



Metering and Accounting for Composite e-Services (MACS)

Arun Kumar

IBM India Research Laboratory

New Delhi, India

(Joint work with Vikas Agarwal and Neeran Karnik)

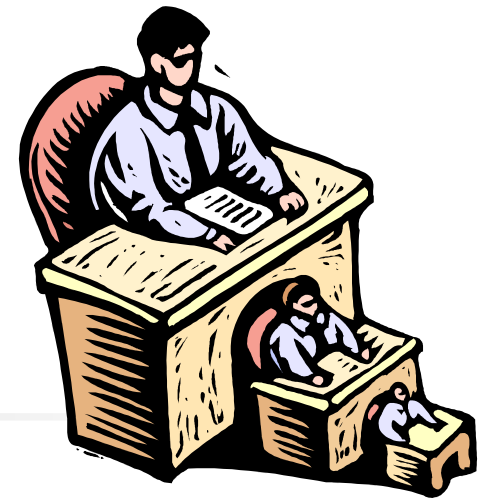
June 24th 2003, Newport Beach, California



Agenda

- Introduction
- Accounting Infrastructure
- e-Service Categories
- Metering and Accounting Architecture
- Summary

Introduction



■ e-Services

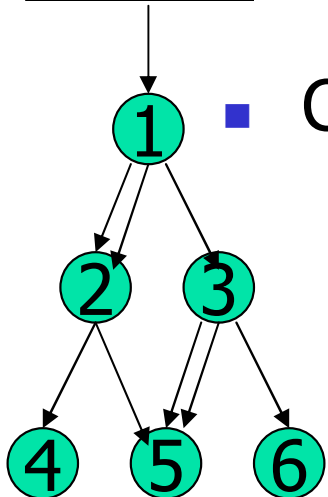
- software services programmatically accessible over the network
- examples include web services, grid services

■ Metering and Accounting

- essential for commercial e-services
- for enforcing usage quotas, for analyzing usage patterns
- what do we mean by service usage ?
- what metrics to use ?

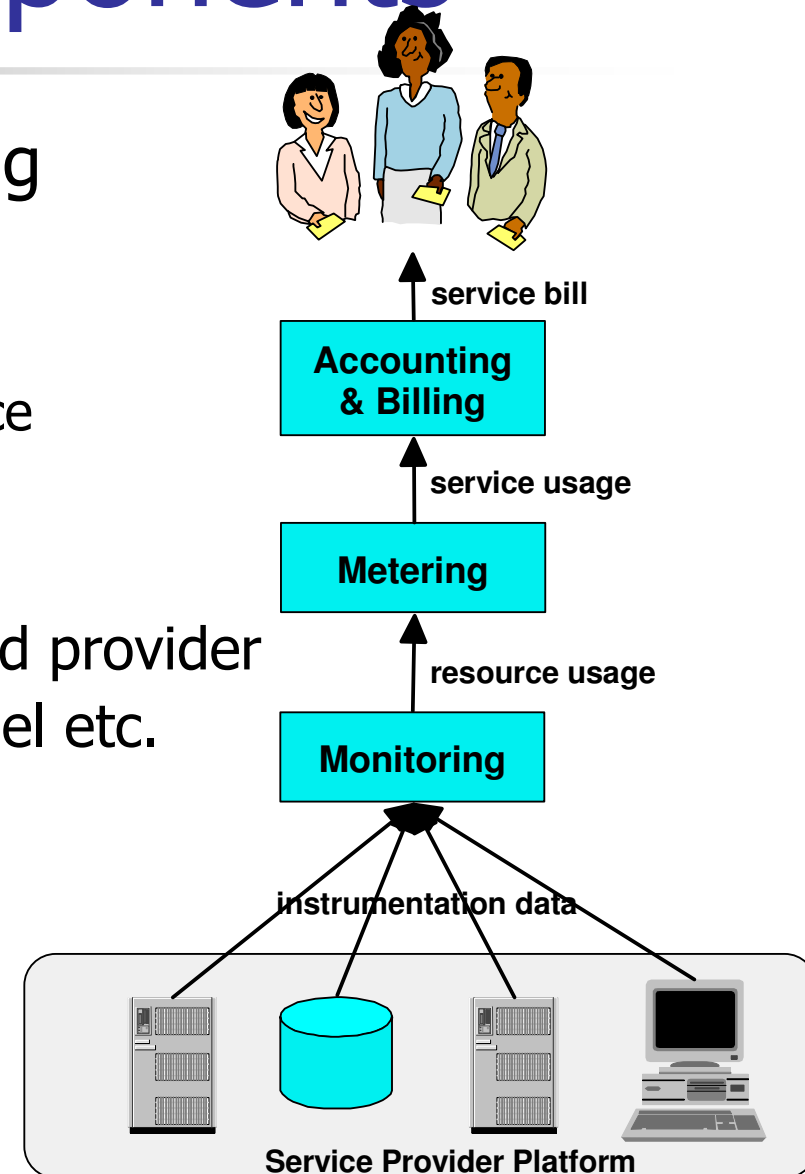
■ Complexity introduced by **composite e-Services**

- a service may invoke other services in processing a request, resulting in a hierarchy
- each service may be autonomously owned & operated
- each request must be independently accounted for



Architectural Components

- Information Flow for usage accounting
- Charging → **metering**
 - defines usage metrics of the service
 - CPU cycles consumed, for Compute Service
 - Size of message, for Sendmail service
- Business Model → **accounting**
 - define relationship between consumer and provider
 - e.g. subscription, pay-per-use, lease model etc.
- Pricing → **billing**
 - service may be commoditized or priced based upon value-add
 - price as a function of usage, customer priority, flat-fee etc.



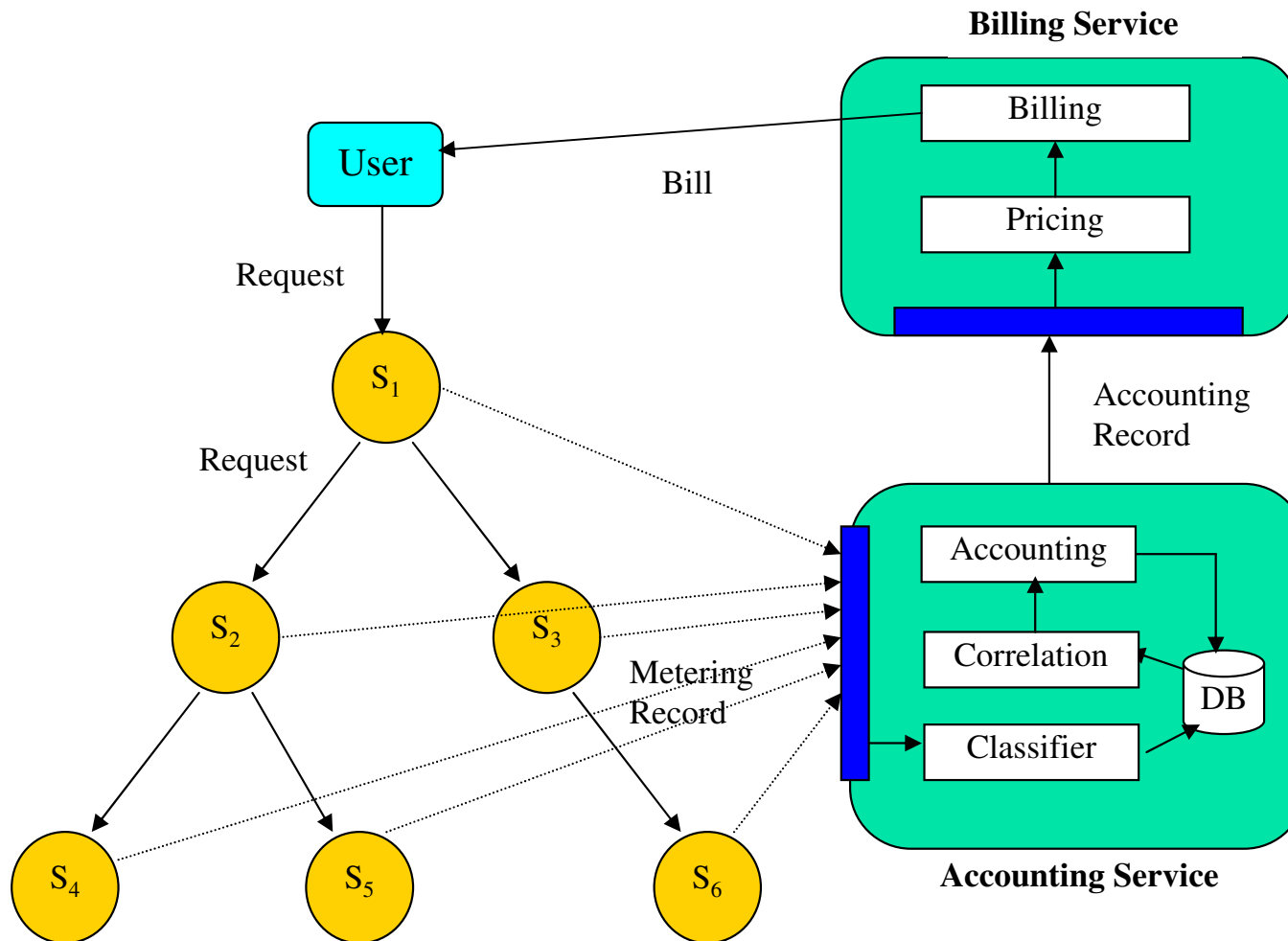


Types of Services

- Resource Centric [Type R]
 - wrappers around *resources*
 - e.g. service interface to a printer
 - resource usage metrics – need active metering
- Transactional [Type T]
 - charges for the *function* it offers
 - e.g. email delivery service
 - usage is expressed in *application-level* metrics
 - ... can be predicted from request parameters
 - does not require active metering

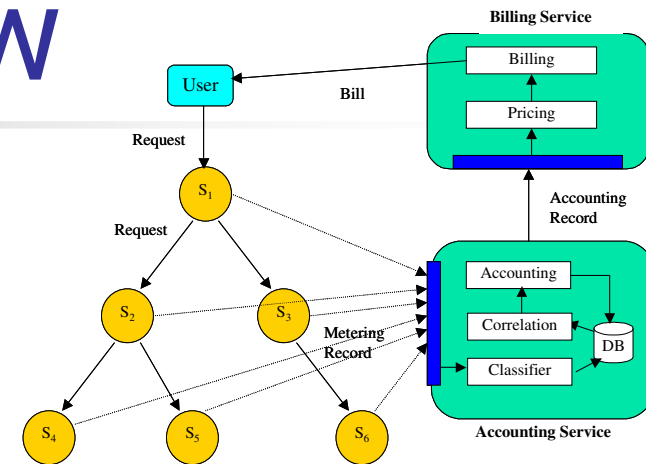
For composite services, the charging model of higher-level service determines the charging model of the composite service

The MACS Architecture

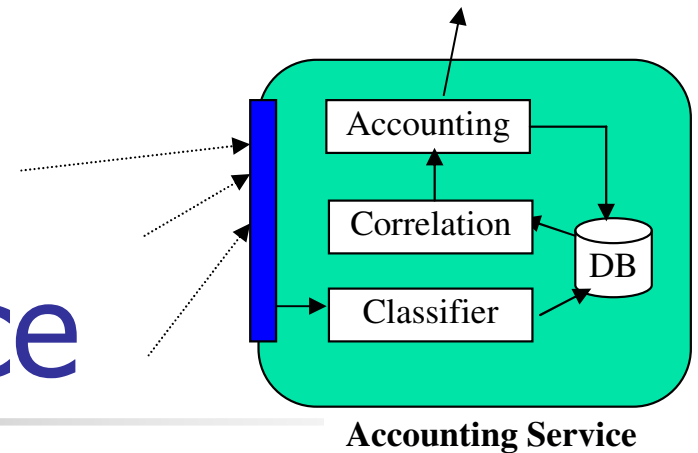


Architecture overview

- Services register with an accounting service
 - operations and their category
 - business models supported
 - usage metrics
- Each service treats its immediate caller as its client and charges it accordingly
- Each e-Service reports per-request usage, in a **partial metering record**, to the Accounting service
 - ‘partial’ because it contains usage metrics for that service alone
 - self-contained for a leaf node
- Accounting service generates accounting records from these partial metering records
- Billing service computes bills periodically
 - applies provider specified pricing policy to accounting record to compute the amount to be billed



Accounting Service



- Creates and maintains an *account* for each <customer, provider> pair
- The account
 - is the entity to which the service usage is attributed
 - maintains business relationship between customer and provider
 - can be shared by multiple users in an organization
- The *classifier* indexes the incoming records and stores them
- For each request, the *correlator* creates a **complete metering record**
 - correlates its partial metering record with complete metering records of requests to underlying services
- The *accounting* module aggregates complete metering records
 - generates *accounting records* for each account and sends to Billing service



Usage Reporting Protocol

- Each incoming request contains a unique request-id
- Partial metering record is generated and reported after local processing of the request is complete
 - all invocations to other services have been initiated by then
 - the invocations may be asynchronous

request-id	service-id	user-id	operation	list of req ids	metric-1		metric-n
------------	------------	---------	-----------	-----------------	----------	--	----------

Metering Record



Correlation Algorithm

function correlate(**Record**)

- if operationType(**Record**) = 'T' // a record for transactional
- mark (**Record**, 'complete') // operation is always complete
- return // no need to aggregate further
-
- **List** = getInvocationList(**Record**) // requests to underlying services
- if (**List** is empty) // if record is for a leaf node
- mark (**Record**, 'complete') // then it is complete
- return // trivial case
-
- if (complete(**List**)) // check if all records in the list are complete
- for each request **ReqId** in **List**
- **MetRec** = getMeteringRec(**ReqId**) // get complete record from DB
- aggregate(**Record**, **MetRec**) // compute the aggregate usage
- mark (**Record**, 'complete')
- else
- skip this record // will be processed later
- **end**



Correlation Across Accounting Services

- Services in a composition hierarchy may subscribe to different Accounting services
- Correlation:
 - The *correlator* of *callee's* accounting service must know the accounting service of *caller*
 - Correlator checks whether caller's accounting service is different from itself
 - If so, sends the *complete* metering record to caller's accounting service for further correlation
- This can be enabled by...
 - Caller identifying its accounting service in request. The callee includes this in its metering record
 - Accounting service of caller can be specified while establishing SLA with the callee



Summary & Future Work

- Metering and Accounting Architecture
 - scalable, supports distributed accounting
 - supports dynamic composition
 - supports service autonomy
 - requires standardization
- The Road Ahead
 - IBM Emerging Technologies Toolkit (ETTK)
[previously known as WSTK]
 - MACS subsystem implemented by Ashwin T.V. and Sunil Chandra
 - available from <http://www.alphaworks.ibm.com/ettk>
 - Policing and admission control
 - pre-paid calling card
 - requires interval accounting



Questions ?



THANK YOU .