interactive	Heavy traffic between client and server Must maintain session information to avoid frequent save/restore
avg. session length	short (banking)	No need to provide long running sessions

.....<sup>30</sup>

:

### Requirements Customer Service Direct (2)

Feature	Business Reqs.	Technology Implication
User population change rate	high	Effective load balancing needed to respond to varying # of users
# request/responses	small to medium	No special requirements needed
sessions and states	yes	System must be able to maintain sessions and state between client interactions
# of hits	large	Proportional to # of concurrent sessions

.....<sup>31</sup>

:

### Requirements Customer Service Direct (3)

Feature	Business Reqs.	Technology Implication
Response time	immediate	System must provide upward bounded response time with little degradation
inquiries to update ratio	5:1	System should support management of database connection pool
distributed workflow	simple	Connection between connected components could be resolved statically before making call

.....<sup>32</sup>





:

### Customer Service Indirect

- Typical for customer service apps., e.g. complex bookings, complaints, billing clarifications, etc.
- Corporate personnel access system and interact with user
- Corporate personnel must access multiple information sources
- Heterogeneity and bandwidth larger than in direct customer interactions
- Number of users is limited by call-center personnel

.....<sup>33</sup>

:

### Requirements Customer Service Indirect (1)

Feature	Business Reqs.	Technology Implication
# concurrent users	$10^2 < 10^3$	Must handle reasonable and very predictable # of service reps.
kind of interactivity	very interactive	Conversational interactivity w. user may result in access to many backends
avg. session length	very long	Caching of intermediate values to minimize repeated access to backend systems within a session

.....<sup>34</sup>

:

### Requirements Customer Service Indirect (2)

Feature	Business Reqs.	Technology Implication
User population change rate	very stable during work hrs.	No special needs since user number is very stable
# request/responses	medium to large	System must be able to handle high number of interactions
sessions and states	yes	System must be able to maintain sessions and state between client interactions
# of hits	large	System should be able to handle traffic to multiple sources transferring large data volumes

.....<sup>35</sup>

:

### Requirements Customer Service Indirect (3)

Feature	Buss. Reqs.	Technology Implication
Response time	variable	System must provide upward bounded response time with little degradation
inquiries to update ratio	2:1	System should support management of database connection pool. Updates may be across heterogeneous systs.
distributed workflow	complex	System should support workflows across various combinations of business components

.....<sup>36</sup>



:

## Customer On-Call

- Simple call center applications
- Users call in and perform small variety of common services (e.g. check account information, transfer funds from savings to checking)
- Intention is to reduce intervention of service personnel
- Reduce load on system ==> the quicker the better
- Must manage high loads and queues

.....<sup>37</sup>

:

## Requirements Customer on Call (1)

Feature	Buss. Reqs.	Technology Implication
# concurrent users	may be very large	Multiple users, queueing, and high availability
kind of interactivity	minimal & predefined	Less than in full-fledged customer service applications
avg. session length	short	Major requirement to make each session as short as possible

.....<sup>38</sup>

:

## Requirements Customer on Call (2)

Feature	Buss. Reqs.	Technology Implication
User population change rate	high	Effective load balancing needed to respond to varying # of users
# request/responses	small	few requests per interaction, but proportional to number of users
sessions and states	yes	System must be able to maintain sessions and state as cost for reestablishing a session may be high
# of hits	large	Proportional to # of concurrent sessions, therefore high # of hits

.....<sup>39</sup>

:

## Requirements Customer on Call (3)

Feature	Buss. Reqs.	Technology Implication
Response time	minimal	System must minimize response time for all users
inquiries to update ratio	7:1	Mostly for information purposes, minimal update activity, small amounts of information
distributed workflow	medium	System may have to support predefined component combinations

.....<sup>40</sup>



:

### Customer with a Vote

- Typical applications are online auctions or brokerages (used as example for requirements)
- Primary business objective is to handle unpredictable customer requests in timely manner with integrity guarantees
- High availability and fault tolerance are essential
- This kind of application may be well suited for event-based systems (see Cilia et al. 2000)

.....41

:

### Requirements Customer with a Vote (1)

Feature	Buss. Reqs.	Technology Implication
# concurrent users	extremely large and variable	Multiple users accessing same object Must handle quickly varying system loads, multithreading
kind of interactivity	medium	Heavy traffic between client and server Type and level of interaction fairly predictable
avg. session length	short	System must provide multiple discrete parallel sessions

.....42

:

### Requirements Customer with a Vote (2)

Feature	Buss. Reqs.	Technology Implication
User population change rate	extremely high	Very volatile application with a lot of unpredictable picks
# request/responses	small to medium	No special requirements needed for most information going back and forth in both directions
sessions and states	absolutely yes	Critical requirement with integrity and transactional properties and delivery guarantees
# of hits	variable	System must handle unpredictable # of hits

.....43

:

### Requirements Customer with a Vote (3)

Feature	Buss. Reqs.	Technology Implication
Response time	reasonable	Timely confirmation of executed operations is critical
inquiries to update ratio	5:1	Reasonable update/retrieval rate. Cacheing may be difficult
distributed workflow	short	Not a major requirement for this application

.....44



:

### Indecisive Customer

- Pattern typical for decision support system apps.
- Example used here is loan selection service
- Multiple complicated ad hoc requests to many data sources
- Intensive calculations require multithreaded environment, memory and CPU mgmt. At backend
- Asynchronous communication to start different scenarios without waiting for previous calculation to finish
- Sufficient bandwidth needed for graphical data

.....<sup>.45</sup>

:

### Requirements Indecisive Customer (1)

Feature	Buss. Reqs.	Technology Implication
# concurrent users	medium	Typical intranet application, corporations can predict # of users
kind of interactivity	high	Very long interactive sessions must be supported by system
avg. session length	medium to long	User will interact with application systems doing long computations

.....<sup>.46</sup>

:

### Requirements Indecisive Customer (2)

Feature	Buss. Reqs.	Technology Implication
User population change rate	medium	After sign-in user spends predictably large amount of time in one session
# request/responses	large to very large	System must be able to handle very large answers sent back to user, also graphics and charts
sessions and states	yes	Conversational system must be able to maintain sessions and state and recover without losing session data
# of hits	high	User should be allowed to run concurrent sessions

.....<sup>.47</sup>

:

### Requirements Indecisive Customer (3)

Feature	Buss. Reqs.	Technology Implication
Response time	reasonable	Response time requirements ammeliorated by concurrent queries
inquiries to update ratio	10:1	Mostly query oriented system but probably requires access to many sources
distributed workflow	short	Query oriented system doesn't require workflows

.....<sup>.48</sup>



:

### Customer with Cash

- Typical application are on-line reservation and purchasing systems
- Sample pattern based on simple on-line purchasing application
- Important to integrate system with existing components
- Must ensure data integrity at all times, even across heterogeneous systems
- Must handle state between sessions and recover without losing state information

.....<sup>49</sup>

:

### Requirements Customer with Cash (1)

Feature	Buss. Reqs.	Technology Implication
# concurrent users	medium	Loads are often known because existing systems are handling it
kind of interactivity	high	Similar to conversation type systems Must maintain session information to avoid frequent save/restore
avg. session length	long	Sessions may span multiple log-ins

.....<sup>50</sup>

:

### Requirements Customer with Cash (2)

Feature	Buss. Reqs.	Technology Implication
User population change rate	medium	Fairly predictable
# request/responses	medium	Average, manageable size
sessions and states	yes	System must maintain state even across sessions
# of hits	medium	Reasonable and predictable

.....<sup>51</sup>

:

### Requirements Customer with Cash (3)

Feature	Buss. Reqs.	Technology Implication
Response time	immediate for small # of requests	Immediate response required for portions of every request
inquiries to update ratio	2:1	System may require updates to multiple sources
distributed workflow	very long	May span sessions that include many cooperating components

.....<sup>52</sup>



:

## Evaluation of Patterns

- Patterns can be used as a generic description of the business model
- Patterns are useful characterizations of application scenarios but are not detailed enough for quantitative analysis
- Must move to more detailed models  
 ==> functional models, customer models, resource models

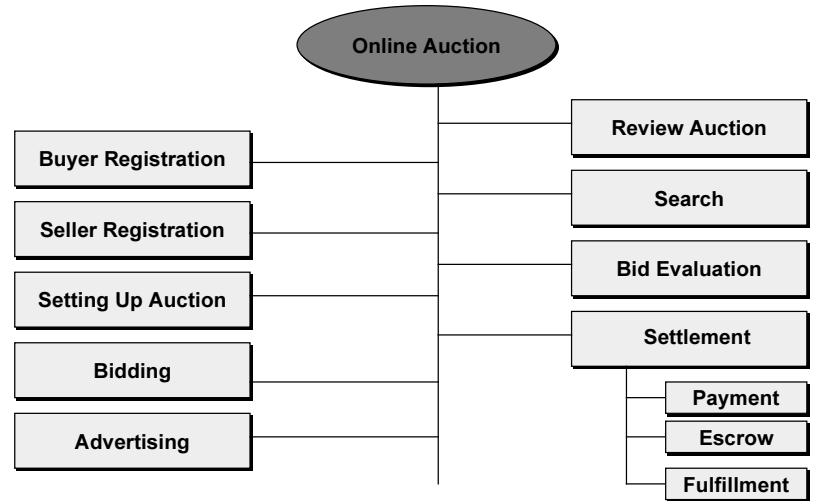
:

## Functional Model

- The functional model describes the processes that deliver services to the customers (select item, review shopping cart, checkout, pay...)
- Processes can be broken down into smaller services or activities (possibly provided by 3rd parties)
- Functional model provides framework to
  - identify navigational structure of site
  - analyze the possible paths taken by customers
- Functional model is the result of functional analysis

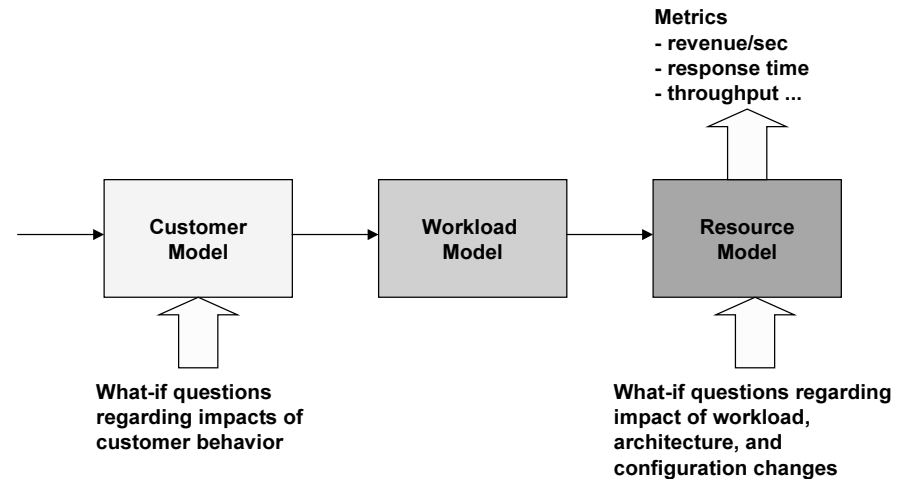
:

## Functional Model of an Online Auction



:

## Customer, Workload and Resource Models



:

## Customer Models

- Customer Model captures user behavior in terms of
  - navigational patterns
  - e-commerce functions used
  - frequency of access to the various e-commerce functions
  - times between accesses to the various services offered
- Customer models are the basis for navigational and workload predictions

.....<sup>57</sup>

:

## Workload Model

- Workload Model describes workload of an e-business site in terms of
  - workload intensity (e.g. transaction arrival rates)
  - service demands (e.g. CPU, I/O subsystem, network)
- Workload Model can be derived from the Customer Models

.....<sup>58</sup>

:

## Resource Model

- Resource Model represents the resources of a site and the effects of the workload on them
- Resource Model can be used for predictive purposes regarding performance impact due to
  - changes in configuration
  - software architecture
  - hardware architecture
- Used for computing metrics, such as
  - response time
  - throughput

.....<sup>59</sup>

:

## Customer Behavior Model: Online Bookstore

- Functions provided by the Online Bookstore:
  - home page with links to bestsellers and promotions
  - search for titles by keyword, ISBN, author
  - select book and view info (description, price, ranking, reviews, shipping time)
  - register as new customer (name/password, mailing data, credit card, e-mail for notifications)
  - login with name and password
  - add items to shopping cart
  - pay for items in shopping cart

.....<sup>60</sup>

:

## Customer Behavior Model

- Customer Behavior Models are used for modeling navigational patterns for predictive purposes
- Used for answering what-if questions when site layout changes
- Predictions of future moves to prefetch data

. . . . .61

:

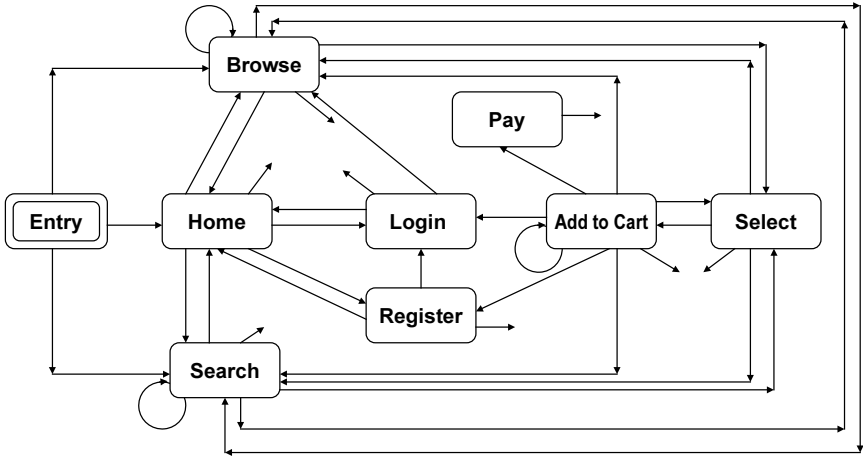
## Online Bookstore (cont.)

- During a session the user may execute several functions
- Executing a function results in the user being in a certain state
- Transitions from one state to another depend on the site layout
- The possible transitions between states are modelled by a graph: the Customer Behavior Model Graph (CBMG)
  - states are represented by squares, transitions by arrows

. . . . .62

:

## States and Transitions of the Online Bookstore



. . . . .63

:

## State Transitions

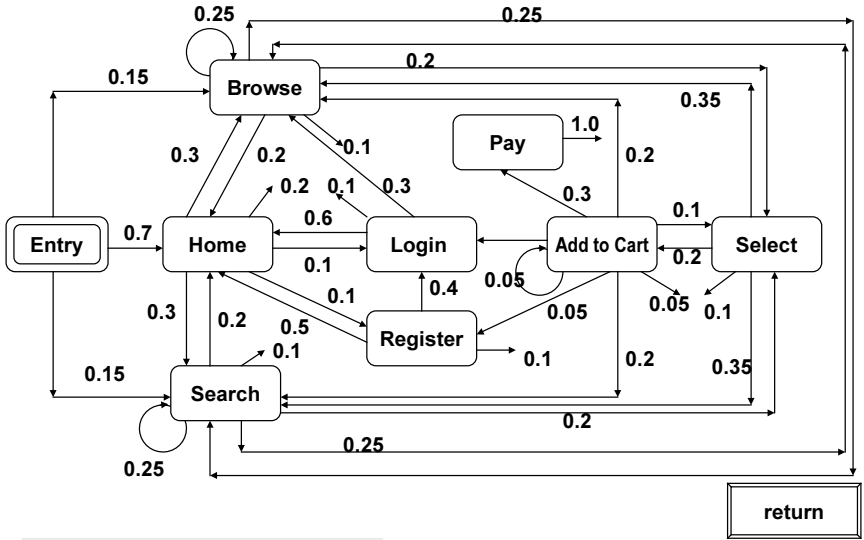
- The states visited depend on the user and may vary from session to session
- Frequency of visits to a state may characterize a user or a type of user
  - casual users browse more and buy less
  - frequent shoppers visit the Add to Cart and Pay states more often
- For each session the ratio of visits to a state to the total number of visits can be computed
- Averaging over all sessions yields state transition probabilities

. . . . .64



:

### CBMG of Online Bookstore



.65

:

### Customer Behavior Model Graphs

- A CBMG consists of  $n$  states including an Entry and an Exit state, and a set of possible transitions
- The sum of state transition probabilities (outbound edges) for each state must be 1.0
- All state transition probabilities  $p_{ij}$  form the  $n \times n$  state transition probability matrix  $P$
- The CBMG is a characterization of the navigational patterns of users
- CBMG is a characterization from the server view (requests serviced by a cache are not seen)

.66

:

### Matrix $P$ for CBMG of Online Bookstore

	y	h	b	s	l	p	r	a	t	x
Entry (y)	0.00	0.70	0.15	0.15	0.00	0.00	0.00	0.00	0.00	0.00
Home (h)	0.00	0.00	0.30	0.30	0.10	0.00	0.10	0.00	0.00	0.20
Browse (b)	0.00	0.20	0.25	0.25	0.00	0.00	0.00	0.00	0.20	0.10
Search (s)	0.00	0.20	0.25	0.25	0.00	0.00	0.00	0.00	0.20	0.10
Login (l)	0.00	0.60	0.30	0.00	0.00	0.00	0.00	0.00	0.00	0.10
Pay (p)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
Register (r)	0.00	0.50	0.00	0.00	0.40	0.00	0.00	0.00	0.00	0.10
Add to Cart (a)	0.00	0.00	0.20	0.20	0.05	0.30	0.05	0.05	0.10	0.05
Select (s)	0.00	0.00	0.35	0.35	0.00	0.00	0.00	0.20	0.00	0.10
Exit (x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

- Probabilities can be obtained by
  - Data mining and OLAP techniques on HTTP logs (evaluation and development of running system)
  - Assumptions made during design phase (initial sizing)

.67

:

### Building a CBMG

- Determine the set of functions provided by site
  - combination of standard functions and site-specific functions
  - refine the set of functions according to resource consumption or function-specific workflow
    - separate audio download from video download
    - separate trading of stocks from bonds and mutual funds
  - determine the transitions between states
    - analyze existing site layout or
    - analyze design of site
- Assign transition probabilities (as discussed earlier)

.68

## Examples of Common E-Business Functions

Category	Function	Description
Common	Login	login to site
	Register	register a new user
	Search	search site database
	Select	view one of the results of the search
	Browse	follow links within the site
Retail	Add Item	add item to shopping cart
	Remove Item	remove item from shopping cart
	See Cart	check contents and value of shopping cart
	Create Registry	create a gift registry
	Add to Registry	add item to gift registry
	Check Status	check status of previous order
Information	Pay	pay for items in shopping cart
	Download	download software/report/music/video
	Subscribe	subscribe to regular downloads or mailings
	Listen Watch	listen to real-time audio watch real-time video



.69

## Aggregate Metrics for Web and E-Business Sites

- Determine a site's popularity or revenue generated
  - hits/sec measures every link retrieved (incl. embedded objects), therefore imprecise
  - page views/day counts individual pages served per day (counts times a banner ad is seen)
  - click-throughs counts how often a user clicks on an ad to get to page behind ad (specificity?)
  - unique visitors counts distinct visitors per unit time
  - revenue throughput measures \$/sec derived from sales, good indicator of customer satisfaction
  - potential loss throughput measures worth of cart not converted into sale (e.g. because of poor performance)

.70

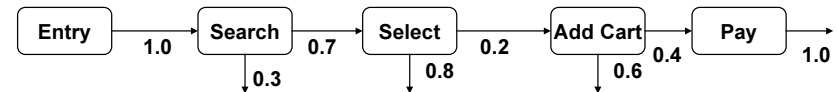
## Metrics Derived from the CBMG

- Interesting questions to be answered based on the CBMG are
  - how many times on average is each e-business function invoked per visit to the e-commerce site? Ex. 1
  - on average, how often do customers buy something each time they visit the e-commerce site? Ex. 2
  - what is the average number of e-commerce function executions requested by a customer during a visit to the store? Ex. 3

.71

## Ex. 1: Average number of visits to a state

Consider the simple CBMG below:



Of every 100 visits to the site

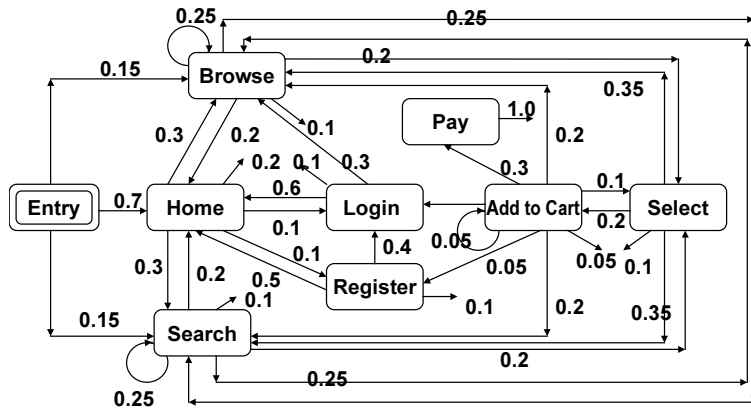
100	move to Search ( $V \times 1.0$ )
70	move to Select ( $V_{\text{search}} \times 0.7$ )
14	move to Add Cart ( $V \times 1.0 \times 0.7 \times 0.2$ )
5.6	move to Pay ( $V \times 1.0 \times 0.7 \times 0.2 \times 0.4$ )

In this simple case there was only one path from one state to the next  
 In a more complex CBMG there may be transitions from multiple nodes  
 The average number of visits to state  $j$  is equal to the sum of the visits to all other states  $k$  multiplied by the transition probability  $p_{k,j}$

$$V_j = \sum_{k=1}^{n-1} V_k \times p_{k,j} \quad j = 2, \dots, n-1 \quad V_1 = 1 \quad V_n = 1$$

.72

### Ex. 1 : Average number of visits to a state (cont.)



State	Visit Ratio
Entry	1.000
Home	1.862
Browse	2.303
Search	2.193
Login	0.274
Pay	0.058
Register	0.196
Add Cart	0.193
Select	0.919
Exit	1.000

Table is obtained by solving system of linear equations on previous slide



.73

### Ex. 2: Visit to sale ratio

State	Visit Ratio
Entry	1.000
Home	1.862
Browse	2.303
Search	2.193
Login	0.274
Pay	0.058
Register	0.196
Add Cart	0.193
Select	0.919
Exit	1.000

Given a ratio of 0.058 for the Pay state only 5.8% of visits result in a sale

19.6% of visitors add items to the shopping cart

Given the high ratio of Add Cart / Pay

$$0.196 : 0.058 = 3.38$$

management may conclude that there is one of several problems:

- slow site
- cumbersome checkout or payment method
- low level of trust in site's security
- ...



.74

### Ex. 3: Average number of requests per session

State	Visit Ratio
Entry	1.000
Home	1.862
Browse	2.303
Search	2.193
Login	0.274
Pay	0.058
Register	0.196
Add Cart	0.193
Select	0.919
Exit	1.000

What is the average session length?

Based on the CBMG and the visit ratios to its states this is answered by adding the visit ratios for all internal nodes (i.e. exclude the fictitious Entry and Exit states)

$$\begin{aligned}
 \text{AverageSessionLength} = & 1.862 \\
 & + 2.303 \\
 & + 2.193 \\
 & + 0.274 \\
 & + 0.058 \\
 & + 0.196 \\
 & + 0.193 \\
 & + 0.919 \\
 & \text{-----} \\
 & 7.998
 \end{aligned}$$



.75

### Customer Visit Model

- CVM models a session as a vector of visits to each state of the Customer Behavior Model Graph but has no information about the transition that was traversed to reach a state
- Customer Visit Models are less detailed than Customer Behavior Models
- CVM has no predictive capabilities to answer what-if questions but is rather useful as input for workload models

.76

## Ex. 4: Customer Visit Model and its use

Function	Session1	Session2	Session3	
Home	1	2	3	Session 1 browses, searches and selects 3 items but doesn't login nor buy anything
Browse	4	8	4	
Search	5	5	3	
Login	0	1	1	Session 2 is a session of a previously registered user who logs in, places items in the cart but abandons the purchase
Pay	0	0	1	
Register	0	0	1	
Add to Cart	0	2	1	
Select	3	3	2	Session 3 is a new customer who registers, logs in, selects an item and buys it

Use of CVM as source of workload information:

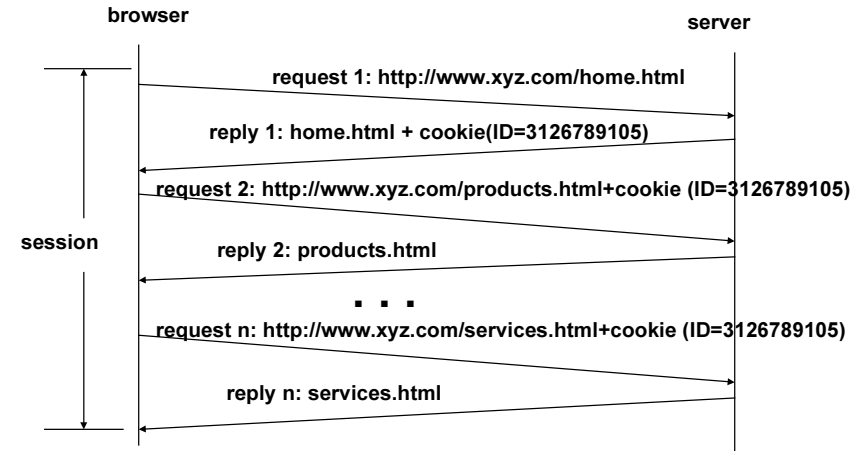
Assume Search requires 3 I/Os (index) and Select requires 2 I/Os (product catalog). What is the average number of I/Os generated by these three sessions?

5+5+3=13 Searches ==> 39 I/Os ==> 39/3 = 13 I/Os average  
3+3+2=8 Selects ==> 16 I/Os ==> 16/3 = 5.33 I/Os average

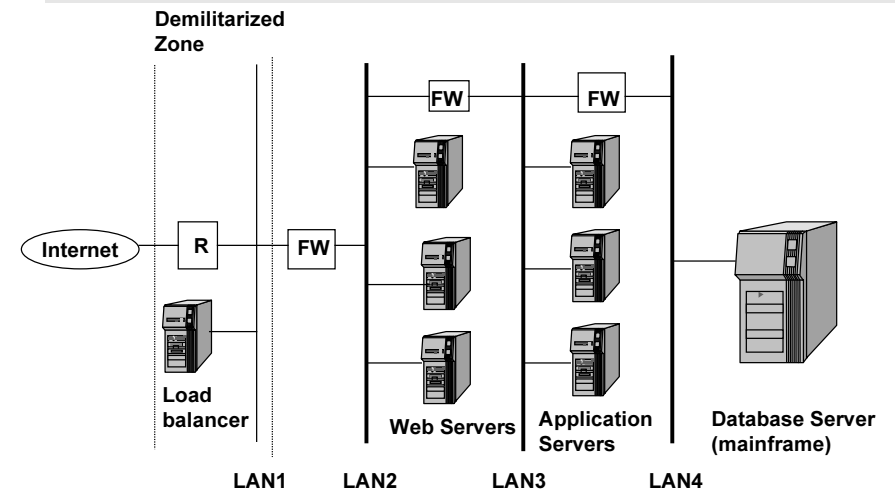
## Session Identification

- User sessions are the basis of user models
- However, HTTP is a stateless protocol
- How can requests be identified as
  - coming from same customer
  - coming from different sessions of same customer
- Cookies are a popular way of storing customer, session and/or state of an application
- Cookies are pieces of information sent by the server and stored at the client

## Ex. 5: Session Identification with Cookies

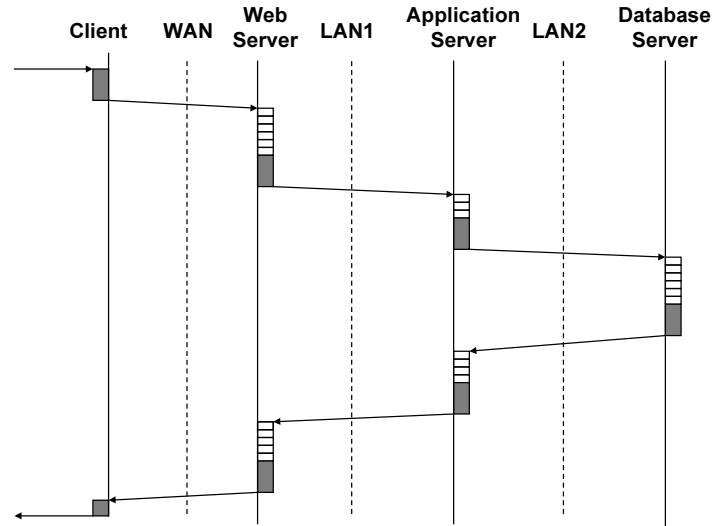


## Reference Architecture: Multi-Tier Site



:

### Typical Interaction in Today's Client/Server Systems



.81

:

### First Generation e-Commerce Systems

- Time to market is dominant factor
- Simplest interaction mechanisms (request-reply)
- Simplest invocation mechanisms (synchronous RPC or equivalent semantics, e.g. RMI or SOAP, based on point-to-point messaging)
- Systems do not scale well for some scenarios (e.g. information dissemination applications)

.82

:

### Next Generation e-Business Systems

- Flexibly composed services ==> service provider may not be known in advance
- Flexible invocation mechanisms ==> events
- Information and events will flow ==> filters and publish/subscribe mechanisms
- Service providers not known a priori ==> subject- or content-based addressing
- Consumers of events and consumption modes not known a priori ==> client-side QoS control, safety
- New platforms begin at square 1 ==> evolution
- No single platform ==> platform interoperability, ADL

.83

:

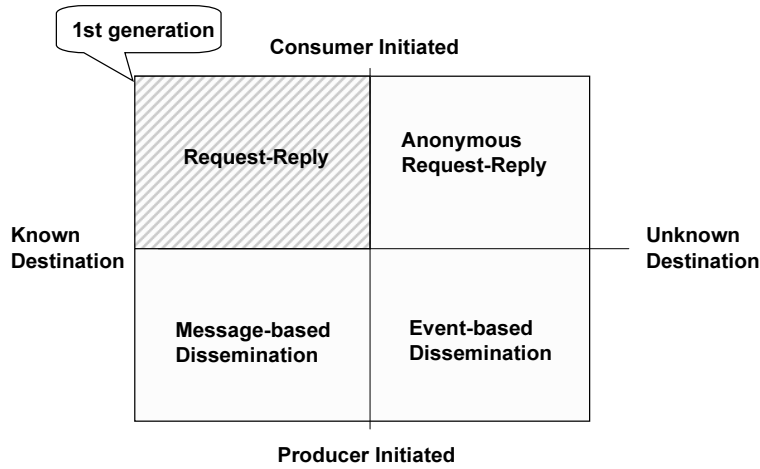
### Interaction vs. Invocation/Communication

- Must separate mode of interaction and mode of invocation (i.e. communication)
- Separation must occur at various levels of abstraction

.84

⋮

## Modes of Interaction



⋮ . . . . .85

⋮

## Invocation/Communication Mechanisms

- Synchronous vs. Asynchronous
- 1:1, 1:n, m:n
  - RPC: synchronous, simple call-return semantics, hard-wired termination and ordering
  - Multicast: 1:N messaging for group communication
  - Peer-to-peer messaging: flexible sequencing of messages, asynchronous (synchronous possible), optimizable, sequencing and timing reflected in application logic therefore more difficult to use
  - (Event) Queues: fully asynchronous, maximum flexibility for handling client/server/communication failures

⋮ . . . . .86

⋮

## Remote Procedure Call Model

- Processes in same address space communicate via procedure calls
- Remote procedure calls mimic behavior of local procedure calls for communication among different processes
- Synchronous communication
  - calling process stops, waits for procedure to execute and control to be returned

⋮ . . . . .87

⋮

## Advantages of RPC

- Programming can ignore distribution, remote calls treated same as local calls
- Safe: every call receives exactly one return
  - return from called procedure or
  - exception from exception handler
- No sequencing of messages required as in peer-to-peer (sequence hard-wired)
- Language transparency for parameters
- HW transparency

⋮ . . . . .88

:

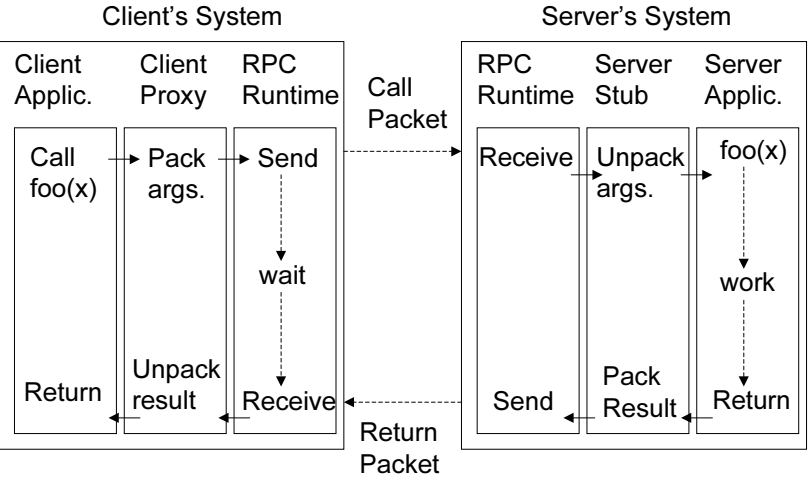
## Disadvantages of RPC

- Blocking because of synchronous nature
- Lack of parallelism
- Always client-initiated
- Performance

. . . . .89

:

## RPC Walk-through



. . . . .90

:

## Performance of RPC

- 3 main components
  - marshaling/unmarshaling of parameters
  - RPC runtime and communication software
  - physical network transfer
- Total cost of one RPC ~ 10 000 -15 000 machine instructions
- Local PC approx. 100 times faster!

. . . . .91

:

## Implications of Architecture

- Today the vast majority of functioning e-business sites are based on synchronous request reply models
  - Known performance modeling techniques concentrate on this paradigm
  - Performance modeling and capacity planning for asynchronous, event-driven systems based on service composition are research issues
- ==> we will concentrate on traditional C/S models in this course

. . . . .92

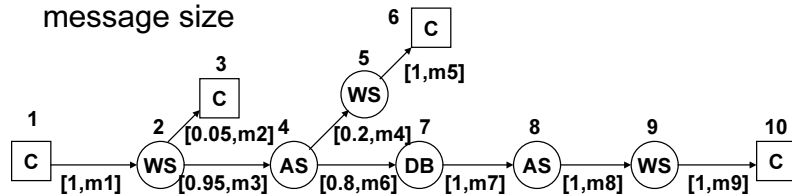
## Client/Server Interaction Diagrams (CSID)

- Need a notation to describe interactions among software components
- Examples of common notations are not performance oriented (e.g. UML sequence diagrams, Specification and Description Language SDL, Message Sequence Charts, etc.)
- CSIDs are diagrams that describe the flow of Client/Server interactions and are annotated with performance information

.93

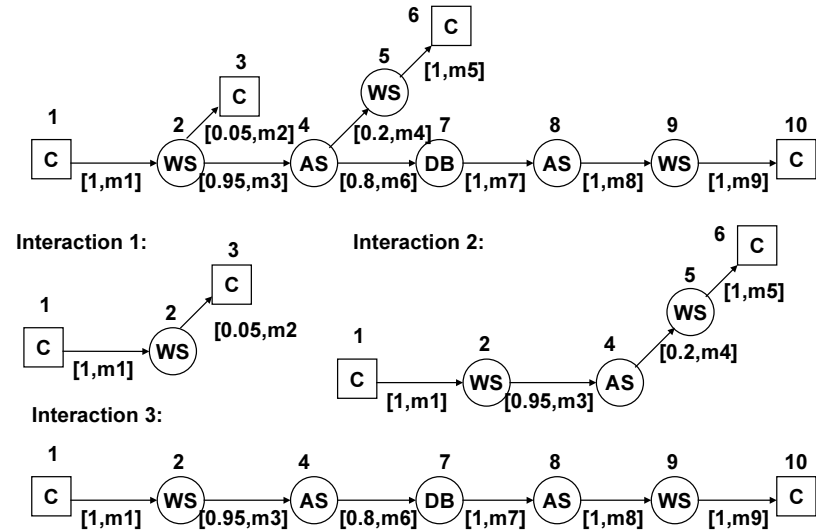
## C/S Interaction Diagrams

- CSIDs are directed graphs
  - nodes represent visits to clients (squares) or servers (circles) during the execution of an e-business function
  - arcs indicate the messages that are sent between nodes
  - nodes are annotated with an identifier and the type of the node (client, WS, AS, DB)
  - arcs are annotated with pairs  $[p,m]$  where  $p$  stands for the probability that a message is sent and  $m$  is the message size



.94

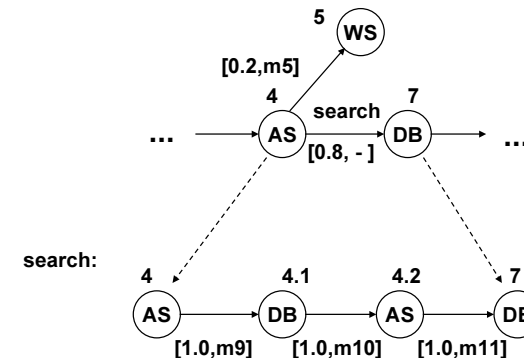
## Composition/Decomposition of CSIDs



.95

## Hierarchical Representation of CSIDs

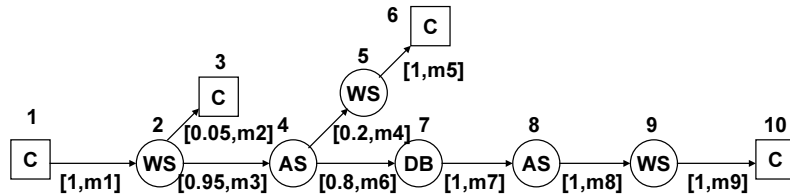
- Complex interactions may be represented as a single arc that can be expanded at a lower level



.96



## Ex 6: Answering Quantitative Questions w. CSIDs



What is the probability that the DB server is used in the above e-function?

DB server is used only in interaction 1 → 2 → 4 → 7 → 8 → 9 → 10  
 Probability that DB server is invoked is  $1.0 \times 0.95 \times 0.8 = 0.76$

What is the average number of times the application server is activated?

App. Server is invoked twice along path 1 → 2 → 4 → 7 → 8 → 9 → 10 and once along path 1 → 2 → 4 → 5 → 6  
 Probability 1 → 2 → 4 → 7 → 8 → 9 → 10 =  $1.0 \times 0.95 \times 0.8 \times 1.0 \times 1.0 \times 1.0 = 0.76$   
 Probability 1 → 2 → 4 → 5 → 6 =  $1.0 \times 0.95 \times 0.2 \times 1.0 = 0.19$   
 Activations of Application Server =  $2 \times 0.76 + 1 \times 0.19 = 1.71 = 0.95 + 0.76$

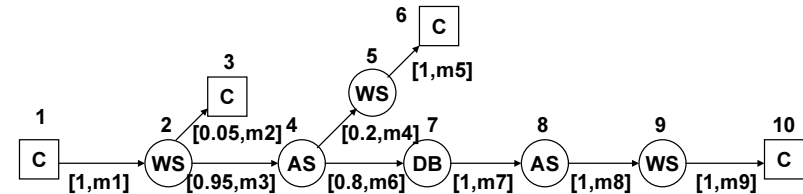
## Design Principle

- Compute the number of activations of a server by either
  - calculating the probability of the whole interaction and multiplying by the times a server is activated in that interaction
  - calculate the probability that the state is reached from the beginning of the interaction
- Add individual number of activations

## Ex. 7: Deriving Network Utilization from the CSID

Suppose Web server, application server and database server run on different machines connected to the same LAN.

What is the average number of bytes that cross the LAN for each execution of the CSID?



For each interaction determine the probability of execution and compute the messages due to each interaction

Interaction 1: 1 → 2 → 3  $.05 \times (m1 + m2)$   
 Interaction 2: 1 → 2 → 4 → 5 → 6  $.19 \times (m1 + m3 + m4 + m5)$   
 Interaction 3: 1 → 2 → 4 → 7 → 8 → 9 → 10  $.76 \times (m1 + m3 + m6 + m7 + m8 + m9)$

## Deriving Network Utilization from the CSID (2)

Assuming the following message lengths:

m1 = 200 Bytes	m2 = 200 Bytes	m3 = 400 Bytes
m4 = 2000 Bytes	m5 = 2100 Bytes	m6 = 500 Bytes
m7 = 3000 Bytes	m8 = 2000 Bytes	m9 = 2100 Bytes

Interaction 1:  $.05 \times (m1 + m2) = .05 \times (400)$   
 Interaction 2:  $.19 \times (m1 + m3 + m4 + m5) = .19 \times (4700)$   
 Interaction 3:  $.76 \times (m1 + m3 + m6 + m7 + m8 + m9) = .76 \times (8200)$

Average number of bytes per execution: 7145 Bytes

Protocol overhead = 10%

Average number of bytes per execution:  $7145 \times 1.1 = 7859.5$  Bytes = 62876 bits

LAN = 100 Mbps Ethernet nominal

80 Mbps real due to contention

How many executions can be realized per unit time?

$80\,000\,000 / 62\,876 = 1\,272$  tps

:

### Design Principle

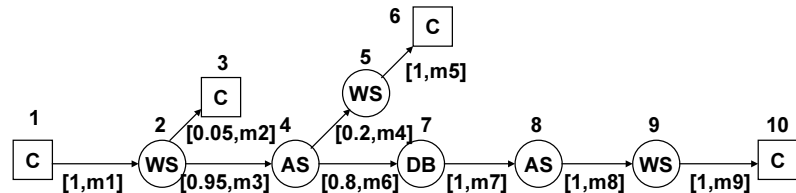
- Number of bytes generated by the execution of a C/S interaction is
  - the sum of the message sizes of all arcs associated with the interaction
  - corrected for protocol overhead
- Number of transactions per unit time
  - network capacity (corrected for contention) divided by corrected sum of message sizes

..... 101

:

### Ex: 8 Deriving Service Demands from CSID

If application server requires 50 msec of CPU time for execution of node 4 and 80 msec for execution of node 8



What is the average CPU time at the application server?

Interaction 1:	p = .95	CPU demand: .95 x 50 = 47.5 msec
Interaction 2:	p = .76	CPU demand: .76 x 80 = 60.8 msec
		CPU total: 108.3 msec = .1083 sec

What is the maximum number of transactions per unit time with one CPU?

1 / 0.1083 = 9.23 tps

..... 102

:

### Design Principle

- Service demands (i.e. the total time spent for a request at a resource) can be computed by
  - identifying all the nodes in the CSID associated with the resource
  - computing the probabilities associated with each node
  - obtaining the service demand for each resource at the node in question
  - adding for all nodes the products of service demands by the probability of occurrence

..... 103

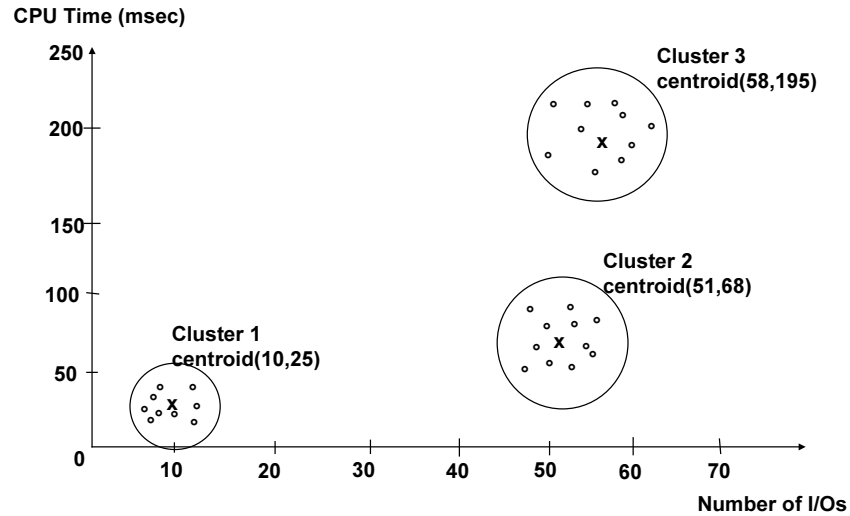
:

### Workload Characterization

- Workload is characterized by
  - workload intensity (e.g. transaction arrival rate)
  - service demand parameters at each resource
- Workloads can be obtained from
  - performance monitors
  - accounting systems
  - system logs
- Measured values are too detailed and voluminous ==> must compact them through a workload model

..... 104

## Clustering for Workload Characterization



105

## Service Time and Service Demand

- Definition: the average service demand  $D_i$  at resource  $i$  is defined as the total service time of a transaction at resource  $i$
- The average service demand can be computed as

$$D_i = V_i \times S_i$$

where  $D_i$  = average service demand

$V_i$  = average number of visits to resource  $i$

$S_i$  = average time spent at resource  $i$

106

## Service Time

- The average service time is the time a transaction spends at a given resource
- The service time does *not* include waiting time, i.e. it is only the exact time a transaction requires to be serviced
- Service time at a server

$$S_i = D_i / V_i$$

107

## Utilization

- The utilization  $U_i$  of a resource  $i$  is defined as the fraction of time that the resource is busy
- If resource  $i$  is monitored during the period  $\tau$  and the resource was busy during  $B_i$  time units

$$U_i = B_i / \tau$$

108

## Ex 9: Calculation of Average Service Time

Consider a site that provides B2B services to the pharmaceutical industry. The Web server interfaces with a legacy mainframe system. The site processes 20 000 orders per day. 90% of the orders have an average of 6 line items, 10% of the orders have 28 line items. Processing of each line item takes 0.5 sec on the mainframe.

What is the average service time at the mainframe?

$$S_{\text{mainframe}} = 0.10 \times (28 \times 0.5) + 0.90 \times (6 \times 0.5) = 4.1 \text{ sec/transaction}$$

What is the average service demand per transaction at the mainframe?

$$D_{\text{mainframe}} = V_{\text{mainframe}} \times S_{\text{mainframe}} = 1 \times 4.1 \text{ sec}$$

What is the daily service demand at the mainframe?

$$D_{\text{mainframe}} = 20\,000 \times 4.1 = 82\,000 \text{ CPUsec}$$

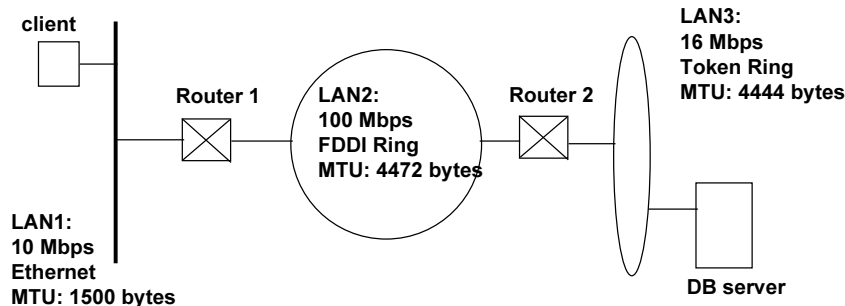
What is the utilization?

$$U_{\text{mainframe}} = 82\,000 / (3600 \times 24) = 0.949$$

109

## Service Time in Communication Networks

A message from client to server or viceversa must go through several protocol layers (transport - TCP, UDP; internet - IP; network - Ethernet, Token Ring) must be transmitted by one or more networks



110

## PDU, MTUs and Fragmentation

- Protocol entities at each layer exchange Protocol Data Units (PDU)
- Depending on the protocol the PDU is named either segment (TCP), datagram (UDP, IP), frame (Ethernet, IEEE 802.5 Token Ring, FDDI)
- The Maximum Transmission Unit (MTU) is the maximum size of data handled by that protocol at the network level
  - Ethernet: 1500 bytes
  - FDDI Ring: 4472 bytes
  - Token Ring: 4444 bytes

111

## Fragmentation

- If a message that fits into the MTU of one network (e.g. 2500 byte packet in FDDI frame w. 4472 bytes) crosses into a network with smaller MTU (e.g. Ethernet w. 1500 byte frame) it may have to be fragmented
- Fragments are reassembled at the IP level by the destination host

112

## Overhead: headers, trailers, MSS

- Each protocol layer adds a header (and some add also a trailer)
- The Maximum Segment Size (MSS) is the largest chunk of data a side expects to receive in a TCP exchange
- $MSS + TCP \text{ header} + IP \text{ header} \leq MTU$  at network layer
- Ex: IP v.4 over Ethernet  
 $MSS = 1500 - 20 - 20 = 1460$  bytes

113

## Characteristics of Network Protocols

Protocol	PDU Name	Max. PDU Size (bytes)	Overhead (bytes)	Max. Data Area (bytes)
TCP	segment	65 515	20	65 495
UDP	datagram	65 515	8	65 507
IP version 4	datagram	65 535	20	65 515
IP version 6	datagram	65 535	40	65 495
ATM	cell	53	5	48
Ethernet	frame	1 518	18	1 500
IEEE 802.3	frame	1 518	21	1 497
IEEE 802.5 Token Ring	frame	4 472	28	4 444
FDDI (RFC 1390)	frame	4 500	28	4 472

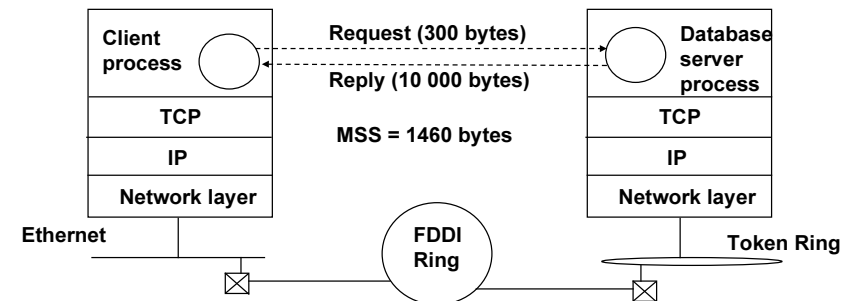
114

## Service Time in Networks

- Service time in network = time it takes to transmit message (incl. protocol header and trailer overhead)  
 $S = \text{message size (with overhead)} / \text{bandwidth}$

115

## Ex: 10: Network Service Time Calculation



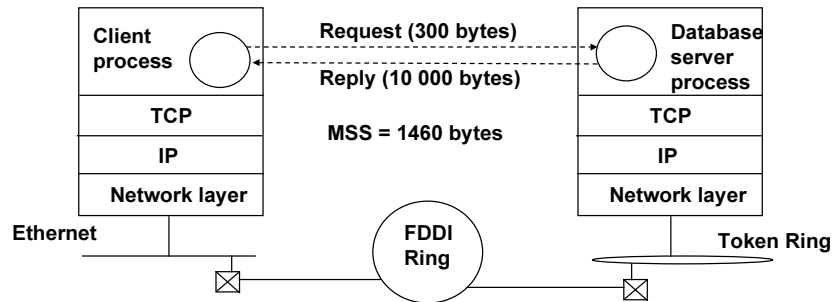
Message sizes:  $((\text{request} + \text{TCP hdr.}) + \text{IP hdr.}) + (\text{fr.LAN1 or fr.LAN2 or fr.LAN3}) =$   
 $300 + 20 + 20 + 18 = 358$  bytes in LAN1  
 $300 + 20 + 20 + 28 = 368$  bytes in LAN2, LAN3

Service time request:

LAN1:  $(358)(8)/(10\ 000\ 000) = 0.000286$  sec  
 LAN2:  $(368)(8)/(100\ 000\ 000) = 0.00002944$  sec  
 LAN3:  $(368)(8)/(16\ 000\ 000) = 0.000184$  sec

116

## Ex: 10: Network Service Time Calculation (cont.)



Message size response: 10 000 bytes divided into  
 6 frames with 1460 bytes TCP data + 20 + 20 + 28 = 1528 bytes  
 1 frame with 1240 bytes rest TCP data + 20 + 20 + 28 = 1308 bytes

Service times:

LAN 3:  $((6)(1528) + 1308)(8) / (16\ 000\ 000) = 0.00524\ \text{sec}$   
 LAN 2:  $((6)(1528) + 1308)(8) / (100\ 000\ 000) = 0.000838\ \text{sec}$   
 LAN 1:  $((6)(1460+20+20+18) + (1240+20+20+18)(8)) / (10\ 000\ 000) = 0.00832\ \text{sec}$

## Summary of Service Time Calculations

- Current IP standards recommend that source host discovers the minimum MTU along a path before choosing initial datagram size (no fragmentation)
- Number of datagrams (no fragmentation)  

$$\lceil \text{NDatagrams} = \text{MessageSize}/\text{MSS} \rceil$$
- Overhead in a network  

$$\text{Overhead}_n = (\text{NDatagrams}) (\text{TCPOvhd} + \text{IPOvhd} + \text{FrOvd}_n)$$
- Service time in a network  

$$\text{ServiceTime}_n = ((\text{MessageSize} + \text{Ovhd}_n)(8)) / (10^6)(\text{Bw}_n)$$

## Router Service Time

- Incoming datagrams are queued until router processor is available to inspect them
- Router processor determines, based on datagram's destination address the next best outgoing link (lookup of routing tables)
- Datagram is enqueued in outgoing queue
- Time taken by router is known as router latency (given by router vendor in  $\mu\text{sec}/\text{packet}$ )

$$\text{RouterServiceTime} = \text{NDatagrams} \times \text{RouterLatency}$$

## Utilization of Network

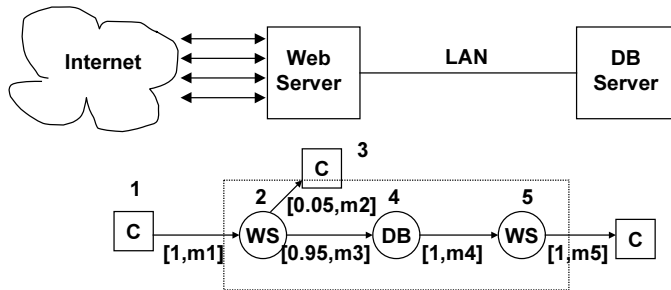
The utilization of the network is defined as

$$U_n = \sum_{\text{messages } j} \lambda_j \times \text{ServiceTime}_n^j$$

where

$\lambda_j$  is the arrival rate of messages of type  $j$   
 $\text{ServiceTime}_n^j$  is the average service time for messages of type  $j$  on network  $n$

## Ex. 11: Utilization of Network



m3 = 400 bytes    m4 = 9150 bytes    throughput = 24.3 searches/sec

Web Server uses CGI stubs to send DB requests over TCP over a 100BASE-T Ethernet (MSS = 1460 bytes)

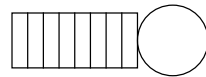
NDatagrams:     $\lceil 400 / 1460 \rceil = 1$              $\lceil 9150 / 1460 \rceil = 7$   
 ServiceTime:    0.366 msec                            7.64 msec  
 $U_{network} = 24.3 \times (0.366 + 7.64) \times 10^{-3} = .195 = 19.5\%$

121

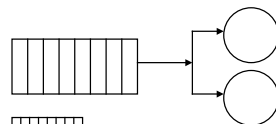
## Queues

- Each time a transaction visits a resource it may have to queue for its use
- Depending on the number of queues and resources we distinguish:

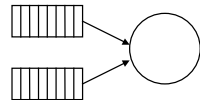
– single server, single queue  
(authentication server)



– multiple server, single queue  
(multiprocessor DB server)



– single server, multiple queue  
(router)



122

## Waiting Time, Response Time, Delay Resources

- Response time encompasses service time and waiting time

$$\text{ResponseTime} = \text{WaitingTime} + \text{ServiceTime}$$

- If availability of resource far exceeds demand, no queue builds up ==> WaitingTime = 0 and ResponseTime=ServiceTime
- These resources are known as *delay resources*
- Delay resources are represented as resources without queue (typically used for 3rd party services)

123

## Notation

- $V_i$  - average number of visits to queue  $i$  by an e-commerce transaction
- $S_i$  - average service time of a transaction to resource  $i$  per visit to the resource
- $W_i$  - average waiting time of a transaction in the queue of resource  $i$  per visit to the queue
- $R_i$  - average response time of transaction at queue  $i$
- $\lambda_i$  - average arrival rate of requests to queue  $i$
- $X_i$  - average throughput of queue  $i$

124

:

### Flow Equilibrium Assumption

- Queues are assumed to be observed over a long time
- If system is in steady state (i.e. it is in equilibrium) the rate of arrival is the same as the rate of completion

$$\lambda_i = X_i$$

- $X_0$  - average system throughput defined as the average number of transactions that complete per unit time

..... 125

:

### Notation (cont.)

- $N_i^w$  - average number of transactions waiting at queue i
- $N_i^s$  - average number of transactions receiving service at any resource of queue i. For single resource queue  $N_i^s$  is the fraction of time that the resource is busy, i.e.  $N_i^s = U_i$
- $N_i$  - average number of transactions at queue i waiting or receiving service from any resource at queue i

$$N_i = N_i^w + N_i^s$$

..... 126

:

### Operational Results and Operational Quantities

- A basic set of measurements is enough to derive some meaningful results (a.k.a. performance laws)
- The basic set of measurements (operational quantities) is
  - observation period  $\tau$
  - time the system (or resource) was busy  $B_0$
  - number of arrivals of a request  $A_0$
  - number of completed requests  $C_0$
- Some (major) simplifications underly the performance laws!!!

..... 127

:

### Ex. 12: Arrival Rates

Under certain conditions, the arrival rate can be derived from measured values of requests arriving during the measuring period  $\tau$

During the measurement interval of 1200 sec the number of requests that arrived at the web site was 24000. What is the arrival rate?

$$\begin{aligned} \lambda &= (\text{number of request arrivals}) / \text{measurement interval} \\ &= A_0 / \tau \\ &= 24\,000 / 1\,200 \\ &= 20 \text{ tps} \end{aligned}$$

Underlying assumption: uniform distribution of arrivals

..... 128



⋮

### Ex. 13: Completion Rate

During measurement interval of 1 200 sec the number of requests executed by the Web Server was 27 300. What is the completion rate  $X_o$ ?

$$\begin{aligned}
 X_o &= (\text{number of completed requests}) / (\text{measurement interval}) \\
 &= C_o / \tau \\
 &= 27\,300 / 1\,200 \\
 &= 22.75 \text{ tps}
 \end{aligned}$$

Simplifying assumptions: single request that always completes

..... 129

⋮

### Ex. 14: Mean Service Time

During the measurement interval of 1 200 sec the CPU of a Web Server was busy for 1 040 sec. What is the processor mean service time per request?

$$\begin{aligned}
 S &= (\text{busy interval}) / (\text{number of completed requests}) \\
 &= 1\,040 / 27\,300 \\
 &= 0.038 \text{ sec/request}
 \end{aligned}$$

Simplifying assumption: single class of request (recall clustering)

..... 130

⋮

### Utilization Law

Utilization:  $U_i = B_i / \tau$

If during monitoring interval  $\tau$ ,  $C_o$  transactions are completed, we obtain the

Average throughput:  $X_i = C_o / \tau$

Combining the definitions:

$$U_i = B_i / \tau = B_i / (C_o / X_i) = (B_i / C_o) X_i = S_i X_i$$

Assumption1: av. service time = busy time / trans.

Assumption2: steady state (equilibrium)  $\lambda_i = X_i$

$$U_i = X_i S_i = \lambda_i S_i$$

..... 131

⋮

### Ex. 15: Utilization Law

During observation interval the database server executed 45 Search transactions per second. Each transaction takes an average of 19.0 msec to execute.

What is the utilization of the DB-server during the interval?

$$U_i = X_i S_i = (45) (0.019) = 0.855 = 85.5\%$$

..... 132

⋮

### Forced Flow Law

From definition of  $V_i$ , each completing transaction must visit queue  $i$   $V_i$  times  
 If  $X_0$  transactions complete per unit time,  $V_i X_0$  visits to the queue  $i$  per unit time will occur  
 The average throughput of queue  $i$  is then

$$X_i = V_i X_0$$

..... 133

⋮

### Ex. 16: Forced Flow Law

During monitoring interval of 20 min, 4800 e-commerce transactions were executed. From the corresponding CSID we know that a typical interaction visits the Web Server 5.2 times and the DB Server 3.8 times. The database service time is 59 msec and the web service time is 35 msec.

What is the average throughput of the DB Server and the Web Server?

$$\text{Site throughput} = (4800) / ((20)(60)) = 4 \text{ tps}$$

Applying forced flow law

$$X_{DB} = V_{DB} X_0 = (3.8) (4) = 15.2 \text{ tps}$$

$$X_{WS} = V_{WS} X_0 = (5.2)(4) = 20.8 \text{ tps}$$

Note the meaning of transaction!

..... 134

⋮

### Service Demand Law

From definition of service demand  $D_i = V_i S_i$  and Utilization Law and Forced Flow Law

$$D_i = V_i S_i = (X_i / X_0) (U_i / X_i) = U_i / X_0$$

Utilization can also be interpreted as the average number of transactions in the resource since there is one transaction in the resource  $U_i$  percent of the time and  $(1 - U_i)$  there is no transaction in the resource

..... 135

⋮

### Ex. 17: Service Demand Law

Same case as Ex. 16. Want to determine the service demands at the DB Server and the Web Server, as well as the utilization of the two.

From definition of service demand

$$D_{WS} = V_{WS} S_{WS} = (5.2)(0.035) = 0.182 \text{ sec}$$

$$D_{DB} = V_{DB} S_{DB} = (3.8)(0.059) = 0.224 \text{ sec}$$

Applying Service Demand Law  $D_i = V_i S_i = (X_i / X_0) (U_i / X_i) = U_i / X_0$

$$U_{WS} = D_{WS} X_0 = (0.182)(4) = 0.728 = 72.8\%$$

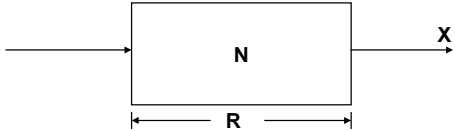
$$U_{DB} = D_{DB} X_0 = (0.224)(4) = 0.896 = 89.6\%$$

..... 136

:

### Little's Law

Model system as a black box

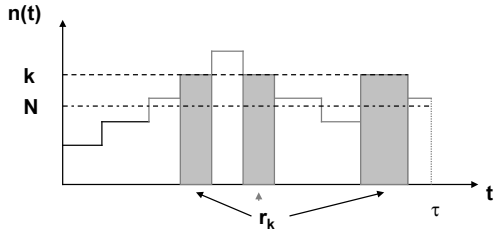


where  $N$  = number of customers in black box  
 $X$  = average departure rate = throughput  
 $R$  = residence time in black box

Little's Law:  $N = X R$

:

### Derivation of Little's Law



- $n(t)$  is the number of customers in black box at  $t$
- $N$  is the average number of customers in observation interval  $\tau$
- $N$  is the sum of  $k f_k$  where  $k$  is the number of customers in black box during fraction  $f$  of the interval

:

### Derivation of Little's Law (cont.)

$$N = \sum_k k f_k = \sum_k k (r_k / \tau)$$

Multiplying and dividing by  $C_0$  (the number of completed transactions or customers leaving the system) and rearranging

$$N = (C_0 / \tau) (\sum_k k r_k) / C_0$$

$(C_0 / \tau)$  is the throughput  $X$ . The summation is the total customer-seconds accumulated in the system. Divided by  $C_0$  we obtain  $R$ , the average time each customer spent in the black box and

$$N = X R$$

:

### Ex. 18: Use of Little's Law

Consider a large portal service that offers free e-mail to its users. The portal has 2 000 000 registered users. During the peak hour, 30% of the users send e-mails through the portal. Each mail takes 5.0 sec on average to be processed. The average message size is 7 120 bytes and each user sends an average of 3.5 mails.

What must be the capacity of the spool for outgoing mails during the peak hour?

Applying Little's Law

$$\begin{aligned} \text{AverageNumberOfMails} &= \text{Throughput} \times \text{ResponseTime} \\ &= (2\,000\,000)(0.3)(3.5) / 3600 \times 5.0 \\ &= 2916,7 \text{ mails} \end{aligned}$$

Based on the average mail size the spool file should be

$$(2916,7)(7120) = 20.76 \text{ Mbytes}$$

:

### Applying Little's Law to Queues

- Little's Law can be applied to a waiting queue

$$N_i^w = X_i W_i$$

- Little's Law applied to the set of m resources

$$N_i^s = X_i S_i$$

Given that this represents the number of transactions at the set of m resources, then the average number of transactions per resource (i.e. the utilization) is  $X_i S_i / m$

- Little's Law applied to an entire queue

$$N_i = X_i R_i$$

. . . . . 141

:

### Ex. 19: Use of Little's Law to obtain response time

A brokerage firm runs a 3-tier site consisting of Web Server, Application Server and Database Server.

The Web trading system is used by 1.1 million customers. During the peak hour 20 000 users are logged in simultaneously. During the peak hour the system processes 3.6 million business functions per hour.

What is the average response time of an e-commerce function during the peak hour?

Consider the 3-tier system as a black box and apply Little's Law  $N = XR$

AverageResponseTime = AverageNumberUsers / SiteThroughput

$$\begin{aligned} &= (20\ 000) / (3\ 600\ 000 / 3\ 600) \\ &= ((20\ 000) (3\ 600)) / (3\ 600\ 000) \\ &= 20\ \text{sec} \end{aligned}$$

. . . . . 142

:

### Ex. 20: Average Response Time at DB Server

Considering the same Web brokerage, we know additionally that each e-commerce function generates 1.4 transactions to the database system on the mainframes.

The aggregate capacity of the mainframes is 11 500 tps.

What percentage of the customer average response time is spent on the mainframes?

Consider the DB Server as the black box and apply Little's Law

AverageResponseTime = AverageNumberOfTransactions / MainframeThroughput

$$\begin{aligned} &= (20\ 000) (1.4) / (11\ 500) \\ &= 2.43\ \text{sec} \end{aligned}$$

Since each e-commerce function generates 1.4 transactions the fraction of time spent on the mainframes is given by

$$\text{FractionOnMainframe} = (2.43) (1.4) / (20.0) = 0.17$$

. . . . . 143

:

### Ex. 21: Performance laws in capacity planning

# completed customer sessions	35 000	1) What is the effect on performance if # of sessions doubles during peak hour?
Web server busy period (sec)	1 200	
DB server busy period (sec)	2 100	
Measurement interval (sec)	3 600	
# searches per customer session	2.5	2) What is the LAN utilization if the size of catalog pages doubles due to new high resolution graphics?
# visits to web server per search	1.95	
# visits to DB server per search	0.95	
Web server service demand (msec)	13.71	
DB server service demand (msec)	24.00	3) What is the effect of a different search pattern with 50% more catalog searches?
Web server service time (msec)	7.03	
DB server service time (msec)	25.26	

1) SystemThroughput = (2) (35 000) (2.5) / (3 600) = 48.61 tps  
 $U_{WS} = X_0 D_{WS} = (48.61) (0.01371) = 0.667$   
 $U_{DB} = X_0 D_{DB} = (48.61) (0.024) = 1.167$  !!! System can't support forecast load !!!

2) See Ex. 11 doubling the message size of the response ==> Unetwork = 0.36

3) NewSearchesPerSession = (2.5) (1.5) = 3.75  $X_0 = (35\ 000) (3.75) / (3\ 600) = 36.46\ \text{tps}$   
 $U_{WS} = X_0 D_{WS} = (36.46) (0.01371) = 0.4997$   
 $U_{DB} = X_0 D_{DB} = (36.46) (0.024) = 0.8748$  System can support new search pattern

. . . . . 144

:

### Bottlenecks and Scalability Analysis

- Performance bounding techniques are used to calculate optimistic and pessimistic bounds
  - throughput upper bounds (optimistic)
  - response time lower bounds (optimistic)
  - worst case execution time (pessimistic)
- Resource with utilization = 1 is saturated
- The single limiting resource is the bottleneck (e.g. DB server in Ex. 21)
- If a bottleneck can't be removed, the system is considered non-scalable in terms of performance
- Bounding techniques are often enough to make quick decisions

..... 145

:

### Asymptotic Bounds

- Asymptotic Bound Analysis determines bounds on throughput and response time under extreme load conditions
- Two main types of models:
  - open models: requests arrive, go through the various resources and leave the system
  - closed models: have a fixed number of requests, e.g. limitation imposed by maximum number of TCP connections

..... 146

:

### Open Models

- Open models are limited by the upper bound on throughput ==> arrival rate  $\lambda$  may not exceed throughput
- In equilibrium
  - $\lambda = X_0$
  - $\lambda_i = X_i$
- From utilization law
  - $U_i = X_i S_i = \lambda_i S_i$
- From Service Demand Law
  - $U_i = D_i X_0 = D_i \lambda$

..... 147

:

### Open Models (cont.)

- For a saturated resource
  - $\lambda D_i = 1$  and  $\lambda = 1/D_i$
- Since for every resource
  - $U_i \leq 1$  it follows that  $\lambda \leq 1/D_i$
  - Therefore, the largest value of  $D$ , i.e.  $D_{max}$  will limit the throughput and the arrival rate and
  - $\lambda_{max} = 1 / D_{max}$
- The resource  $i$  with  $D_i = D_{max}$  is the bottleneck

..... 148

## Ex. 22: Asymptotic bounds with open models

Consider a typical three tier e-commerce site. After clustering the „typical“ e-commerce business function can be characterized by the following table

Layer	Visits	S [msec]
Web Server	1.8	110
Applic. Server	2.5	230
Database Server	2.3	180

Computing the service demands at each layer we obtain:

$$D_{web} = (V_{web})(S_{web}) = (1.8)(0.11) = 0.198 \text{ sec}$$

$$D_{app} = (V_{app})(S_{app}) = (2.5)(0.230) = 0.575 \text{ sec} \quad \implies D_{app} = D_{max}$$

$$D_{DB} = (V_{DB})(S_{DB}) = (2.3)(0.180) = 0.414 \text{ sec}$$

The maximum arrival rate is thus  $\lambda_{max} = 1/0.575 = 1.74$  e-commerce functions/sec

What is the minimum parallelism required to achieve 10 e-functions/sec?

$$\lceil 10/1.74 \rceil = 6 \text{ App. Servers}$$

$$\lceil (10)(0.414) \rceil = 5 \text{ DB Servers}$$

$$\lceil (10)(0.198) \rceil = 2 \text{ Web Servers}$$

149

## Closed Models

- Bounds on closed models can best be obtained by varying the number of transactions in the system
- The ideal situation corresponds to the case where N transactions are in the system and none must wait in a queue
- Combining Little's Law and no queueing

$$X_0(N) \leq N / D_{total}$$

150

## Closed Models (cont.)

- Knowing that the utilization of any resource may not exceed 100% and applying Service Demand Law

$$X_0(N) = U_i(N) / D_i$$

$$U_i(N) \leq 1$$

$$X_0(N) \leq 1 / D_i$$

- Since the bottleneck is the first resource that saturates

$$X_0(N) \leq 1 / D_{max}$$

- The asymptotic upper bound is then

$$X_0(N) \leq \min [N / D_{total}, 1 / D_{max}]$$

151

## Ex. 23: Asymptotic Bounds in Closed Models

Consider a Call-Center situation with a maximum of 20 users connected. The Call-Center personnel connects through a browser to the Web server, an Application Server and a backend Database Server.

The service demands at the servers are:

$$D_{web} = 12.1 \text{ msec}$$

$$D_{app} = 54.5 \text{ msec}$$

$$D_{database} = 65.6 \text{ msec}$$

$$D_{total} = D_{web} + D_{app} + D_{database} = 0.0121 + 0.0545 + 0.0656 = 0.1322 \text{ sec}$$

$$D_{max} = D_{database} = 0.0656 \text{ sec}$$

$$X_0(20) \leq \min [20/0.1322, 1/0.0656] = \min [151.3, 15.2]$$

$$X_0(20) \leq 15.2 \text{ tps}$$

The site is capable of performing a maximum of 15.2 e-business functions/sec

152

:

### Modelling with Queues

- Depending on the level of detail needed and the input data available a queueing model may be established at the system or the component level
- A queue models a resource with its waiting queue
- We distinguish the following types of resources:
  - load-independent resources: queueing,  $S(n) = S \forall n$
  - load-dependent resources: queueing,  $S(n)$  is an arbitrary function of  $n$
  - delay resources: no queueing,  $S(n) = S \forall n$

..... 153

:

### Possible Assumptions when Modelling Queues

- **Infinite population assumption:** the pool of the user population is so large that the arrival rate is independent of the requests that arrived previously ==> requests arrive at an average arrival rate of  $\lambda$  requests/sec
- **Homogeneous workload assumption:** all requests are assumed to be statistically indistinguishable, i.e. only the number of requests present is important, not the individual requests

..... 154

:

### Possible Assumptions when Modelling Queues

- **Infinite queue assumption:** no requests are refused and all arriving requests are queued for service.
- **Finite queue assumption:** the size of the queue is finite and arriving requests may be refused if the queue is full

..... 155

:

### Possible Assumptions when Modelling Queues

- **Operational equilibrium assumption:** the number of requests in the system at the start of the observation interval is the same as the number of requests at the end of the interval. Although the number of requests in the system may vary, for reasonably large observation intervals the error is negligible.
- **Markovian assumption:** systems are memoryless, i.e. it doesn't matter how a system got to a given state, only that it is in that state

..... 156

⋮

### Goals when modelling with queues

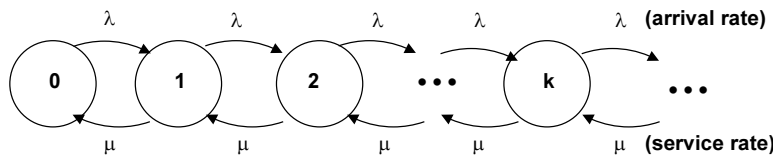
- Want to compute:
  - probability  $p_k$  that there are  $k$  ( $k = 0, 1, \dots$ ) requests in the server
  - average number of requests present
  - average response time of a request
  - a server's utilization
  - a server's throughput

⋮ 157

⋮

### Single Queue: Infinite Population/Infinite Queue

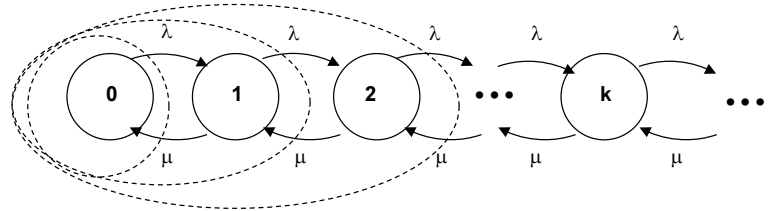
- Assumptions:
  - infinite population
  - infinite queue
  - Markovian assumption (memoryless)
  - operational equilibrium
  - state is characterized by one parameter: # requests at server
- State diagram (STD)



⋮ 158

⋮

### Infinite population / infinite queue

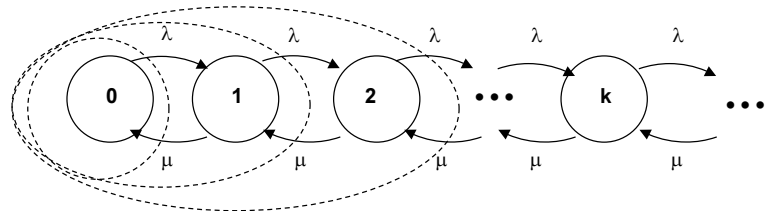


- Because of operational equilibrium: # of transitions into a state = # transitions out of a state
- Flow equilibrium equation or flow-in = flow-out principle

⋮ 159

⋮

### Infinite population / infinite queue



flow-in = flow-out

$$\mu p_1 = \lambda p_0$$

$$\mu p_2 = \lambda p_1$$

⋮

$$\mu p_k = \lambda p_{k-1}$$

⋮ 160



⋮

Infinite population / infinite queue

Combining all the flow equations we obtain

$$p_k = (\lambda/\mu)p_{k-1} = (\lambda/\mu)(\lambda/\mu)p_{k-2} = \dots = p_0(\lambda/\mu)^k, \quad k=1,2,\dots$$

Since the server must always be in one of the possible states (from 0 to  $\infty$ ), the sum of the probabilities that it is in any possible state is 1

$$p_0 + p_1 + p_2 + \dots + p_k + \dots = \sum_{k=0}^{\infty} p_k = \sum_{k=0}^{\infty} p_0 (\lambda/\mu)^k = 1$$

and  $\sum_{k=0}^{\infty} (\lambda/\mu)^k = 1/p_0$  and  $p_0 = 1 - (\lambda/\mu)$

⋮

Infinite population / infinite queue

$$\sum_{k=0}^{\infty} (\lambda/\mu)^k = 1/p_0$$

The infinite sum is the sum of a geometric series. It only converges if its ratio  $\lambda/\mu < 1$

Therefore, an equilibrium solution to the system can only be found if the arrival rate of requests is smaller than the service rate

⋮ 161

⋮

Ex. 24: Single Q, infinite population/infinite Q

Requests arrive to the DB server at a rate of 30 requests/sec. Each request takes 20 msec on average to be processed. What is the fraction of time that k (k = 0, 1, ...) requests are found in the DB server?

The average service rate  $\mu$  is the inverse of the average service time per request.

$$\mu = 1 / 0.02 = 50 \text{ requests / sec} \quad \lambda = 30 \text{ requests / sec}$$

The fraction of time that the DB server is idle,  $p_0$ , is

$$p_0 = 1 - \lambda/\mu = 1 - 30 / 50 = 1 - 0.6 = 0.4 \quad \Rightarrow \text{the server is utilized } 1 - p_0 = 0.6$$

The fraction of time that there are k requests at the server is given by

$$p_k = (1 - \lambda/\mu)(\lambda/\mu)^k = (0.4)(0.6)^k \quad k = 0, 1, 2, \dots$$

k	0	1	2	3	4	5	6	7
$P_k$	0.4	0.24	0.144	0.086	0.052	0.031	0.019	0.011

⋮

Infinite population / infinite queue

- The utilization U of the server is

$$U = 1 - p_0 = \lambda / \mu$$

$$p_k = (1-U)U^k \quad \text{for } k = 0, 1, 2, \dots$$

- The state distribution depends only on the utilization and not on the individual values of arrival and service rates
- The average number of requests at the server is

$$N_{av} = \sum_{k=0}^{\infty} kp_k = \sum_{k=0}^{\infty} k(1-U)U^k = (1-U)\sum_{k=0}^{\infty} kU^k$$

$$\sum_{k=0}^{\infty} kU^k = U/(1-U)^2 \text{ for } U < 1 \quad \text{and} \quad N_{av} = U/(1-U)$$

⋮ 164



⋮

### Summary of Modelling Procedure

- Determine proper representation of state of system
- Determine the set of feasible states
- Determine the possible transition among states and the events that cause them (arrivals, completion,...)
- For each possible transition between states determine the transition rate (e.g. rate at which requests arrive when system is in state k)
- Use the flow equilibrium principle to find the set of equations that relate the values of  $p_k$
- Solve for the  $p_k$ s and use them to compute performance metrics ( $U, X, N_{av}, R$ )

⋮

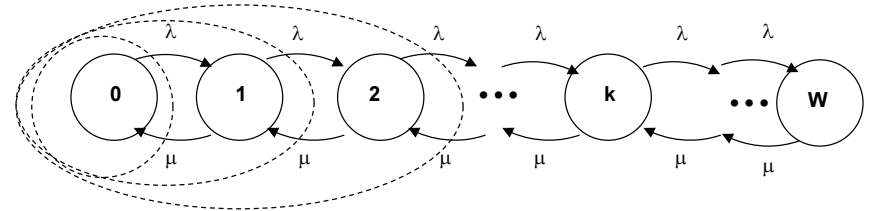
### Simple Server: Infinite Population/Finite Queue

- In this case the queue has a finite length, i.e. the server does not accept any requests when the queue is full ( $W$  requests)
- The possible states are then  $0, 1, 2, \dots, W$
- A request that finds  $k$  (where  $k < W$ ) requests in the system causes a transition to state  $k+1$  at rate  $\lambda$
- A completing request at state  $k$  ( $k = 1, \dots, W$ ) causes a transition to state  $k-1$  with rate  $\mu$

⋮

⋮

### Simple Server: Infinite Population/Finite Queue



$$p_k = p_0(\lambda/\mu)^k \quad k = 1, \dots, W$$

$$p_0 + p_1 + \dots + p_W = p_0 \sum_{k=0}^W (\lambda/\mu)^k = p_0(1 - (\lambda/\mu)^{W+1}) / (1 - \lambda/\mu) = 1$$

$$p_0 = ((1 - \lambda/\mu)) / (1 - (\lambda/\mu)^{W+1})$$

⋮

### Ex. 26: infinite population/finite queue

$\mu = 1 / 0.02 = 50$  requests / sec       $\lambda = 30$  requests / sec      queue length = 4

What is the fraction of time that the server is in state  $k$  ( $k=0, \dots, 4$ ) ?

From  $p_0 = ((1 - \lambda/\mu)) / (1 - (\lambda/\mu)^{W+1})$        $p_0 = (1 - 30/50) / (1 - (30/50)^5) = 0.43$

$p_k = (0.43)(0.6)^k$       for  $k = 1, \dots, 4$

$k=1$        $p_1 = 0.258$

$k=2$        $p_2 = 0.155$

$k=3$        $p_3 = 0.093$

$k=4$        $p_4 = 0.056$

⋮

⋮

:

infinite population/finite queue

- The utilization of the server is the fraction of time it is not idle, i.e.  $U = 1 - p_0$
- Substituting for  $p_0$  and rearranging  

$$U = (\lambda/\mu) [1 - (\lambda/\mu)^W] / [1 - (\lambda/\mu)^{W+1}]$$
- In servers with a finite queue, the fraction of requests that are lost because the queue is full is an important performance metric
- Since requests are only refused when the server is in state  $W$  it follows that  $p_{loss} = p_W$

..... 173

:

infinite population/finite queue

- The average number of requests at the server is

$$N_{av} = \sum_{k=0}^W k p_k = p_0 \sum_{k=0}^W k (\lambda/\mu)^k$$

- We know that

$$\sum_{k=0}^W k (\lambda/\mu)^k = [W a^{W+2} - (W+1) a^{W+1} + a] / (1-a)^2$$

Combining with the expression for  $p_0$  and rearranging

$$N_{av} = (\lambda/\mu) [W(\lambda/\mu)^{W+1} - (W+1)(\lambda/\mu)^W + 1] / [1 - (\lambda/\mu)^{W+1}] (1 - \lambda/\mu)$$

..... 174

:

infinite population/finite queue

- The throughput  $X$  of the server is  $\mu$  when the server is busy (ist utilization) and 0 otherwise

$$X = U\mu + 0(1-U) = \lambda [1 - (\lambda/\mu)^W] / [1 - (\lambda/\mu)^{W+1}]$$

- Applying Little's Law

$$R = N_{av} / X$$

$$R = S [W(\lambda/\mu)^{W+1} - (W+1)(\lambda/\mu)^W + 1] / [1 - (\lambda/\mu)^W] (1 \lambda/\mu)$$

..... 175

:

Ex. 27: infinite population / finite queue

Assuming the same DB server as before with

$$\mu = 1 / 0.02 = 50 \text{ requests / sec} \quad \lambda = 30 \text{ requests / sec}$$

What should the queue length be so that less than 1% of requests are rejected?

Compute  $W$  such that  $p_w = p_0 (\lambda/\mu)^W < 0.01$

$$p_0 = ((1 - \lambda/\mu) / (1 - (\lambda/\mu)^{W+1})) = 0.4 / (1 - 0.6^{W+1})$$

$$\implies (0.4)(0.6)^W / (1 - 0.6^{W+1}) < 0.01$$

Solving the above inequality one obtains  $W = 8$

..... 176

⋮

### Summary: infinite population/finite queue

Fraction of time server has k requests:  $p_k = \frac{1 - \lambda/\mu}{1 - (\lambda/\mu)^{W+1}} (\lambda/\mu)^k \quad k = 0, \dots, W$

Server utilization:  $U = \frac{(\lambda/\mu) [1 - (\lambda/\mu)^W]}{1 - (\lambda/\mu)^{W+1}}$

Average server throughput:  $X = U \mu = \frac{\lambda [1 - (\lambda/\mu)^W]}{1 - (\lambda/\mu)^{W+1}}$

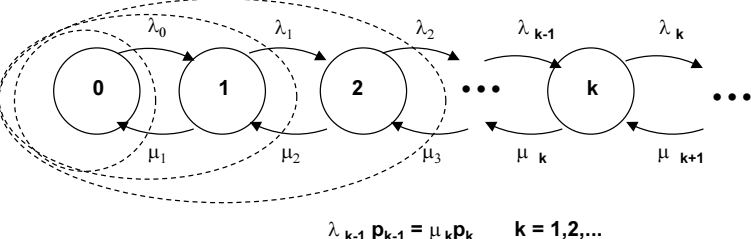
Average number of requests in the server:  $N_{av} = \frac{(\lambda/\mu) [W(\lambda/\mu)^{W+1} - (W+1)(\lambda/\mu)^W + 1]}{[1 - (\lambda/\mu)^{W+1}](1 - \lambda/\mu)}$

Average response time:  $R = \frac{N_{av}}{X} = \frac{S[W(\lambda/\mu)^{W+1} - (W+1)(\lambda/\mu)^W + 1]}{[1 - (\lambda/\mu)^W](1 - \lambda/\mu)}$

⋮

### Generalized System-Level Models

- So far we assumed that arrival and service rates were independent of the state
- In generalized system-level models we introduce state-dependent arrival and service rates



⋮

### Generalized System Level Models

Applying the flow-in = flow-out principle recursively

$$p_1 = (\lambda_0/\mu_1)p_0$$

$$p_2 = (\lambda_1/\mu_2)p_1 = (\lambda_1/\mu_2) (\lambda_0/\mu_1)p_0$$

$$\vdots$$

$$p_k = (\lambda_{k-1}/\mu_k)p_{k-1} = (\lambda_{k-1}/\mu_k) \dots (\lambda_1/\mu_2) (\lambda_0/\mu_1)p_0$$

$$p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}$$

The sum of probabilities is then  $\sum_{k=0}^{\infty} p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} = 1$

It follows that  $p_0 = \left[ \sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \right]^{-1}$

⋮

### Generalized System Level Model Equations

Fraction of time server has k requests:  $p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}$

where  $p_0 = \left[ \sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \right]^{-1}$

Server utilization:  $U = 1 - p_0$

Average server throughput:  $X = \sum_{k=0}^{\infty} \mu_k p_k$

Average number of requests in server:  $N_{av} = \sum_{k=0}^{\infty} k p_k$

Average response time:  $R = N_{av} / X = \frac{\sum_{k=0}^{\infty} k p_k}{\sum_{k=0}^{\infty} \mu_k p_k}$

⋮

Deriving Specialized Models from General Model

Specialized system-level models can be derived from the general equations by selecting

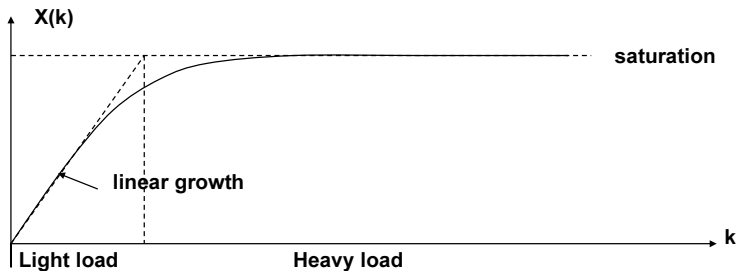
- population size
  - infinite population (open models)
  - finite population (closed models)
- service rate at the server
  - fixed service rate  $X(k) = \mu$
  - variable service rate  $X(k) = \mu_k$
- queue size
  - infinite queue
  - finite queue

⋮ . . . . . 181

⋮

Infinite population, infinite queue, variable rate

- Throughput of the server is usually a function of the number of requests present in the system



- At light loads there is little contention for internal queues, at heavy loads the bottleneck device limits the throughput

⋮ . . . . . 182

⋮

Infinite population, infinite queue, variable rate

- Assume variable arrival rate  $\lambda_k = \lambda$  for  $k = 0, 1, \dots$  and variable service rate  $\mu_k = X(k)$  for  $k = 1, 2, \dots$
- Parting from the general expressions for  $p_0$  and  $p_k$

$$p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \qquad p_0 = \left[ \sum_{k=0}^{\infty} \prod_{i=0}^k \frac{\lambda_i}{\mu_{i+1}} \right]^{-1}$$

$$p_k = \left[ 1 + \sum_{k=0}^{\infty} \frac{\lambda^k}{\prod_{i=1}^k X(i)} \right]^{-1} \frac{\lambda^k}{\prod_{i=1}^k X(i)} \qquad \mu_k = \begin{cases} X(k) & k \leq J \\ X(J) & k > J \end{cases}$$

⋮ . . . . . 183

⋮

Infinite population, infinite queue, variable rate

Substituting the values for  $\mu$  and using the following definitions

$$\beta(k) = X(1) \times X(2) \times \dots \times X(k) \qquad \rho = \lambda / X(J)$$

$$p_0 = \left[ 1 + \sum_{k=1}^J \frac{\lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \frac{\rho}{1-\rho} \right]^{-1}$$

$$p_k = \begin{cases} p_0 \lambda^k / \beta(k) & k \leq J \\ p_0 X(J)^J \rho^k / \beta(J) & k > J \end{cases}$$

$$N_{av} = p_0 \left[ \sum_{k=1}^J \frac{k \lambda^k}{\beta(k)} + \frac{\rho \lambda^J [\rho + (J+1)(1-\rho)]}{(1-\rho)^2 \beta(J)} \right]$$

⋮ . . . . . 184

:

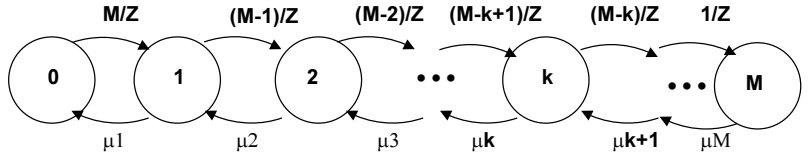
### Single queue, fixed service rate, finite population

- These closed models apply mainly to intranet situations, e.g. a finite number of clients accessing the DB server
- In this situation it is common to have a think time Z at the client, i.e. each client submits a request every 1/Z seconds
- If there is a fixed number M of clients, the rate going from state 0 to state 1 is M/Z, from state 1 to state 2 it is (M-1)/Z, etc.

..... 185

:

### Single queue, fixed service rate, finite population



- Considering a fixed service rate  $\mu_k = \mu$  for  $k=1...M$

$$p_k = p_0 \frac{M!}{(M - k)! (\mu Z)^k} \quad k = 0, \dots, M$$

$$p_0 = \left[ \sum_{k=0}^M \frac{M!}{(M - k)! (\mu Z)^k} \right]^{-1}$$

..... 186

:

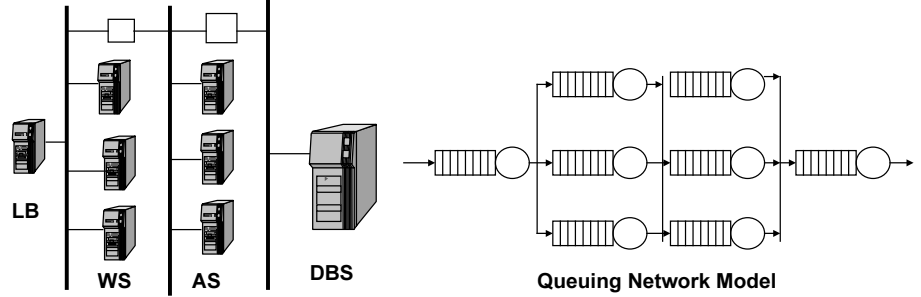
### Other Situations

- By making the appropriate assumptions and following the same derivation principles it is possible to derive the probabilities, throughput, average number of users and response time for other cases, e.g. finite population and variable service rate
- For derivations and the corresponding sets of system-level model equations see Menascé, Daniel A., Almeida, Virgilio A. F.; "Capacity Planning for Web Performance: Metrics, Models, & Methods", Prentice Hall PTR, 1998

..... 187

:

### Component-Level Performance Models



- Want to models that are capable of representing individual components and their interaction
- Queuing Network Models are collections of queues arranged in the same configuration as real system

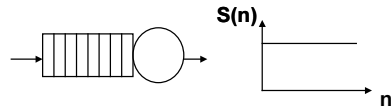
..... 188

:

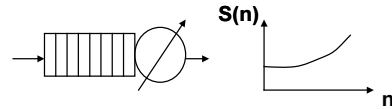
## Queuing Networks

- Queuing Networks (QNs) are made up of individual queues that can be

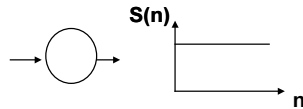
- load independent



- load dependent



- delay resources



. . . . . 189

:

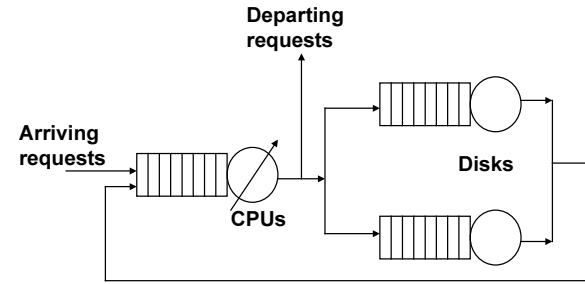
## Queuing Networks

- Queuing Networks can be characterized by the classes of requests they handle
- We distinguish
  - single class vs. multiple classes
  - open vs. closed
- Will analyze the cases for
  - single class open QNs (DBS w. 1 type of query)
  - single class closed QNs (DBS w. fixed # threads)
  - multiple class open QNs (DBS w. query & update)
  - multiple class closed QNs (DBS w. query & update & fixed # of threads)

. . . . . 190

:

## Ex. 28: Open Queuing Network

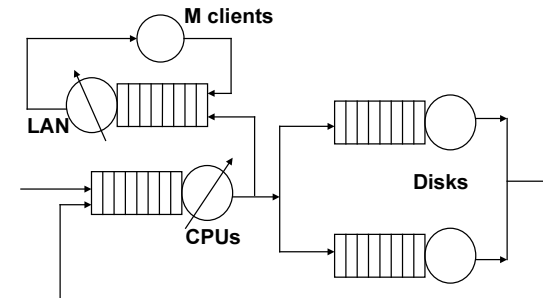


The DB server runs on a 4-CPU mainframe modelled as a single queue with a load-dependent resource

. . . . . 191

:

## Ex. 29: Closed QN



DB server receiving requests from a fixed number of clients  
 Clients are modelled as delay resource because returning answers don't queue  
 Service demand at clients is the average time spent there before submitting another request, i.e. the think time

. . . . . 192



:

### Single Class Open QNs

- Examine first case where all resources are either delay or load-independent resources
- Notation:
  - $\lambda$ : average arrival rate of requests to QN
  - K: number of queues
  - $X_0$ : average throughput of the QN (for open QNs in equilibrium  $X_0 = \lambda$ )
  - $V_i$ : average number of visits to queue i by a request
  - $S_i$ : average service time of a request at queue i per visit
  - $W_i$ : average waiting time of a request at queue i per visit

:

### Notation QNs (cont.)

- $X_i$ : average throughput of queue i
- $R_i$ : average response time of a request at queue i, defined as  $R_i = W_i + S_i$
- $R_i'$ : average residence time of a request to queue i, defined as total waiting time (i.e. the queuing time) + total service time (i.e. the service demand) over all visits to queue i ( $R_i' = V_i R_i$ )
- $R_0$ : average response time (i.e. the sum of the residence times over all queues)
- $n_i$ : average number of requests in queue i
- N: average number of requests in the QN

:

### Single Class Open QN

- Want to compute average response time  $R_i$  at queue i
  - $$R_i = W_i + S_i$$
- The average waiting time depends on the number of requests seen by the incoming request, i.e. the average number of requests in the queue upon arrival (Arrival Theorem)
  - $$R_i = n_i S_i + S_i$$
- From Little's Law  $n_i = X_i R_i$  and from Utilization Law  $U_i = X_i S_i$  and
  - $$R_i = S_i / (1 - U_i)$$

:

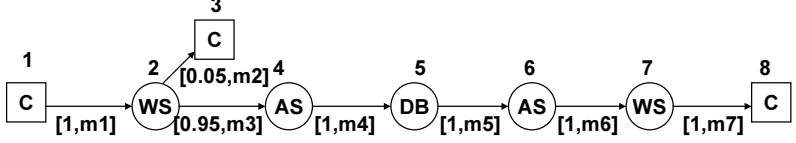
### Single Class Open QN

- The residence time at queue i is then
  - $$R_i' = V_i R_i = V_i S_i / (1 - U_i) = D_i / (1 - U_i)$$
- Using again Little's Law and the Utilization Law we obtain the average number of requests at queue i as
  - $$n_i = U_i / (1 - U_i)$$

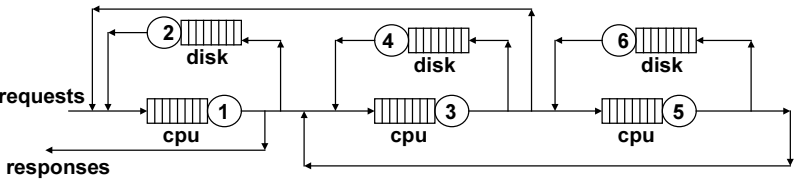
⋮

## Ex. 30: Single Class Open QN

One of the most used e-business functions of an on-line brokerage is the show portfolio function (CSID below).

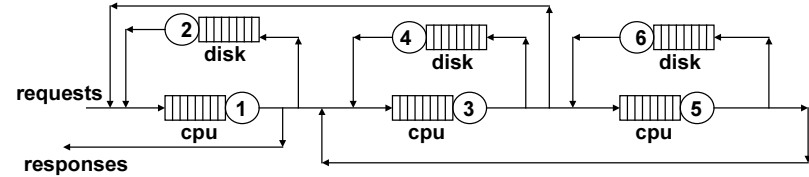


Project management is concerned with scalability as traffic to the site grows. The site is therefore modelled as an open, single class queuing network



⋮

## Ex. 30 (cont.): Single Class Open QN



The following measurements were taken

Server	$D_{CPU}(msec)$	$D_{I/O}(msec)$
Web	19	12
Appl.	40	48
DB	35	56

For arrival rate  $\lambda = 9$  req/sec

$$U_i = X_{i0} D_i = \lambda D_i = (9)(0.019) = 0.171$$

$$R_1' = (0.019)/(1 - 0.171) = 0.23 \text{ sec}$$

$$R_i = S_i / (1 - U_i)$$

$$R_i' = D_i / (1 - U_i)$$

$$Rsp = R_1' + R_2' + R_3' + R_4' + R_5' + R_6'$$

$$R_2' = 0.013$$

$$R_3' = 0.063$$

$$R_4' = 0.085$$

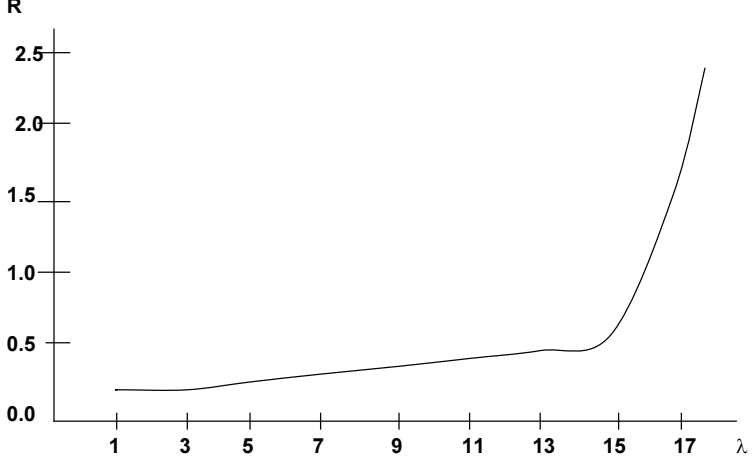
$$R_5' = 0.051$$

$$R_6' = 0.113$$

$$Rsp = 0.347$$

⋮

## Response Time vs. Arrival Rate



⋮

## Single Class Closed QN-Models

- Typically used whenever there is a fixed and finite number of requests in the system
  - model the maximum level of multiprogramming under heavy load
  - multithreaded Web server
  - database server with limited number of connections
- Will apply Mean Value Analysis (MVA)
- MVA applies recursively three equations:
  - residence time equation
  - throughput equation
  - queue length equation

⋮

### Mean Value Analysis

- Consider a closed QN with n requests
- Compute the response time  $R_i(n)$  per visit to i

$$R_i(n) = S_i + W_i(n)$$

- Waiting time is the time to service all the requests that an arriving request found in the queue

$$W_i(n) = n_i^a(n) S_i$$

- Substituting

$$R_i(n) = S_i + n_i^a(n) S_i = S_i [1 + n_i^a(n)]$$

⋮ . . . . . 201

⋮

### MVA - Arrival Theorem

- Arrival Theorem for closed QNs states that the yaverage number of requests seen upon arrival to queue i when there are n requests in the queue i, is equal to the average number of requests in a QN with n-1 requests (i.e. arriving request doesn't see itself)

$$n_i^a(n) = n_i(n-1)$$

- Combining

$$R_i(n) = S_i [1 + n_i^a(n)] = S_i [1 + n_i(n-1)]$$

⋮ . . . . . 202

⋮

### MVA - Residence Time and Throughput

- The residence time to a queue i is given by the response time multiplied by the visits  $V_i$  to queue i

$$R'_i(n) = V_i R_i(n) = V_i S_i [1 + n_i(n-1)] = D_i [1 + n_i(n-1)]$$

- Adding the residence times for all queues i one obtains the response time  $R_0$

$$R_0(n) = \sum_{i=1}^K R'_i(n)$$

- Applying Little's Law

$$X_0(n) = n/R_0(n) = n / \sum_{i=1}^K R'_i(n)$$

⋮ . . . . . 203

⋮

### MVA - Queue Length Equation

- The queue length can be obtained by applying Little's Law and the Forced Flow Law to queue i

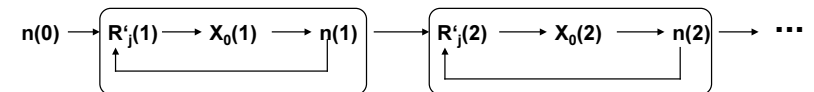
$$n_i(n) = X_i(n) R_i(n)$$

$$n_i(n) = X_0(n) V_i R_i(n)$$

$$n_i(n) = X_0(n) R'_i(n)$$

$$n_i(n) = X_0(n) D_i [1 + n_i(n-1)]$$

- To calculate the queue length with n requests we need to know the queue length with n-1 requests



⋮ . . . . . 204

## Single-Class MVA (no load-dep. Resource)

Residence Time Equation:

$$R'_i(n) = \begin{cases} D_i & \text{Delay resource} \\ D_i[1 + n_i(n-1)] & \text{Queuing resource} \end{cases}$$

Throughput Equation:

$$X_0(n) = \frac{n}{\sum_{i=1}^K R'_i(n)}$$

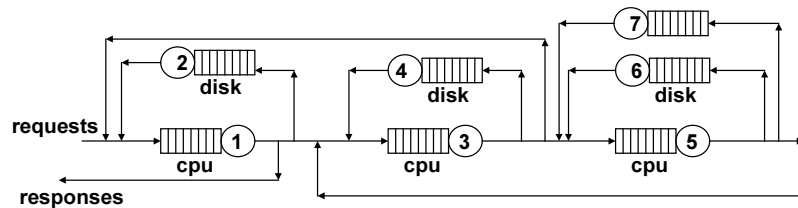
Queue Length Equation:

$$n_i(n) = X_0(n) R'_i(n)$$

205

## Ex. 31: Closed QN solved with MVA

Consider a 3-tier architecture represented by the following QN:



Every request to the database requires a database connection. However, the DB connections are limited and the DB server becomes the bottleneck as traffic increases. If the database is limited to 5 connections, what is the throughput of the DB server during the peak hour? What is the average response time of the search transaction?

Considering that the number of transactions executed by the DB server during peak hours is constant, we can model the system as a closed QN.

206

## Ex. 31(cont.): Closed QN solved with MVA

$D_{cpu} = 21 \text{ msec}$

$D_{disk6} = 17 \text{ msec}$

$D_{disk7} = 28 \text{ msec}$

$$R'_i(n) = D_i[1 + n_i(n-1)]$$

$$X_0(n) = \frac{n}{\sum_{i=1}^K R'_i(n)}$$

$$n_i(n) = X_0(n) R'_i(n)$$

n	$R'_{cpu}$	$R'_{disk6}$	$R'_{disk7}$	$R_0$	$X_0$
0	0.00	0.00	0.00	0.00	0.00
1	21.00	17.00	28.00	66.00	15.15
2	27.00	21.00	41.00	89.00	22.41
3	33.00	24.00	55.00	113.00	26.52
4	39.00	27.00	72.00	138.00	29.07
5	43.00	29.00	90.00	163.00	30.75

What would happen if the number of connections is increased, e.g. to 10,20 or 30?

n	$X_0$
10	34.09
20	35.29
30	35.52

Bottleneck is disk 7:  $1/0.28 = 35.71 \text{ tps}$

207

## Ex. 32: Closed QN

A DB server receives requests from 50 clients. Each request requires 5 records to be read from a single disk. The average read time is 9msec, the CPU-time is 15 msec.

What is the throughput of the server?

What is the average time spent at the CPU and the disk by each request?

What is the average number of requests at the CPU and disk?

What is the average response time as a function of the concurrent requests?

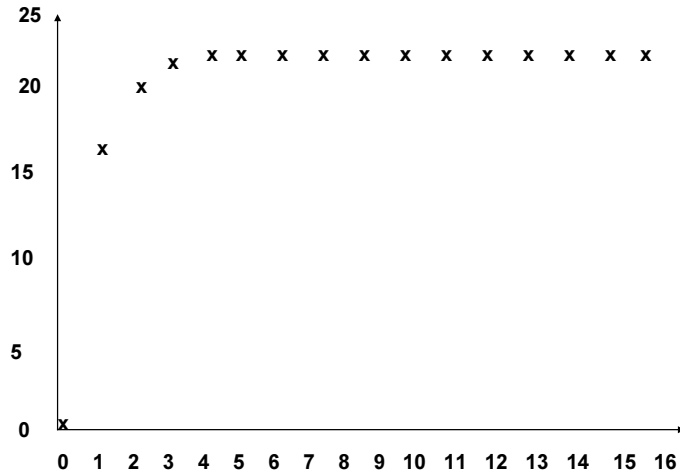
$D_{disk} = (5)(9) = 45 \text{ msec}$

$D_{cpu} = 15 \text{ msec}$

n	$R'_{cpu}$	$R'_{disk}$	$R_0$	$X_0$	$n_{cpu}$	$n_{disk}$
0	0	0	0	0	0	0
1	15.0	45.0	60.0	0.0167	0.250	0.750
2	18.75	78.75	97.50	0.0205	0.385	1.615
3	20.77	117.69	138.46	0.0217	0.450	2.550
4	21.75	159.75	181.50	0.0220	0.479	3.521
5	22.19	203.43	225.62	0.0222	0.492	4.508
6	22.38	247.87	270.25	0.0222	0.497	5.503
7	22.45	292.64	315.10	0.0222	0.499	6.501

208

### Ex. 32: Throughput vs Concurrent Requests



209

### Bounds for Closed QNs

- The maximum throughput is always limited by the device that is the bottleneck

$$\lambda \leq 1/\max D_i \quad X_0(n) \leq 1/\max D_i$$

$$X_0(n) = \frac{n}{\sum_{i=1}^K R'_i(n)} \leq \frac{n}{\sum_{i=1}^K D_i}$$

$$X_0(n) \leq \min \left[ \frac{n}{\sum_{i=1}^K R'_i(n)}, \frac{1}{\max_{i=1}^K D_i} \right]$$

- Under light load, throughput grows linearly at rate  $1/D_i$  and flattens to  $1/\max D_i$
- At maximum value  $R_0 \approx n/X_{\max}$  and  $R_0(n) \approx n_{\max} D_i$

210

### Bounds for Closed QNs (cont.)

- For very small values of  $n$  the response time is equal to the sum of the service demands of all resources (no queuing situation)
- A lower bound on the response time is then

$$R_0(n) \geq \max \left[ \sum_{i=1}^K D_i, n \max D_i \right]$$

211

### Ex. 33: Bounds on Closed QNs

Consider the DB server of the previous example but with the following changes  
 a) more indexes were built, so the average number of reads drops from 5 to 2.5  
 b) a faster disk was installed dropping the average service time to 5.63 msec  
 c) a CPU that is twice as fast was installed, i.e. service demand at CPU = 7.5 msec

Scenario	Dcpu	Ddisk	$\sum D_i$	$1/\max D_i$	Bottleneck
a	15	(2.5)(9)=22.5	37.5	0.044	disk
b	15	(5)(5.63)=28.15	43.15	0.036	disk
c	7.5	45	52.5	0.022	disk
a+b	15	(2.5)(5.63)=14.08	29.08	0.067	CPU
a+c	7.5	(2.5)(9) = 22.5	30.00	0.044	disk

212

## Multiple Class Open Queuing Networks

- Multiple Class QNs are needed for more realistic scenarios or scenarios in which there is no single dominant request class
- We must identify the class in each queue (subscripts now become i,r)
- Different classes may have different arrival rates  $\lambda_1, \dots, \lambda_r, \dots, \lambda_R$
- The set of all arrival rates is denoted as the vector  $\vec{\lambda}$  denoted henceforth by  $\underline{\lambda}$

213

## Summary of Equations for Multiclass Open QNs

Input parameters:  $D_{i,r} \quad \lambda_r$

Utilization:  $U_{i,r}(\underline{\lambda}) = \lambda_r V_{i,r} S_{i,r} = \lambda_r D_{i,r}$

$U_i(\underline{\lambda}) = \sum_r U_{i,r}(\underline{\lambda})$

Average number of requests of class r at resource i:

$$n_{i,r}(\underline{\lambda}) = U_{i,r}(\underline{\lambda}) / (1 - U_i(\underline{\lambda}))$$

Average residence time of requests of class r at resource i:

$$R_{i,r}^t(\underline{\lambda}) = \frac{D_{i,r}}{D_{i,r} / (1 - U_i(\underline{\lambda}))} \quad \begin{array}{l} \text{Delay resource} \\ \text{Queuing resource} \end{array}$$

Average response time for requests of class r:

$$R_{0,r}(\underline{\lambda}) = \sum_{i=1}^K R_{i,r}^t(\underline{\lambda})$$

Average number of requests at resource i:

$$n_i(\underline{\lambda}) = \sum_{r=1}^R n_{i,r}(\underline{\lambda})$$

214

## Ex. 34: Multiple-Class Open QN

A DB server is subject to 2 classes of transactions: queries arrive at a rate of 5 tps and updates at a rate of 2 tps. Under the service demands given below, what are the response times, the residence times and the utilizations?

	Queries	Updates
Arrival Rate (tps)	5	2
Service Demands (sec)		
CPU	0.10	0.15
Disk1	0.08	0.20
Disk2	0.07	0.10
Utilizations (%)		
CPU	50	30
Disk1	40	40
Disk2	35	20
ResidenceTime (sec)		
CPU	0.50	0.75
Disk1	0.40	1.00
Disk2	0.16	0.22
ResponseTime (sec)	1.06	1.97

215

## Multiple Class Closed QNs

- In a closed multiple class QN there is a fixed number of requests  $N_r$  in the system for each class r
- $\underline{N} = (N_1, \dots, N_r, \dots, N_R)$  is the vector of the load intensity for all classes of requests
- $\underline{l}_r$  is a vector where all components are 0 except for the rth component which is equal to 1, e.g.  $\underline{l}_2 = (0, 1, 0)$
- $R_{i,r}^t(\underline{N})$  stands for the residence time of class r requests at queue i when the number of class 1 requests in the system is  $N_1$ , the number of class 2 requests is  $N_2, \dots$

216

⋮

## MVA equations for multiclass closed QNs

Residence time equation for class r at queue i:

$$R_{i,r}^c(\underline{N}) = \frac{D_{i,r}}{D_{i,r} [1 + n_{i,r}(\underline{N} - l_r)]}$$

Delay resource  
Queuing resource

Throughput equation for class r:

$$X_{0,r}(\underline{N}) = \frac{N_r}{\sum_{i=1}^K R_{i,r}^c(\underline{N})}$$

Queue length equation for class r at queue i:

$$n_{i,r}(\underline{N}) = X_{0,r}(\underline{N}) R_{i,r}^c(\underline{N})$$

Queue length equation for queue i:

$$n_i(\underline{N}) = \sum_{r=1}^R n_{i,r}(\underline{N})$$

⋮ 217

⋮

## Solving closed multiclass QNs, Schweitzer Approx.

- To compute the residence time of a single queue we need to compute multiple queue lengths for the various classes and loads, e.g. for a QN with 2 classes and  $\underline{N} = (2,4)$  the computation of the residence time at queue i for class 1 for  $\underline{N} = (2,4)$  requires the queue length at queue i for  $\underline{N} = (1,4)$ . The computation of the residence time for the same queue for class 2 and the same load intensity vector  $\underline{N} = (2,4)$  requires the queue length for  $\underline{N} = (2,3)$
- In general for  $\underline{N}$  we need queue lengths for load intensity vectors  $\underline{N} - l_1, \underline{N} - l_2, \dots, \underline{N} - l_R$

⋮ 218

⋮

## Schweitzer Approximation

- Schweitzer observed/assumed that the number of class r requests in each queue increases proportionally with the number of class r customers in the QN
- From this observation it follows that
 
$$n_{i,r}(\underline{N} - l_r) / n_{i,r}(\underline{N}) = (N_r - 1) / N_r$$

$$n_{i,r}(\underline{N} - l_r) = n_{i,r}(\underline{N}) (N_r - 1) / N_r$$
- This approach requires an iterative solution (begin with guess, typically use equal distribution)

⋮ 219

⋮

## Ex. 35: Schweitzer Approximation

Iteration	Query			Thput	Update			Thput
	Queue Length				Queue Length			
	CPU	Disk1	Disk2		CPU	Disk1	Disk	
1	1.667	1.667	1.667	12.63	0.667	0.667	0.667	2.69
2	0.7576	3.0303	1.2121	10.18	0.3226	1.2903	0.6667	2.13
3	0.3927	3.8396	0.7678	9.22	0.1633	1.5911	0.2456	1.90
4	0.2725	4.1785	0.5489	8.88	0.1122	1.7152	0.1726	1.82
5	0.2363	4.3055	0.4582	8.77	0.0969	1.7601	0.1430	1.79
6	0.2255	4.3511	0.4235	8.73	0.0922	1.7760	0.1318	1.78
7	0.2221	4.3674	0.4106	8.71	0.0908	1.7816	0.1276	1.78
8	0.2210	4.3732	0.4059	8.71	0.0903	1.7836	0.1261	1.78

⋮ 220

:

## Contention for Software Servers

- So far we have modeled the effect of contention for hardware resources (CPU, I/O devices, etc.)
- Contention also exists for software resources (threads in a software server, database items and their locks, etc.)
- Software resources often are kept for a longer time and often in parallel with other resources
- Total response time is

$$t_{SW\ contention} + t_{HW\ contention} + \text{service demands}$$

..... 221

:

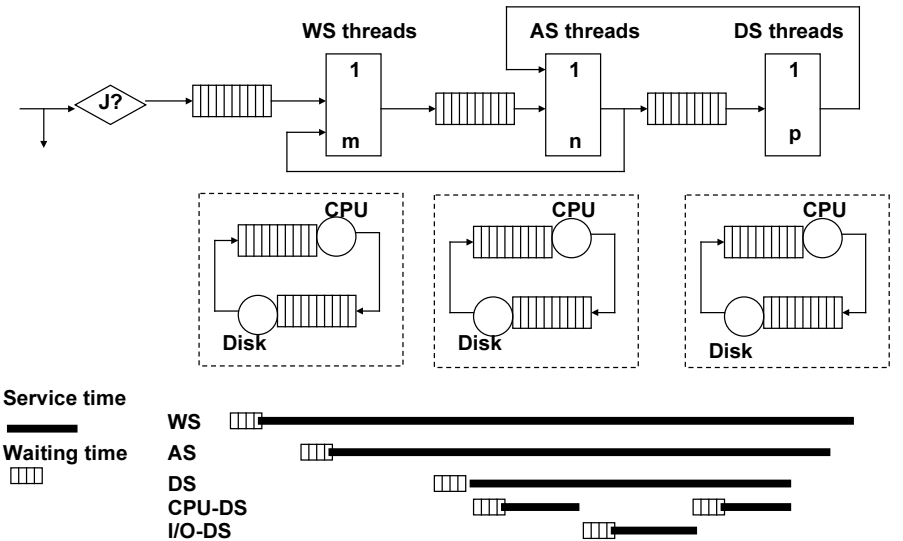
## Modeling Software Contention

- The main difference between modeling hardware and software contention lies in a basic assumption: in modeling hardware contention we assumed that no request is allowed to hold simultaneously more than one resource
- Under certain conditions exact solutions can be found for queuing networks.
- These solutions are called product-form solutions
- Product-form solutions don't allow simultaneous resource possession

..... 222

:

## Ex. 36 : Simultaneous Resource Possession



..... 223

:

## Solving Simultaneous Resource Possession QNs

- Solving simultaneous resource possession QNs requires approximate solutions
- Approximate methods include
  - Method of Layers
  - Stochastic Rendez-vous Networks
- An alternate approach is based on simulations (usually preferred for complex systems where analytical solutions are hard to obtain)

..... 224



:

## Layered Queuing Networks (LQN)

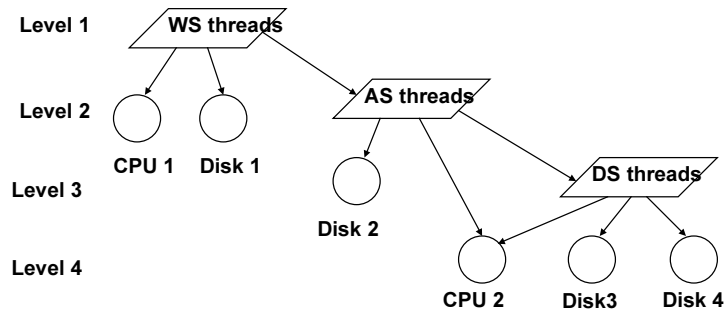
- LQNs are well suited for representing hardware and software contention
- Processes with similar behavior form a class of processes that may invoke services from a lower level
- Services at lower levels may be either software or hardware resources

225

:

## Ex. 37: Layered Queuing Network

Consider a 3-tiered e-commerce site with the Web server running on one machine and the Application Server and the Database Server running on another, common machine but access different disks. Contention and simultaneous holding of resources are as depicted in example 36.



226

:

## Method of Layers (MOL)

- Method of Layers consists in decomposing a layered queuing network into a sequence of two-level QN submodels
- Each submodel is solved using Mean Value Analysis techniques
- Performance estimates for each subnetwork are calculated and used as inputs for subsequent levels
- The iterative technique starts by assuming no HW or SW contention. Algorithm stops iterating when response times of successive groups reach a fixed point.

227

:

## Simulation of HW and SW Contention

- Simulation is the technique of choice when obtaining exact models is difficult or impossible for the system to be modeled.
- A network of queues is used to represent the modeled system and events, such as arrivals of requests, are randomly generated
- Counters accumulate the metrics of interest (e.g. waiting time, length of queues, etc.)
- Simulations can be made as precise as needed (but more precise models require more input data)
- (Approximate) analytic models are enough for scalability analysis and capacity planning.

228

:

## The Cost of Security

- E-commerce systems are vulnerable to attack
- Therefore, security is a major issue.
- But security is expensive in terms of performance
- The throughput of Web servers is significantly reduced when they have to deal with secure sessions
- Sources of performance loss may be e.g. the firewall, the Transport Layer Security protocol, encryption needed for authentication, etc.

..... 229

:

## Categories of Security

- Authentication: process by which 2 parties are given a guarantee that they are interacting with the „right“ person
  - server authentication (no impostor between client and actual server)
  - client authentication (order is placed by someone known and registered at the site, e.g. online banking)
- Confidentiality: protection of the content of a message (e.g. protect credit card information when sending it over the Internet)

..... 230

:

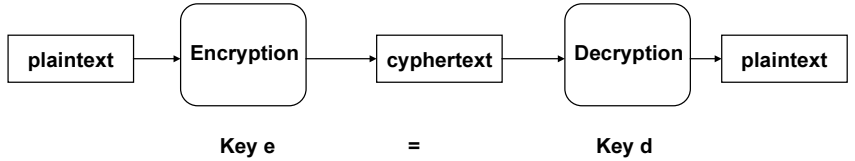
## Categories of Security (cont.)

- Data integrity: ensures that data is not modified by an unauthorized party (e.g. modifying shipping address for a book order)
- Availability: can be compromised by a denial of service attack, e.g. by swamping the site with meaningless work, such as repeated authentication, and preventing it from doing productive work
- Non-repudiation: prevents the sender of a message from denying it (e.g. placement of orders)

..... 231

:

## Symmetric Cryptography

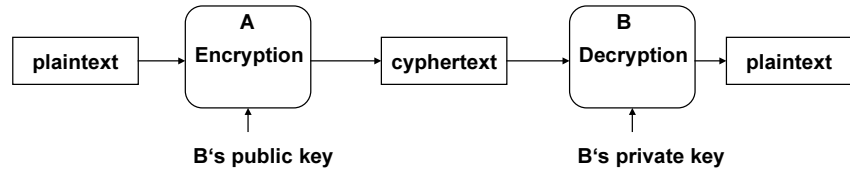


- In symmetric cryptography the plaintext message is encrypted with the same key used by the decryption algorithm
- Advantage: cheap
- Disadvantage: key must be shared by sender and receiver. How accomplish sharing?

..... 232

:

## Public Key Cryptography



- Public Key Cryptography uses 2 different keys:
  - private key known only to receiver of message
  - public key associated with a receiver, used by sender for encryption
- A encrypts Msg by using PKEncrypt algorithm and B's (small) public Key
- B decrypts Msg' by using PKDecrypt algorithm and B's (large) private key

..... 233

:

## Cost of Encryption

- The various security categories can be supported by the use of cryptography
- Cryptography is expensive, especially public key encryption (modular exponentiation)
  - private key operation time grows with  $k^3$  ( $k$  = key length)
  - public key operation time grows with  $k^2$

Key size (bits)	Public Key Op. Time (msec)	Private Key Op. Time (msec)
512	3.5	39.9
768	7.3	112.2
1024	12.8	255.8
1280	19.0	455.5
1536	26.8	771.8
1792	36.2	1214.8
2048	46.8	1796.0

..... 234

:

## Cost of Encryption

- The public key operation time (even for a small 128 byte block) is of the same order of magnitude as a disk access
- The private key operation time is from 1 to 2 orders of magnitude larger than a disk access
- ==> PK cryptography cannot be used for bulk data transfer
- Authentication protocols use PK crypto to exchange the key and use symmetric crypto for bulk messages

..... 235

:

## Digital Signatures - Signing

- Digital signatures can be used to sign a document digitally so that its authenticity can later be validated
- Three steps are involved in producing a digitally signed document:
  - hashing the message to produce a message digest
    - it should be easy to compute  $h(Msg)$
    - it should be hard to obtain  $Msg$  given  $h(Msg)$
    - it should be difficult to find a  $Msg'$  such that  $h(Msg)=h(Msg')$
  - encrypting the hashed message digest using private key
  - sending (original message, encrypted message digest)

..... 236

:

## Digital Signatures - Verification

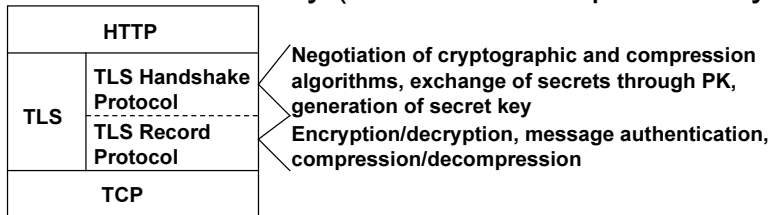
- Verifying a digital signature involves:
  - B receives the pair (Msg, Encrypt(MD, Key<sup>A<sub>priv</sub></sup>))
  - B computes the message digest MD using the same hash function as A
  - B decrypts the encrypted message digest (MD, Key<sup>A<sub>priv</sub></sup>) received from A using A's public key
  - The result of the decryption done by B should be the original hashed form of the message digest
  - If the decrypted form of the message digest does not correspond to the MD computed by B, either the message was corrupted or the sender was not A

..... 237

:

## Authentication Protocols

- Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) protocol offer authentication, confidentiality and non-repudiation.
- SSL and TLS run on top of TCP and allow a server to authenticate itself by presenting to the client a verifiable certificate containing a public key and by demonstrating that it can decrypt a message produced with that key (i.e. it owns the private key)



..... 238

:

## TLS Protocol

- TLS consists of two parts:
  - TLS Record Protocol
    - compresses data
    - applies Message Authentication Code
    - encrypts data using symmetric encryption
  - TLS Handshake Protocol
    - selects the PK algorithm (e.g. RSA) and key used for transmission of shared secret
    - selects bulk encryption algorithm (e.g. DES) and secret keys to be used during the session by the Record Protocol
    - selects the compression protocol to be used by the Record Protocol
- During handshake phase server authenticates itself with certificates to client, client optionally to server

..... 239

:

## Authentication with Certificates

- When a server authenticates itself to a client through a certificate it must present to the client a certificate signed by a trusted authority.
- The Certification Authority endorses the identity of the sites registered with it.
- Server information (e.g. name, issuer CA, serial number, validity) and the server's public key are part of the certificate along with a digest of the server information encrypted with the CA's private key ==> standard X.509 certificate

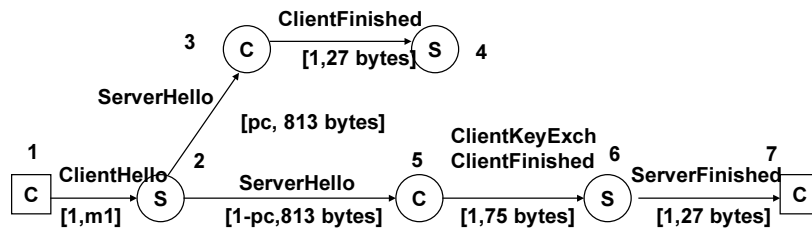
..... 240

## Authentication with Certificates (cont.)

- Browsers have a list of trusted CAs.
- List of trusted CAs includes public key for each CA.
- When a browser receives a server certificate it checks for issuing CA and retrieves its public key
- The CA's public key is used to decrypt the message digest in the certificate
- The browser uses the same hash function used to create the message digest and recreates the message digest from the server information
- If the recreated digest matches the decrypted digest, the server is authenticated

241

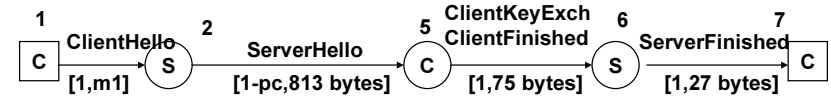
## Description of TLS



- Two possible paths:
  - 1,2,3,4 (cached state)
  - 1,2,5,6,7 (setting up from scratch)
- Caching of session states allows for faster session establishment

242

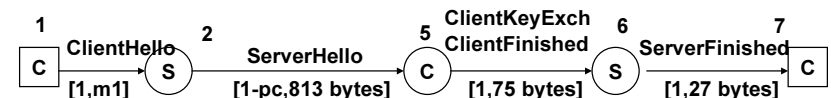
## Steps in TLS Protocol w. Full Handshake



- Client sends „ClientHello“ message to start handshake.
- Message contains:
  - a 28 byte random number
  - timestamp at client
  - session ID (0 to 32 bytes)
  - set of cryptographic algorithms supported by the client (cypher suites) for key exchange, bulk encryption, and message authentication (2 bytes)
  - compression method (1 byte)
  - protocol version (1 byte)

243

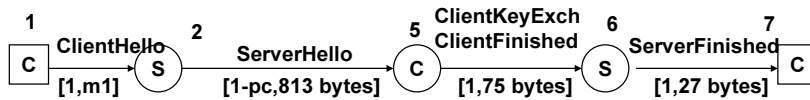
## Steps in TLS Protocol w. Full Handshake (cont.)



- Server answers with a „ServerHello“ message containing
  - X.509 server certificate (750 bytes)
  - server random number (28 bytes)
  - server session ID different from the client ID (0 to 32 bytes)
  - cypher suites supported by the server (2 bytes)
  - compression method supported by server (1 byte)
- Client receives „ServerHello“, authenticates the server through its certificate and produces a premaster secret (48 bytes). Premaster secret is encoded with the server's public key and sent in a „ClientKeyExchange“ message

244

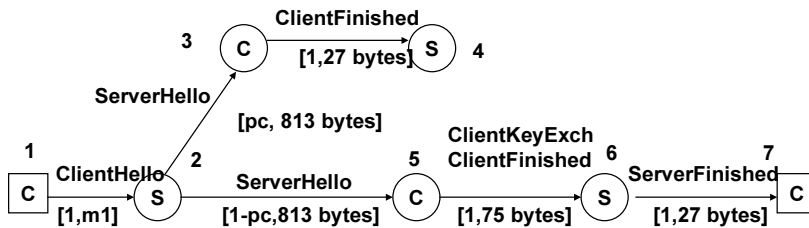
## Steps in TLS Protocol w. Full Handshake (cont.)



- Client generates the common key to be used for bulk encryption from the premaster secret and the client and server random numbers.
- Client sends a „ClientFinished“ (27 bytes) to the server to conclude the handshake.
- Server receives „ClientKeyExchange“ message, decrypts premaster secret using its private key, and generates the common key for bulk encoding from the premaster and the client and server random numbers.
- For verification purposes the server encrypts a digest of all previous messages using the key for bulk encryption and sends it to the client in a „serverFinished“ message to end the handshake

245

## Steps in TLS Protocol w Session Caching



- If the client establishes a new session while its session state is cached at server, TLS can skip authentication and secret negotiation steps.
- If client wants to establish a session reusing cached state it sends the session number it wants to reuse.
- Server answers with the same session ID and new random numbers to generate the new bulk transfer key from the cached state and the new random numbers. No PK must be exchanged.

246

## Analysis of TLS Protocol

- Full handshake adds 2 roundtrip times (RTT) between client and server to the time it would require to fulfill the HTTP request.
- Round Trip latency for
  - slow Internet ~ 160 msec
  - fast Internet ~ 90 msec
- TLS adds between 180 and 320 msec to perceived response time during authentication

247

## Analysis of TSL Protocol (cont.)

- The byte overhead of a TLS connection (full handshake) is 983 bytes.
- The average size of a page returned by an HTTP request is ~ 4 Kbytes ==> overhead ~ 25%
- Over a slow modem connection effective transmission rate is ~ 32 Kbps
- The byte overhead causes  $(983/4096) = 240$  msec additional latency.

248

## Ex. 38: Impact of TLS on Server Throughput

Timings for Client Operations during TLS Handshake (msec):

Key Size (bits)	Verification of Server Certificate	Encryption of Master Secret w/Public Key	Key Generation from Master Sec.	Total Time
512	2.40	1.31	0.1	3.81
768	3.61	2.16	0.1	5.87
1024	7.09	5.20	0.1	12.39

Timings for Server Operations during TLS Handshake (msec)

Key Size (bits)	Decryption of Master Sec. with Private Key	Generation of Keys from Master secret	Total Time
512	10.13	0.10	10.23
768	23.66	0.10	23.76
1024	47.93	0.10	48.03

Encryption/Decryption and Message Digest Generation/Verification Rates (Mbps)

Encryption/Decryption		MD Generation/Verification	
RC4	140	MD5	180
DES	40	SHA	130
3DES	15	SHA1	130

249

## Ex. 38: Impact of TLS on Server Throughput

Problem assumptions:

- Clients are connected to server via high speed LAN
- Clients continuously request files that are 16 384 bytes long
- Average disk access time to retrieve a file is 10 msec
- Average CPU time to process a file (without secure connection) is 2 msec

What is the impact on throughput (in requests/sec) due to TLS using RC4 and MD5?

The impact depends on the key sizes, therefore consider 512, 768, and 1024 bit keys

Must evaluate first service demands at client, network, server CPU and server disk

Time spent at client:  $t_{client} = t_{handshake} + t_{decryption} + t_{verification}$

For a key size of 1024 bits

$$t_{client} = 0.01239 + (16\ 384)(8) / (140\ 000\ 000) + (16\ 384)(8) / (180\ 000\ 000) = 0.01405\ sec$$

250

## Ex. 38: Impact of TLS on Server Throughput

Time spent at server:  $t_{server} = t_{service} + t_{handshake} + t_{encryption} + t_{digest\ generation}$

$$t_{server} = 0.002 + 0.04803 + (16\ 384)(8) / (140\ 000\ 000) + (16\ 384)(8) / (180\ 000\ 000) = 0.05169\ sec$$

Service demands using RC4 and MD5 for all resources and all key sizes:

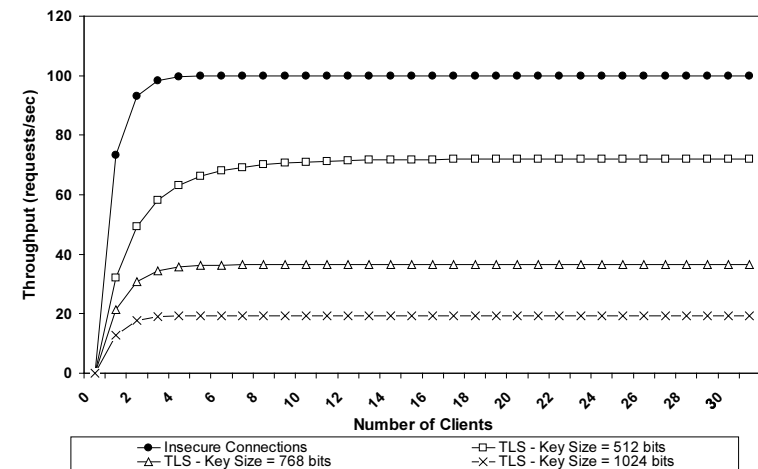
Resource	512	768	1024	insecure
Client	5.474	7.534	14.054	0
Network	1.737	1.737	1.737	1.600
Server CPU	13.894	27.424	51.694	2.000
Server disk	10.000	10.000	10.000	10.000

What is the limiting resource?

Insecure comm.: server disk  $1 / D_{max} = 1 / 0.01 = 100\ req/sec$   
 all others: server CPU  $1 / D_{max512} = 1 / 0.0139 = 71.94\ req/sec$   
 $1 / D_{max768} = 1 / 0.0274 = 36.50\ req/sec$   
 $1 / D_{max1024} = 1 / 0.0517 = 19.34\ req/sec$

251

## Ex. 38: Impact of TLS on Server Throughput



252

:

### Ex. 39: Maximum Secure Load

Under the previous scenario, what is the maximum fraction of secure requests that can be handled by the present CPU using 1024 bit keys and RC4 and MD5?

The maximum secure load is the load when the service demand for the server CPU equals the service demand for the disk, i.e. 10 msec

$$0.002x + (1-x)(0.0517) = 0.010$$

$$0.0517 - 0.010 = (0.0517 - 0.002)x$$

$$x = 0.0417 / 0.0497 = 0.839$$

Even with the simplest algorithms, the present CPU could not handle more than 16.1% of the workload as secure requests.

..... 253

:

### Ex. 40: Effect of Better Cryptographic Algorithms

A rash of recent attacks has management worried and it wants to explore the effects of using better cryptography.

What is the maximum throughput using other encryption algorithms?

Insecure	100.0
RC4-MD5-512	71.9
RC4-MD5-768	36.4
RC4-MD5-1024	19.3
3DES-SHA1-512	45.4
3DES-SHA1-768	28.1
3DES-SHA1-1024	16.7

..... 254

:

### Cryptographic Accelerator Cards

- The most expensive part of PK cryptography is the computation of integer modular exponentiation on large numbers
- Cryptographic accelerator cards implement various modular exponentiation algorithms as well as random number generation
- Speed-up factors of 50 are common
- Recently it was claimed that a software solution was able to beat the cryptographic accelerator cards (correctness proofs pending)

..... 255

:

### Ex. 41: Effect of Cryptographic Accelerators

Assuming a cryptographic accelerator card is used, what is the maximum throughput?

The new service demands with the accelerator card for RC4 and MD5 are

Client:

$$0.01239/50 + (16\ 384)(8) / (140\ 000\ 000) + (16\ 384)(8) / (180\ 000\ 000) = 0.00191\ \text{sec}$$

Server:

$$0.002 + 0.04803 / 50 + (16\ 384)(8) / (140\ 000\ 000) + (16\ 384)(8) / (180\ 000\ 000) = 0.004625\ \text{sec}$$

With an accelerator card the server disk becomes the bottleneck again!

..... 256



:

## Firewalls

- Firewalls are devices or software products that allow the network manager to restrict access to components on the network.
- The most common types of firewalls are
  - frame-filtering firewalls: filters on the type of (LAN) frame (e.g. Ethernet/802.3, token ring/802.5, FDDI)
  - packet filter firewalls: may be a switch with packet filtering capabilities or a dedicated machine. Degrades rapidly when conditional filtering is performed
  - circuit gateway firewalls: perform authentication and validation at session set-up and logging of origin and destination on the network

..... 257

:

## Firewalls (cont.)

- Application-level firewalls (proxy-level firewalls):
  - provide protection at the application level and are special purpose (as opposed to general purpose packet filtering)
  - custom programs are added for each protected application
  - custom programs act as proxies for the real applications (proxy servers) and readdress traffic
  - two types of proxy servers
    - incoming connection is intercepted by proxy and new connection from proxy to destination is created ==> outside connection can't touch destination and full filtering is possible
    - proxy server appears as only destination for all applications on a trusted network from an untrusted network (internal network is completely hidden, internal network can use unregistered address ranges for IP users to access external networks expecting valid address ranges)

..... 258

:

## Firewalls (cont.)

- Statefull firewalls:
  - statefull inspection firewalls combine features of packet filtering and application layer processing by examining the content of each incoming packet
  - statefull inspection firewalls need to examine the transaction condition between two interoperating applications
  - firewall must understand enough of the protocol details to identify specific conditions and what is expected next
  - used to defeat technical attacks, such as IP address spoofing, session hijacking, piggyback session acquisition, etc.

..... 259

:

## A simple firewall performance measurement

- Experiment comparing packet filtering bridge and ftp-gw from TIS firewall toolkit
- Results reported by Andrew Molitor (insert URL)
- Experiment setup eliminated external noise and extraneous service request traffic (e.g. hosts named by IP addresses, no name services, etc.)
- Experiment used on purpose slow hardware to bring performance into a range where differences can be observed

..... 260

:

## Simple firewall measurement (cont.)

- Baseline was established (null firewall) and measurements were performed on 30 sessions

	Null firewall	Packet Filter	ftp-gw
Ping (8 bytes)	3.1 msec	5.3 msec	N/A
Ping (1000 bytes)	5.3 msec	11.8 msec	N/A
Sequential FTP sessions	41.3 sec	41.3 sec	77.8 sec
Parallel FTP sessions	13.2 sec	11.4 sec	17.9 sec
Sequential FTP bulk transfer	29.8 sec	57.5 sec	112.8 sec
Parallel FTP bulk transfer	20.6 sec	44.2 sec	88.5 sec

Throughput null firewall =  $30 / 13.2 = 2.3$  sessions / sec  
 Throughput ftp-gw =  $30 / 17.9 = 1.68$  sessions / sec

FTP session latency null firewall =  $41.3 / 30 = 1.38$  sec  
 FTP session latency ftp-gw =  $77.8 / 30 = 2.59$  sec

Both firewalls interfere substantially with bulk transfer rates, but proxy much more

..... 261

:

## Performance Comparison of Products

- Performance of firewalls is becoming an issue with high-speed connections available
- <http://www.nwfusion.com/reviews/2001/0312rev.html>
- Sixteen products were submitted for testing: Cisco's PIX 525; Check Point's Firewall-1; Computer Associates' eTrust; CyberGuard's KnightStar; Enternet's Enternet Firewall; Lucent's Brick; NetScreen's NetScreen-100; Network-1's CyberwallPlus; Network Associates' WebShield; Novell's BorderManager; Nokia's IP650; Secure Computing's SideWinder; SonicWall's SonicWall Pro VX; Symantec's Raptor; TopLayer's AppSwitch 3500; and WatchGuard's Firebox II.

..... 262

:

## Performance Comparison

- Three test suites were used:
  - raw throughput, i.e. traffic conditions a firewall might encounter while being used at the core of an enterprise network or in an Internet environment where activities such as file-sharing must pass through it
  - massive connections, i.e. maintain many TCP connections without losing packets or having to retransmit
  - real stressful connections, i.e. many users surfing simultaneously

..... 263

:

## Performance Comparison - Raw Throughput

- Two extremes: long packets (1400 bytes) and short packets (64 bytes)
- Short packets stress the system because packet processing overhead dominates
- Long packets are better behaved and yield higher throughput even though they put higher load on the network

..... 264

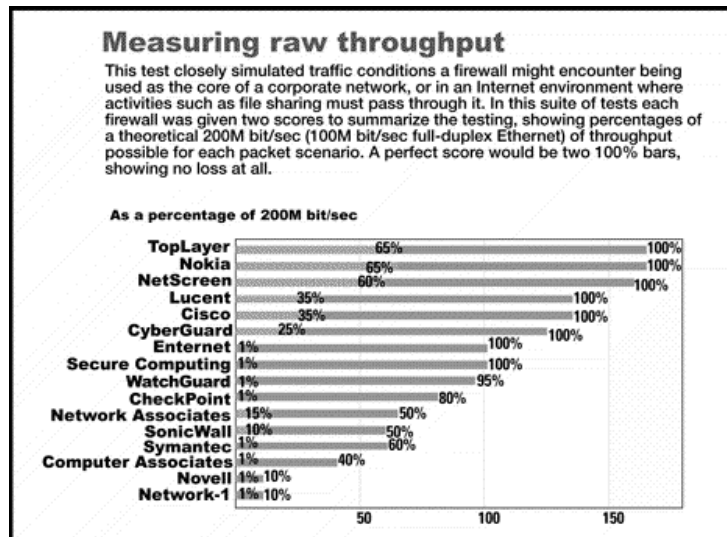
## Summary of Raw Performance Results

Theoretical limit: 200 Mbps (100 Mbps full duplex Ethernet)

	Long Packets	Short Packets
Top Layer	200	130
Nokia	200	130
Netscreen	200	120
Lucent	200	70
Cisco	200	70
Cyberguard	200	50
Enternet	200	2
Secure Computing	200	2
Watchguard	190	2
Checkpoint	160	2
Network Associates	100	30
Sonic Wall	100	20
Symantec	120	2
Computer Associates	80	2
Novell	20	2
Network 1	20	2

265

## Raw Performance



266

## Performance Comparison - Raw Throughput

- Nine firewalls handled the long-packet test without dropping significant amounts of traffic.
- Only three products - TopLayer, Nokia and NetScreen - were able to handle more than 50% load (100M bit/sec throughput) in the small-packet test.
- Finding a firewall to handle well-designed applications is not a difficult task. However, when it comes to the worst-case traffic pattern, you have to be a lot pickier.

267

## Massive Connections

- Maintaining many TCP connections is important in data centers where a single firewall may be protecting multiple Web servers
- Packet loss or delay during connection establishment greatly degrades the perceived response time (important in Web applications in which a page can have tens of elements, each requiring a separate connection to the Web server)
- Resources consumed per connection depend on type of firewall: packet filters consume very little, stateful inspection firewalls/proxy servers must interact with OS to set up a matched set of TCP connections and maintain state tables

268

:

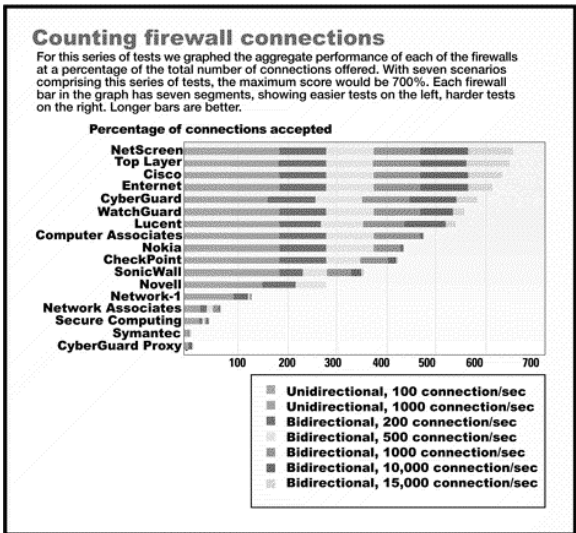
### Experimental setup/Discussion - Massive Connections

- Varying number of connections: 25 000 to 120 000
- Varying rates: 100 conn./sec to 15 000 conn./sec  
(100 x 60 x 60 x 24 x 30 = 259 200 000 hits/month)
- Most products have no problem at 100 conn/sec (except Symantec and Secure Computing)
- Products offering both packet filtering and full proxy mode (e.g. CyberGuard) perform quite differently in each mode (10 000 + conn/sec in packet filter mode and 100 conn./sec in full proxy mode)
- Cisco, CyberGuard, Enternet, NatScreen could handle 10 000 + conn./sec

..... 269

:

### Results: Massive Connections



..... 270

:

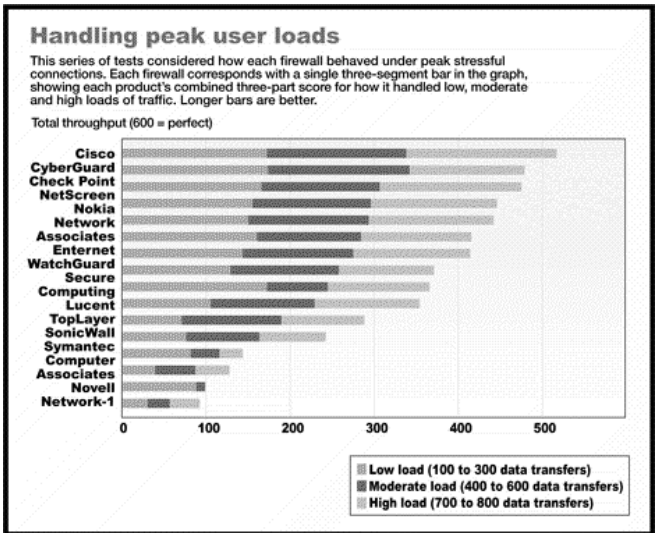
### Peak User Load

- Peak user load occurs when many users are surfing the Web
- Experimental setup:
  - 20 test systems connected by wire-speed switches on each side of the firewall
  - contention and retransmission were produced by funneling multiple systems into a single port on each side of the firewall
  - loads were measured at 100 to 300, 400 to 600, and 700 to 800 simultaneous data transfers

..... 271

:

### Peak User Load



..... 272

:

### Discussion of Results - Peak User Load

- Top end systems (e.g. Cisco, CyberGuard, NetScreen and Nokia) did well across the board and could handle the highest loads
- Secure Computing and Novell did quite well at low loads but froze at high loads
- To protect a building fed by a DS-3 connection with 45 Mbps and approx. 1000 users any firewall capable of handling well 200 simultaneous data transfers will do the job
- Must be more careful for higher speed feeds

..... 273

:

### Firewall Performance Results - Latency

Firewall	T-put (Mbps)	Response T (sec)
Crossover cable	155	0.0220
AXENT Raptor telnet px.	70	0.0480
AXENT Raptor gen. px.	50	0.0670
Check Point Fire Wall 1	125	0.0287
Cisco PIX Firewall	150	0.0234
CyberGuard HTTP	70	0.0500
NetGuard Guardian	45	0.0700
NetScreen 100	105	0.0335
Secure Computing	75	0.0465

Note: minor penalty for Network Address Translation (NAT) except NetGuard with factor 4 !!

..... 274

:

### Quantitative Analysis of Payment Services

- Using authentication protocols such as SSL or TSL takes care of mutual authentication between customer and merchant and protects information sent over the Internet.
- Authentication protocols do not guarantee:
  - the proper storage of credit card information on the merchant's system
  - protection from misuse of credit card info by merchant
  - legitimacy of card holders and merchants
- Payment protocols, e.g. Secure Electronic Transactions (SET) take care of these issues

..... 275

:

### Payment by Credit Card - Authorization

- Clerk swipes card through POS terminal and enters the amount of the transaction
- Authorization request is started and sent to Card Processing Center
- Authorization request contains issuer bank ID, credit card number, transaction amount.
- Card Processing Center contacts issuing bank via Bank Interchange Network and forwards authorization request.
- Bank checks available credit and if greater than purchase amount, reduces it and sends authorization code back to CPC and CPC to merchant

..... 276

:

### Payment by Credit Card - Settlement

- Debit during authorization phase is temporary
- POS terminal generates a capture record for every authorization.
- Capture records are collected and put in a batch settlement file sorted by credit card brand but covering possibly many different issuing banks
- Credit Card Processing Service gets batch file and sorts it by issuing bank and forwards it
- Debit becomes permanent and issuing bank pays the specified amount to merchant via wire transfer to the merchant's acquirer bank

..... 277

:

### SET - Secure Electronic Transactions

- SET was developed by SET Consortium (Visa, MasterCard, GTE, IBM, Microsoft, VeriSign, etc.)
- SET provides confidentiality of order and payment information:
  - payment info cannot be viewed by merchant
  - order information cannot be seen by payment processing entities
- Through SET
  - merchants are authenticated as being valid recipients of credit card payments
  - credit card holders are authenticated as legitimate users

..... 278

:

### SET - Mode of Operation

- Instead of physical credit card, SET uses e-wallet
  - helper application or plug-in to browser
  - stores all the information about the credit cards the user possesses and SET digital certificates for these cards
  - digital certificates replace physical signature and picture
  - cardholder, merchant and payment gateway have SET certificates used for mutual authentication
- The entities involved in a SET transaction are:
  - customer's e-wallet
  - merchant's POS software
  - acquirer bank's payment gateway (performs function of Card Processing Service)

..... 279

:

### SET Cryptography

- SET uses
  - SHA-1 for message digest
  - 1024 bit RSA for encrypting message digest with sender's private key
  - DES for symmetric encryption
- Certification authorities used by SET are organized in a hierarchy (all certificates must be verified):
  - root CA
  - brand CA (e.g. MasterCard, Visa, etc.)
  - geopolitical CA
  - cardholder CA

..... 280

:

### Digital Envelopes

- Digital envelope is a DES-encrypted message along with the RSA-encrypted key used for DES-encryption
- Creating a digital envelope:
  - sender generates random key used for DES-encryption
  - key is used to encrypt message using DES
  - DES-key is encrypted using RSA w. receiver's public key
  - digital envelope is formed by concatenating DES-encrypted message with RSA-encrypted DES-key
  - envelope is transmitted to receiver

..... 281

:

### Digital Envelopes (cont.)

- Opening a digital envelope
  - only valid receiver can open envelope since he needs the private key to decrypt the DES-key
  - with the DES-key the DES-encrypted message can be decrypted
- Creating a digital envelope requires
  - 1 DES-encryption
  - 1 RSA public key operation
- Opening a digital envelope requires
  - 1 DES-decryption
  - 1 RSA private key operation

..... 282

:

### Double Signatures

- SET sends order and payment information from the e-wallet to the merchant's POS software
- SET shields payment information from merchant but passes it through to the payment gateway
- This is accomplished by using a doubly signed message
- A doubly signed message contains 2 digital envelopes (one for the merchant and one for the payment center), the signature for each message plus a double signature

..... 283

:

### Double Signatures (cont.)

- To produce a doubly signed message
  - e-wallet produces digital envelope with order information for merchant using merchant's public key
  - e-wallet produces digital envelope with payment info for payment gateway using PG's public key
  - for each message a message digest is produced using SHA-1
  - both message digests are concatenated and a new message digest for the concatenated MDs is generated using SHA-1
  - combined message digest is encrypted using RSA with e-wallet's private key (double signature)
  - each individual MDs is encrypted using RSA with e-wallet's private key
  - doubly signed message is composed by concatenating DE m1, sig. m1, DE m2, sig. m2, double sig.

..... 284

:

### Verification of signed messages

- Merchant can only open DE intended for him
  - merchant opens DE with his private key and recovers original message
  - SHA-1 is applied to recovered message and a message digest is produced by merchant's POS software
  - the MD2 (intended for payment gateway) is recovered using RSA and e-wallet's public key
  - both message digests are concatenated and a new digest is generated using SHA-1
  - the original double signature is decrypted using RSA and e-wallet's public key
  - the decrypted double signature is equal to new MD iff doubly signed message came from customer's e-wallet, m1 was addressed to merchant, m2 was signed by customer

..... 285

:

### Operations involved in double signing

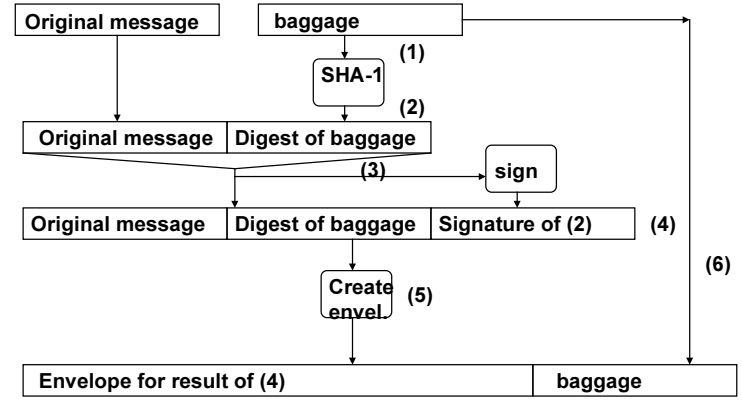
- Double signing involves:
  - 2 create digital envelope ops. (1DES + 1 RSA public)
  - 3 SHA-1 operations
  - 3 RSA private key operations
- Processing a doubly signed envelope involves
  - 1 open envelope op. (1 DES + 1 RSA private)
  - 2 SHA-1 operations
  - 2 RSA public key operations

..... 286

:

### Simple Encapsulation w. Signature and Baggage

- Baggage is an extra piece of information that is used in reply messages for added security



..... 287

:

### Simple Encapsulation w. Signature and Baggage

- 1) Create digest of baggage using SHA-1
  - 2) Concatenate original message with digest of baggage
  - 3) Sign concatenation with sender's private key
  - 4) Concatenate results of steps 2 and 3
  - 5) Create a digital envelope for this concatenation
  - 6) Concatenate result of (5) with baggage = encapsulated signed message with baggage
- Receiver of EncB message can open envelope with his private key and verify signature w. sender's public key. From open envelope receiver gets digest and regenerates it from the baggage

..... 288



:

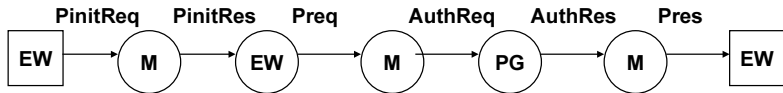
### Operations involved in generating EncB

- Generating EncB requires
  - 1 SHA-1
  - 1 sign operation (1 private RSA, 1 SHA-1)
  - 1 create envelope (1 public RSA, 1 DES)
- Receiving an EncB message requires
  - open envelope = 1 private RSA, 1 DES
  - signature verification = 1 SHA-1
  - checking integrity of baggage = 1 SHA-1

..... 289

:

### SET Message Flow

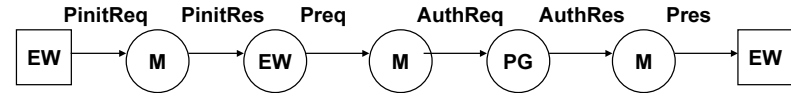


- E-wallet sends Purchase Initiate Request to merchant server (credit card name, bank id#, cardholder challenge string, list of digital certificates stored in customer's e-wallet (4) incl. Public keys)
- Merchant's POS replies w. Purchase Initiate Response, a message signed w. merchant's private key. It includes unique Tx-id, e-wallet challenge string, chain of digital certificates for payment gateway

..... 290

:

### SET Message Flow (cont.)

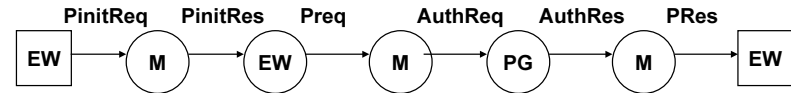


- EW sends doubly signed Purchase Request to merchant. Message consists of order instructions and payment instructions.
- Merchant must obtain authorization from payment gateway. (EncB with baggage being payment information encrypted with gateway's public key)
- Gateway checks available credit, returns authorization code (Authorization Response) w. explanation as an EncB message (expl. = baggage)

..... 291

:

### SET Message Flow (cont.)



- Purchase Response sent by merchant to EW after merchant received AuthRes message. PRes contains completion code and other data intended for EW signed by merchant with EW's public key

..... 292

## SET Performance

- SET performance is largely determined by crypto
- An Overview of crypto operations per SET op.

SET Operation	RSA private	RSA public	SHA-1	DES
Sign	1	-	1	-
Verify signature	-	1	1	-
Verify certificate	-	4	4	-
Create envelope	-	1	-	1
Open envelope	1	-	-	1
Double sign	3	2	3	2
Process double signature	1	2	2	1
Generate EncB message	1	1	2	1
Receive EncB message	1	1	2	1

293

## SET Performance (cont.)

- Table of operations performed by E-wallet, merchant and payment gateway

SET operations	E-wallet	Merchant	Payment Gateway	
Sign	1	2	-	-
Verify signature	2	1	-	-
Double sign	1	-	-	-
Process double signature	-	1	-	-
Generate EcbB message	-	1	1	1
Receive EncB message	-	1	1	1
<b>Total crypto ops</b>	<b>public RSA</b>	<b>private RSA</b>	<b>SHA-1</b>	<b>DES</b>
E-Wallet	4	4	6	2
Merchant's POS	5	5	9	3
Payment gateway	2	2	4	2

294

## Ex. 42: Effect of SET vs. TLS on Server

Management wants to compare SET with TLS

Assumptions: POS software runs on same machine as Web Server. Clients send requests from high speed LAN to the server.

Service demands for TLS and SET:

Resource	TLS	SET
E-Wallet	14.054	213.0 = (4)(47.93) + (4)(5.2) + (6)(0.00788) + (2)(0.2048)
Network	1.737	1.737
Server CPU	51.694	268.3 = 2 + (5)(47.93) + (5)(5.2) + (9)(0.00788) + (3)(0.2048)
Server Disk	10.0	10.0
Payment Gateway	-	106.7 = (2)(47.93) + (2)(5.2) + (4)(0.00788) + (2)(0.2048)
Financial Network	-	80.0

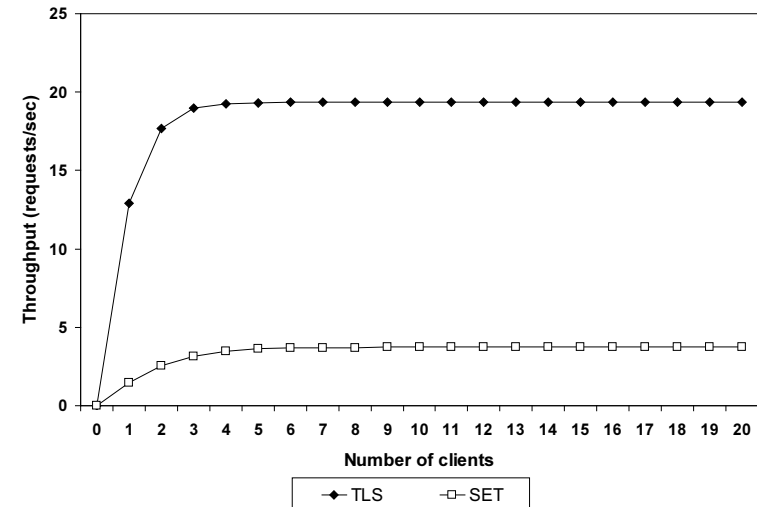
Bottleneck: CPU

MaxThroughputTLS =  $1/0.051694 = 19.345$  requests/sec

Max ThroughputSET =  $1 / 0.2683 = 3.727$  requests / sec

295

## Throughput SET and TLS vs. Number of Clients



296

### Ex. 43: Effect of Cryptographic Accelerator on SET

If a cryptographic accelerator at merchant is used (factor 50 improvement):

Resource	SET
E-Wallet	213.0
Network	1.737
Server CPU	2 + 266.3/50 = 7.33
Server Disk	10.0
Payment Gateway	106.7
Financial Network	80.0

Given a fixed set of clients on high speed LAN, E-Wallet becomes the bottleneck

Throughput =  $1 / 0.213 = 4.69$  requests/sec

If large number of clients exist, E-Wallet would not be the limiting factor, but the Payment Gateway becomes the bottleneck

Throughput =  $1 / 0.1067 = 9.37$  requests/sec

If Payment Gateway also uses cryptographic accelerator, the financial network limits

Throughput =  $1 / 80 = 12.5$  requests / sec

297

### Ex. 44: Effect of Moore's Law on SET

According to Moore's Law, compute power doubles every 18 months.

Year	Proc. Speedup	Secure Trans (tps)
1	1.00	14
2	1.60	19
3	2.56	23
4	4.01	26
5	6.41	29

Elliptic Curve Cryptography (ECC) may provide factor 4 to 10 improvement

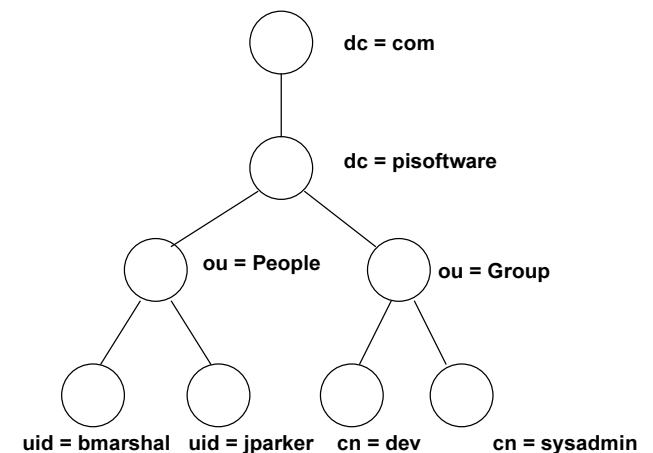
298

### LDAP

- LDAP (Lightweight Directory Access Protocol) is often used (among other things) for
  - storage of personnel information
  - role based access control
  - authentication
  - routing policy management
- LDAP evolved from the X.500 protocol
- LDAP servers store attribute-based data for reading access mostly (no transactions, no rollback) in hierarchical structures

299

### LDAP Hierarchy



300

:

### LDAP Performance

- LDAP can use different types of persistent stores
- Most of the time, the entries can be cached ==> the limiting resource is usually the CPU
- When entries can't be fully cached, the processing time deteriorates rapidly (~ 87% reported)
- Performance figures for LDAP servers are not readily available, existing figures are older (97/98)
- Performance varies wildly from 183 requests/sec for NSDS3 to 0.8 requests/sec for Novell NDS
- Data reported for OpenLDAP are latency of 8 msec at loads of 105 requests/sec

. . . . . 301

:

### Load Characterization

- Want to characterize the workload for e-commerce applications
- Will look at
  - workload characterization of Web Traffic
  - workload characterization of intranets
  - characterization of customer behavior
  - characterization of user or customer behavior from HTTP logs

. . . . . 302

:

### Load Characterization of Web Traffic

- Will analyze first information retrieval Web servers
- File popularity follows a Zipf distribution
- In a Zipf distribution files are ranked. The number of accesses P to a files is inversely proportional to the rank r, i.e.

$$P = k/r$$

where k is a constant

- The rank of the most popular paper is 1, the second most popular paper has rank 2, etc.

. . . . . 303

:

### Ex. 45: Zipf Distribution of Web Requests

The HTTP log of a Web site shows 1800 requests for files during a 5 minute period. Requests are directed at 12 distinct files.

Assuming Zipf's Law, what is the estimated number of accesses to each file?

Files are ranked 1 through 12. Most popular is 1, ...

The number of accesses to each file is k/r

$$1800 = k ( 1/1 + 1/2 + \dots + 1/12 ) = (k)(3.1032)$$

$$k = 1800 / 3.1032 = 580.05$$

File	Accesses
1	580
2	290
3	196.7
4	145
...	
12	48

. . . . . 304

:

## Heavy-Tailed Distributions

- Empirical studies have shown that many distributions of Web-related traffic are heavy-tailed
- A heavy-tailed distribution for a random variable  $X$  is a distribution where
  - the probability that  $X > x$  decreases with  $x^{-\alpha}$  for large values of  $x$  and for  $0 < \alpha < 2$
- In a heavy-tailed distribution the probability that a large value occurs is small but non-negligible

..... 305

:

## Heavy-Tailed Distributions on the Web

- On the Web, while most retrieved files are small, there is a non-negligible probability of large files (images, video-clips)
- Features that follow a heavy-tailed distribution in Web-traffic are
  - size of files requested from Web servers
  - size of files requested from the entire Web
  - number of pages requested per site
  - reading time per page

..... 306

:

## Pareto Distribution

- The Pareto distribution is a good example of a heavy-tailed distribution
- The cumulative distribution function (CDF) is given by

$$F(x) = P[X \leq x] = 1 - (k/x)^\alpha \quad \alpha, k > 0$$

and the tail of the distribution is given by

$$P[X > x] = (k/x)^\alpha$$

..... 307

:

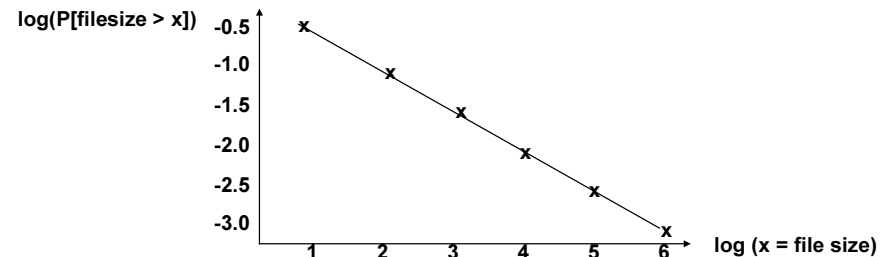
## Ex. 46: File Size Distribution (Pareto)

The HTTP log for a website was analyzed to estimate the size of the files retrieved from the site. Determine whether the distribution follows a Pareto distribution or not.

If a distribution follows the Pareto distribution the log-log plot of the tail of the distribution should be a straight line

$$\log P[X > x] = -\alpha \log x + \alpha \log k$$

This means that the logarithm of the tail of the distribution decreases linearly with the logarithm of the file size with slope  $-\alpha$ .



..... 308

:

### Observed characteristics of HTTP traffic

- HTTP traffic has been shown to be self-similar, i.e. same patterns of burstiness across several time scales ranging from microseconds to minutes.
- 99% of WWW queries (to search engines) do not use any Boolean or other advanced operators
- In cable-modem environments, 40% of total size of unique files retrieved is due to a few very large files ==> as bandwidth increases, users are more willing to download audio, video ==> higher stress on server resources

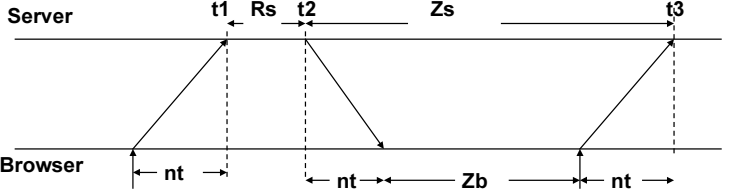
:

### Characterizing the dynamic user behavior

- We used before the Customer Behavior Model Graphs (CBMG) to show how a user accesses the various functions of an e-commerce site, i.e. we showed navigational patterns of user behavior
- Navigational patterns have 2 components:
  - transitional: how a user moves from state to state, given by matrix of transition probabilities
  - temporal: shows the time it takes a user to move from one state to another, i.e. server-perceived think time
- (Server-perceived) think time is the time between the server finishing a request and the next request submitted by the user

:

### Browser vs. Server Think Times



$$Z_s = 2 nt + Z_b$$

- Server-side think time is 2 times the network time + browser-side think time
- A think time can be associated with each transition

:

### Extended CBMG

- The extended CBMG can be expressed as (P,Z)
  - P = [p<sub>ij</sub>] is an nxn matrix of transition probabilities
  - Z = [z<sub>ij</sub>] is an nxn matrix of average think times between the states of the CBMG
- For any state j of the CBMG
 
$$V_j = \sum_{k=1}^{n-1} V_k p_{k,j}$$
- The average number of visits to each state can be obtained by solving this system of equations knowing that V<sub>1</sub> = V<sub>n</sub> = 1 (entry state and exit state are only visited once per session)

⋮

### Metrics derived from CBMG

- Average session length

$$\text{AverageSessionLength} = \sum_{j=2}^{n-1} V_j$$

Calculation AverageSessionLength

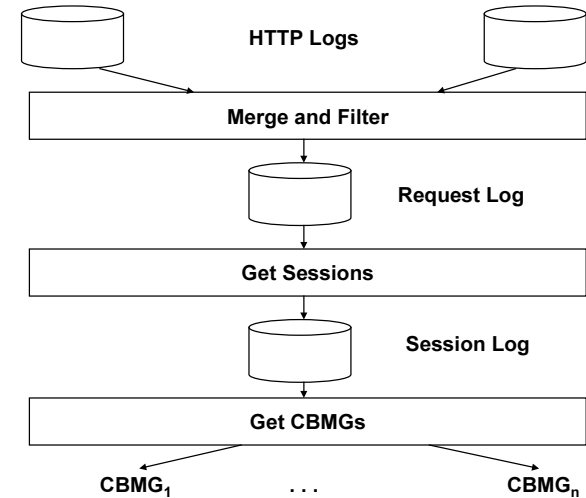
⋮

### From HTTP Logs to CBMGs

- Customer sessions can be derived from HTTP logs
- We want to
  - obtain the CBMGs that characterize customer interactions
  - group CBMGs that are derived from similar sessions
  - represent each group by a CBMG
- Goal: to characterize the workload with a manageable number of CBMGs

⋮

### From HTTP Logs to CBMGs



⋮

### Merging the HTTP logs

- Merge and filter logs from the HTTP servers
  - discard irrelevant entries (image requests, errors, etc.)
  - merge logs into single log based on timestamp (use clock synchronization services)
- Output is request log L consisting of tuples (u,r,t,x)
  - UserID (u - identified via cookies, authentication mechanisms, etc.)
  - RequestType (r - e.g. GET on home page, request to execute search, selection of result of search, CGI, etc.)
  - RequestTime (t - timestamp of arrival at site)
  - ExecTime (x - execution time, usually not provided by HTTP log, but can be obtained by configuring server)

:

## Extracting the Sessions

- Second step takes as input the request log L and generates the session log S
- Structure of the session log S:
  - each entry in the session log consists of a tuple  $(C_k, W_k)$
  - $C_k = [c_{ij}]$  is an nxn matrix of transition counts between states i and j of the CBMG for one session
  - $W_k = [w_{ij}]$  is an nxn matrix of accumulated think times between states i and j of the CBMG for one session
  - Example: between states s and t there were 3 transitions and the think times were 20 sec, 45 sec, and 38 sec, respectively.  
 $\implies c_{s,t} = 3 \quad w_{s,t} = 20 + 45 + 38 = 103 \text{ sec}$

..... 317

:

## Algorithm GetSessions

- 1) Sort the request log L by UserID and RequestTime. Result is sorted log containing subsequences for each user.
- 2) Break up all the requests of a user into sessions. A single user may have visited the site repeatedly. By giving a certain time threshold, requests are separated into different sessions
- 3) For each session repeat
  - initialize matrix  $C[i,j]$  and  $W[i,j]$  to zero
  - For  $k = 2$  to Q do
    - Begin
      - $C[r_{k-1}, r_k] \leftarrow C[r_{k-1}, r_k] + 1;$
      - $W[r_{k-1}, r_k] \leftarrow W[r_{k-1}, r_k] + (t_k - t_{k-1} - x_{k-1});$
    - End;
  - $C[r_Q, n] \leftarrow 1;$  {transition to the Exit state}

..... 318

:

## Precautions when using HTTP logs

- msec precision may not be exact enough ==> Apache server can be configured to give a more precise timestamp
- Clean log from crawler activity (record browser id in the log, may have to use extended log format)

..... 319

:

## GetCBMG Algorithm

Goal: to generate a synthetic workload consisting of a few typical CBMGs

Input: session log S

Output: set of CBMGs given each by  $(P,Z)$  and the arrival rate  $\lambda$

Approach: k-means clustering algorithm

..... 320



•

## k-means Clustering Algorithm

Select  $k$  points in the space of points. The selected points act as initial estimate of the centroids of each cluster

Points in this case are sessions characterized by  $(C,W)$ , where  $C = [c_{i,j}]$  is an  $n \times n$  matrix of transition counts between states  $i$  and  $j$ , and  $W = [w_{i,j}]$  is the  $n \times n$  matrix of accumulated think times between states  $i$  and  $j$ .

All the remaining points are allocated to the cluster with the nearest centroid ==>

Must define some metric for distance between points (sessions):

Assume that session log  $S$  consists of  $M$  "points"  $X_m = (C_m, W_m)$  with  $m = 1, \dots, M$

The distance between two points (sessions) is based on transition count only

$$d_{X_a, X_b} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (C_a[i,j] - C_b[i,j])^2}$$

• • • • • 321

•

## k-means Clustering Algorithm (cont.)

At any point in time of the execution of the  $k$ -means algorithm there are  $k$  centroids.

Algorithm must keep track of number of points  $s(k)$  represented by the  $k^{\text{th}}$  centroid.

Each time a new point is added to a centroid, the matrices  $(C,W)$  of the centroid must be recomputed.

Add  $X_m = (C_m, W_m)$  to the  $k^{\text{th}}$  centroid represented by  $(C,W)$ . The new centroid will have the new matrices  $(C',W')$

$$C' [i,j] = \frac{s(k) C[i,j] + C_m[i,j]}{s(k) + 1}$$

$$W' [i,j] = \frac{s(k) W[i,j] + W_m[i,j]}{s(k) + 1}$$

• • • • • 322

•

## k-means Clustering Algorithm (cont.)

Once all the clusters have been obtained, the matrices  $P$  and  $Z$  for each cluster can be derived

$$p_{i,j} = C[i,j] / \sum_{k=1}^n C[i,k]$$

$$z_{i,j} = W[i,j] / C[i,j]$$

The arrival rate  $\lambda_{k^s}$  of sessions represented by the CBMG of cluster  $k$  is

$$\lambda_{k^s} = s(k) / T$$

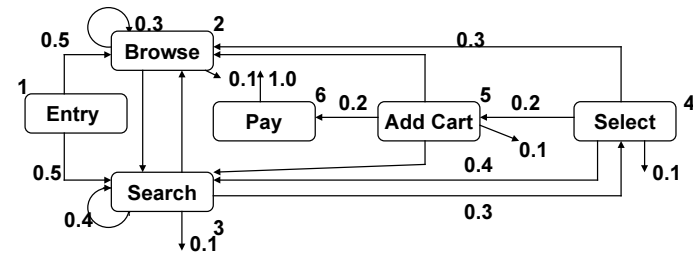
$T$  is the time interval during which the request log  $L$  was obtained

• • • • • 323

•

## Ex. 47 : Derivation of CBMGs

Consider a book store site with the static CBMG given below.



The HTTP log with 340 000 requests was analyzed and 20 000 sessions were generated.

After running the  $k$ -means clustering algorithm on the 20 000 sessions with  $k = 6$  the following table was obtained:

• • • • • 324

## Ex. 47 : Derivation of CBMGs (cont.)

Cluster	1	2	3	4	5	6
% of sessions	44.28	28.0	10.6	9.29	6.2	1.5
BV Ratio (%)	5.7	4.5	3.7	4.0	3.5	2.0
Session Length	5.6	15	27	28	50	81
Va (%)	11	15	21	20	32	50
Vb + Vs (%)	3.6	11.4	20	23	39	70

- 1) Cluster 1 comprises 44.28% of all sessions
- 2) BV is the buy/visit ratio representing the customers who end up buying during a visit to the store
- 3) Session length is the average number of shopper operations requested per visit
- 4) Va is the ratio of AddToShoppingCart / Visit ratio
- 5) Vb + Vs is the number of browse and search operations per session in each cluster

325

## Clustering alternatives

- There exist different approaches to clustering
  - clustering on the whole log
  - partition first by some criterion (e.g. visits resulting in sales), then cluster each partition separately
- Different kind of information can be extracted from the various clustering approaches
- An important issue is: How many clusters characterize the workload accurately?

326

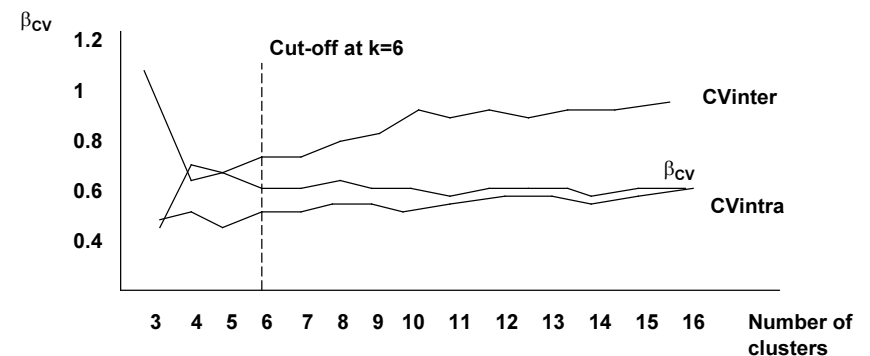
## Selecting the number of clusters

- To determine the best number of clusters, analyze the coefficient of variation CV
- CV is the ratio between the average and the standard deviation
- Goal:
  - to minimize the intracluster CV (the distance between points of a cluster and the cluster centroid)
  - to maximize intercluster CV
  - to achieve the above with a *small* number of clusters  
 ==> find the smallest number of clusters where the ratio  $CV_{intra}/CV_{inter}$  does not change significantly

327

## EX. 48: Variation of $CV_{intra} / CV_{inter}$

(Figure is rough rendering of data from Menasce et al. ACM Conf. On E-commerce 99)



$$\beta_{CV} = CV_{intra} / CV_{inter}$$

328

## Extracting Customer Visit Models from HTTP Logs

- The CVM is a coarser characterization of the workload
- CVM indicates per session the number of visits to each state
- As with the CBMG, the individual sessions should be clustered
- The distance metric when deriving CVMs is the distance between two visit ratio vectors

$$d_{V_a, V_b} = \sqrt{\sum_{i=2}^{n-1} (V_i^a - V_i^b)^2}$$

329

## Algorithm GetCVMSessions

- 1) Sort the request log L by UserID and RequestTime. Result is sorted log containing subsequences for each user.
- 2) Break up all the requests of a user into sessions. A single user may have visited the site repeatedly. By giving a certain time threshold, requests are separated into different sessions
- 3) For each session repeat
  - initialize  $V_i$  to zero for all  $i = 2, \dots, n-1$
  - $V_1, V_n \leftarrow 1$  (initial and final state are visited once)
  - For  $k = 1$  to Q do

$V_{rk} \leftarrow V_{rk} + 1;$

330

## Ex. 49: CVM

Session	Vbrowse	Vsearch	Vadd	Vselect	Vpay
1	4	10	2	4	1
2	15	20	1	18	0
3	5	8	3	5	1
4	16	18	3	16	1
5	10	8	0	5	0
6	3	10	2	8	1
7	5	11	3	8	1
8	10	15	0	12	0
9	8	6	3	4	1
10	7	10	1	8	1
11	10	20	0	15	0
12	5	4	1	2	1

Cluster	VB	VS	VA	VS	VP
1	5.875	8.375	1.875	5.5	0.875
2	12.75	18.25	1.0	15.25	.25
3	4.75	10.25	2.0	7.0	1.0
4	12.75	18.25	1.0	15.25	0.25
5	7.0	6.5	1.75	4.0	0.75

331

## E-Commerce Benchmarks

- Benchmarks serve the purpose of providing a standardized comparison across platforms
- Various benchmarks are emerging for e-commerce
  - TPC-W (Transaction Processing Council)
    - publication date of specification: Feb. 2000
    - B2C Web commerce scenario
    - intends to stress the whole system, heavy DB load
  - Ecper (Sun Microsystems)
    - publication date of specification: May 2001
    - manufacturing, supply chain management, order/inventory
    - intends to stress the EJB-based application server, tries to deemphasize DBMS performance

332

:

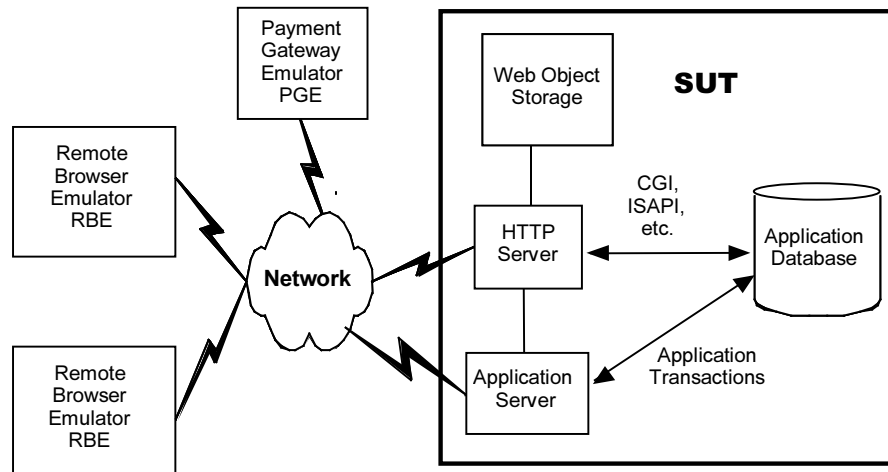
## TPC-W

- Benchmark based on amazon.com business model (although spec. claims that no particular industry is addressed :-)
- Benchmark includes mix of
  - static content (mainly images)
  - dynamic content
- Dynamic content derived from DB distinguishes TPC-W from other e-commerce benchmarks
- TPC-W requires secure transactions (SSL V3) for a subset of the interactions

..... 333

:

## Diagram of whole system



..... 334

:

## Metrics of TPC-W

- The metrics used in reporting TPC-W results are:
  - performance (Web Interactions per Second - WIPS)
  - price/performance (\$ / WIPS)
  - date of system availability
- Performance is reported with three submetrics derived from different traffic mixes
  - Web Interactions per Second (WIPS) (primary metric)
  - Heavy browse traffic (WIPSB) (uses different load mix)
  - Heavy ordering (WIPSO)
- Loads for generating the 3 metrics were produced analyzing real e-commerce sites

..... 335

:

## Scale of TPC-W

- TPC-W defines 5 scales (1 000 to 10 000 000)
- Benchmarks for only 2 scales have been produced
  - 10 000 items (small scale)
  - 100 000 items (“large” scale)
- A larger item count also implies a larger customer base
- The scale has an impact on the DB size and on the traffic (and on cost of platform)
- Benchmark sponsors must decide whether they want to demonstrate highest throughput (usually at high cost) or lower throughput at affordable prices

..... 336

:

### TPC-W Application code

- Application code for TPC-W may either be
  - custom code
  - commercially available e-commerce packages
- No language is specified for application code ==> submitted benchmarks used C instead of Java
- Entire system must display full ACID properties
- The electronic commerce function must include at a minimum
  - SSL, shopping cart, credit card verification, secure on-line payment authorization

..... 337

:

### Components

- Main components of SUT (system under test) are:
  - Web server(s)
  - application server(s)
  - communication interface(s)
  - DB server(s)
- Additional components that may be included are:
  - load balancers
  - Web caches
  - index or directory servers
- All defined components must be commercially available

..... 338

:

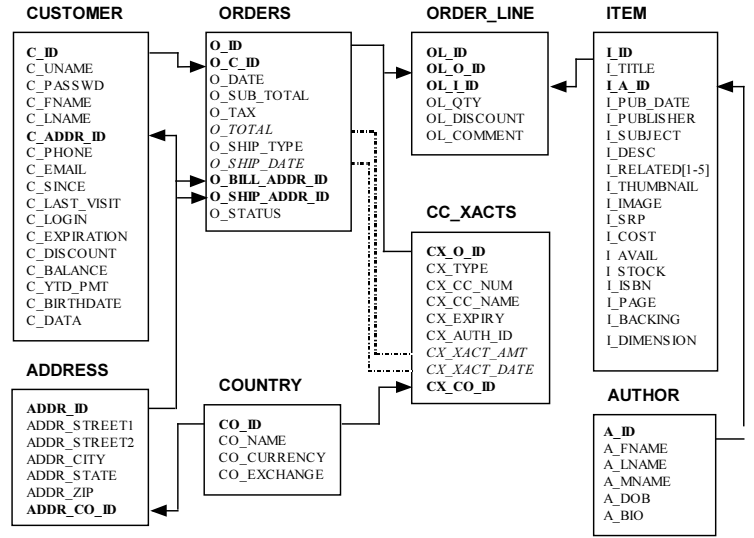
### Benchmark issues

- Most of the Web interactions result in some kind of database access
  - DB server is one of the most stressed components
  - DB design and query formulation is essential
  - to deemphasize DB utilization (or reduce DB workload) the specification allows the use of commercially available
    - index servers
    - Web cache products and caching of dynamic content up to 30 secs
    - multiplexing, routing and load balancing

..... 339

:

### Database Schema



..... 340

## Web Interactions and Workload Profile

- Customer registration and login
- A shopping cart data structure must be maintained throughout each shopping session
- A Web server access log must be maintained
- Promotional items must be displayed
- Best Sellers by topic must be dynamically derived
- Search and browse functionality must be provided
- Order status information must be available
- Buy request and buy confirmation functions exist
- Maintenance functions (e.g. price changes)

341

## Benchmark's home page

TPC Web Commerce Benchmark (TPC-W)

Home Page

Welcome back John Doe

Click on one of our latest books to find out more!

59227 24568 84429 58639 49145

**What's New** **Best Sellers**

ARTS	NON-FICTION	ARTS	NON-FICTION
BIOGRAPHIES	PARENTING	BIOGRAPHIES	PARENTING
BUSINESS	POLITICS	BUSINESS	POLITICS
CHILDREN	REFERENCE	CHILDREN	REFERENCE
COMPUTERS	RELIGION	COMPUTERS	RELIGION
COOKING	ROMANCE	COOKING	ROMANCE
HEALTH	SELF-HELP	HEALTH	SELF-HELP
HISTORY	SCIENCE-NATURE	HISTORY	SCIENCE-NATURE
HOME	SCIENCE-FICTION	HOME	SCIENCE-FICTION
HUMOR	SPORTS	HUMOR	SPORTS
LITERATURE	TRAVEL	LITERATURE	TRAVEL
MYSTERY	YOUTH	MYSTERY	YOUTH

Shopping Cart Search Order Status

342

## Customer Registration Page

TPC Web Commerce Benchmark (TPC-W)

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Customer Registration Page

I am an existing customer  
 I am a first time customer

If you're an existing customer, enter your Username and Password:

Username:   
 Password:

If you're a first time customer, enter the details below:

Enter your birth date (mm/dd/yyyy):   
 Enter your First Name:   
 Enter your Last Name:   
 Enter your Address 1:   
 Enter your Address 2:   
 Enter your City, State, Zip:   
 Enter your Country:   
 Enter your Phone:   
 Enter your E-mail:

Other Customer Data:

Submit Search Home

343

## Buy Request Page

TPC Web Commerce Benchmark (TPC-W)

Buy Request Page

**Billing Information:**

Firstname: John  
 Lastname: Doe  
 Address 1: 1 Some Place  
 Address 2: Apt 42  
 City: Durbide  
 State: CA  
 Zip: 91234  
 Country: Andorra  
 Email: me@myscompany.com  
 Phone: 123-456-7890  
 Username: ALNINBABABASE  
 C\_ID: 2880005

**Shipping Information:**

Address 1:   
 Address 2:   
 City:   
 State:   
 Zip:   
 Country:

**Order Information:**

Qty/Product  
 1 Title: BABABAWATCJIndividual, ethnic results should go. Figures - Backlog USED  
 SSP: \$790.79, Your Price: \$674.54  
 1 Title: Share cover near the great. SALLECOJATTSERDperme items - Backlog PAPERBACK  
 SSP: \$318.75, Your Price: \$308.46

Subtotal with discount (28%): \$785.85  
 Tax: \$64.83  
 Shipping & Handling: \$9.00  
 Total: \$859.68

Credit Card Type:  VISA  MASTER-CARD  DISCOVER  AMERICAN EXPRESS  DINERS  
 Name on Credit Card:   
 Credit Card Number:   
 Credit Card Expiration Date:   
 Shipping Method:  AIR  UPS  FEDEX  SHIP  COURIER  MAIL


Process Order Shopping Cart Home

344



## Admin Request Page

**TPC Web Commerce Benchmark (TPC-W)**



**Admin Request Page**

**Title: Fixed, other values will have to BABAOGULSESEBA meet**


Author +d,N BABABAULREBAAT

Suggested Retail: \$556.44  
 Our Current Price: \$335.53

Enter New Price \$

Enter New Picture

Enter New Thumbnail




[Submit Changes](#) [Search](#) [Home](#)

349

## Admin Confirm Page

**TPC Web Commerce Benchmark (TPC-W)**



**Admin Confirm Page**

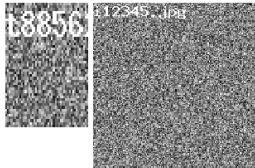
**Product Updated**

**Title: Fixed, other values will have to BABAOGULSESEBA meet**

Author +d,N BABABAULREBAAT

Description: /ua@a^bk9hSLvP3\$ishlnvKIUMf\*O@ 8BM/SMAn00p&j\$Bxy0?/[R#6XY^1&ZUo5v%~pd  
 zjPkd&kOjY9iB2yv\*A\$9h\*maAvhV0#u3#uvY?kF[z[O2yZM##ORVPeLJP3v[r-D06\_Jd{x,b,wHWTkU?CC4u{UXXI?&c&[d#-  
 &Jf,C6X{+LD+vaBAJCdg,(U-Pj#BH[o.v@GbYsck6,Uac kOvx#H{83\_bEm=(L\$HVZH.cK4L6saz^,ajeQz\*32\$X1)3xCP?  
 +W9&OQ+ve)KCLq&BIOh9/B\*ZH[K]is)md[O?LjK5[SLDeve?49]#@/(M=-dG;So1]J[4wds?]?5y[1]\$pe--SvbV7rm.ap Y=-L1  
 Zox OZ@pdW{O]YDvnvV1SBR- 1 Y&.l

Suggested Retail: \$556.44  
 Our Price: \$345.53  
 You Save: \$210.91



[Search](#) [Home](#)

350

## Web Page Consistency

- Any update transaction must be reflected with consistency in subsequent web pages returned to the browser

351

## Mix of Web Interactions

Web Interaction	Browsing Mix (WIPsb)	Shopping Mix (WIPs)	Ordering Mix (WIPSo)
<b>Browse</b>	<b>95 %</b>	<b>80 %</b>	<b>50 %</b>
Home	29.00 %	16.00 %	9.12 %
New Products	11.00 %	5.00 %	0.46 %
Best Sellers	11.00 %	5.00 %	0.46 %
Product Detail	21.00 %	17.00 %	12.35 %
Search Request	12.00 %	20.00 %	14.53 %
Search Results	11.00 %	17.00 %	13.08 %
<b>Order</b>	<b>5 %</b>	<b>20 %</b>	<b>50 %</b>
Shopping Cart	2.00 %	11.60 %	13.53 %
Customer Registration	0.82 %	3.00 %	12.86 %
Buy Request	0.75 %	2.60 %	12.73 %
Buy Confirm	0.69 %	1.20 %	10.18 %
Order Inquiry	0.30 %	0.75 %	0.25 %
Order Display	0.25 %	0.66 %	0.22 %
Admin Request	0.10 %	0.10 %	0.12 %
Admin Confirm	0.09 %	0.09 %	0.11 %

352



## Initial (small) configuration

Table Name	Cardinality (in rows)	Typical Row Length (in bytes)	Typical Table Size (in bytes)
CUSTOMER	2880 *(number of EB)	760	2,188,888 k
COUNTRY	92	70	6.44 k
ADDRESS	2 * CUSTOMER	154	887,040 k
ORDERS	.9 * CUSTOMER	220	570,240 k
ORDER_LINE	3 * ORDERS	132	1,026,432 k
AUTHOR	.25 * ITEM	630	1,575 k
CC_XACTS	1 * ORDERS	80	207,360 k
ITEM	1k, 10k, 100k, 1M, 10M	860	8,600 k

**Note 1:** Table sizes are computed for 1,000 EBs and 10,000 items

**Note 2:** Values for the ITEM table do not include the item's image and thumbnail

**Note 3:** The typical row lengths and table sizes given above are examples of what could result from an implementation. They do not include storage and access overheads.

**Note 4:** ORDER\_LINE cardinality will vary slightly due to the random number of rows generated per order as specified in clause 4.7.1. Cardinality must meet a minimum requirement of 2.95 times the number of rows in the ORDER table.

353

## SSL

- TPC-W specifies the use of SSL V3 for some interactions
- The specification does not prescribe how encryption is accomplished, i.e. whether
  - encryption is done by the Web server in SW
  - encryption is accomplished by specialized HW
- The sponsor of the benchmark must weigh these issues based on cost/performance

354

## Performance Requirements

- Web Interaction Response Time
- During the measurement interval, at least 90% of web interactions of each type must have a WIRT better than the one specified below

	Admin Confirm	Admin Request	Best Sellers	Buy Confirm	Buy Request	Customer Regist.	Home	New Products	Order Display	Order Inquiry	Product Detail	Search Request	Search Results
<b>90% WIRT Constraint</b>	20	3	5	5	3	3	3	5	3	3	3	3	10

355

## Reporting Details

- All measurements must be
  - made in steady state,
  - must be reproducible,
  - must run during an uninterrupted period of at least 30 min
- Reporting details
  - Frequency distribution of WIRT of all web interactions
  - graph of measured throughput vs. elapsed time
  - CPU utilization
  - memory utilization
  - page/swap activity
  - database I/O activity
  - system I/O activity
  - Web server statistics

356

:

### Optional Statistics and Overload Run

- Optional statistics include
  - breakout of CPU busy by key, state, SW comp, etc.
  - breakout of true CPU idle and wait for I/O
  - queue statistics
  - page/swap breakout over specific devices incl. Service times
  - web cache hit rates
  - utilization and contention for I/O components, etc. etc.
- Overload run is required to show how the system will behave under exceptionally high load

..... 357

:

### Cost of Benchmark

- Cost for producing a TPC-W benchmark is considerable (difficult for small companies)
- Members of TPC are large corporations
- Cost of platform must include all specified components plus one year of support
  - small scale \$180 000 - \$ 350 000
  - large scale \$ 500 000 - \$ 1 300 000
  - 3-year cost of ownership (excluding development cost of application code) for IBM's small scale Netfinity benchmark is \$ 348 879.-
  - costs used in pricing may not include discounts not generally available

..... 358

:

### Submission, Review and Appeals

- Sponsor of a benchmark must submit full disclosure of all costs and implementation details
  - clarification of wording submitted to and resolved by Technical Advisory Board
  - based on clarifications requested by sponsors, TAB may modify the specification
- Submissions may be challenged, challenges are resolved by the TAB

..... 359

:

### Recommended Reading for the Course

- Menascé, Daniel A., Almeida, Virgilio A. F.; “Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning”, Prentice Hall PTR, 2000, [www.cs.gmu.edu/~menasce/ebook](http://www.cs.gmu.edu/~menasce/ebook)
- Menascé, Daniel A., Almeida, Virgilio A. F.; “Capacity Planning for Web Performance: Metrics, Models, & Methods”, Prentice Hall PTR, 1998, [www.cs.gmu.edu/~menasce/webbook/](http://www.cs.gmu.edu/~menasce/webbook/)
- Kalakota, Ravi, Whinston, Andrew B.; “Electronic Commerce - A Manager’s Guide”, Addison Wesley, 1996.

..... 360

:

Recommended Reading for the Course

- Boucher, Karin, and Katz, Fima; "Essential Guide to Object Monitors", Wiley 1999, [www.wiley.com/compbooks/boucher](http://www.wiley.com/compbooks/boucher)
- Cilia, M., Liebig, C., Buchmann, A.; "Metaauctions ...", WECWIS 2000
- Cardellini, V., Colajanni, M., Yu, P.; "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, May-June 1999.
- Loosley, C., Douglas, F.; "High-Performance Client/Server", John Wiley, 1998.

..... 361

:

Recommended Reading for the Course

- Franks, G., Hubbard, A., Majumdar, S., Petriu, D., Rolia, J., Woodside, C.M.; „A Toolset for Performance Engineering and Software Design of Client/Server Systems“, Performance Evaluation J., vol. 24, no. 1-2, 1996, pp 117-135.
- Rolia, J.A., Sevcik, K.C.; „The Method of Layers“, IEEE Trans. SW Eng., vol 21, no. 8, 1995.
- Apostolopoulos, G., Peris, V., Saha, D.; „Transport Layer Security: How much does it really cost?“, Proc. IEEE Infocom 99

..... 362

:

Recommended Reading for the Course

- SET, „The SET Standard Technical Specification“, [www.setco.org/download.html](http://www.setco.org/download.html)
- Pitkow, J. E.; „Summary of WWW Characterization“, Proc. World Wide Web Conf., 2(1), Jan. 1999, pp. 3-13.
- Wang, X., Schulzrine, H., Kandlur, D., Verma, D.; „Measurement and Analysis of LDAP Performance“
- [www.tcp.org](http://www.tcp.org)

..... 363