Monitoring and Diagnosing Applications with ARM 4.0

(ARM = Application Response Measurement)

Mark W Johnson, IBM/Tivoli

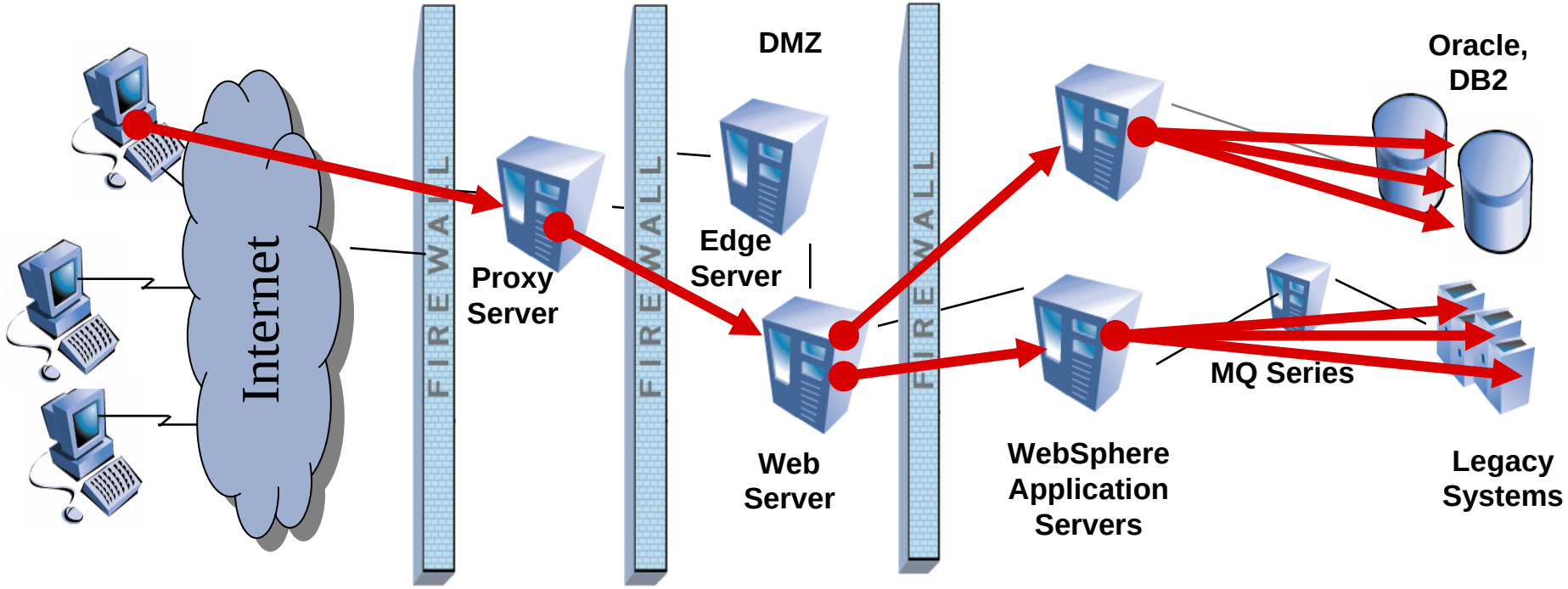
mwj@us.ibm.com

Agenda

- ARM Overview
- Using ARM Measurements
- Adding Instrumentation to Applications
- Summary

e-Business Transactions are Complex

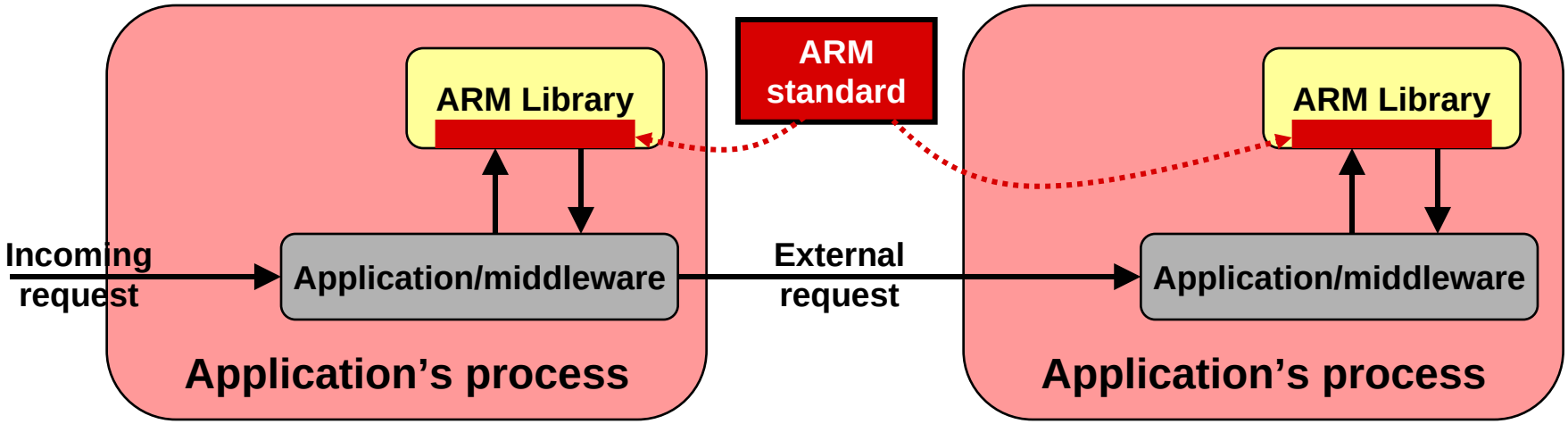
(and this is a simple example)



- What are application service levels (response time, availability)?
- Where are transactions failing?
- Where are there bottlenecks?
- Which IT resources support a business service?

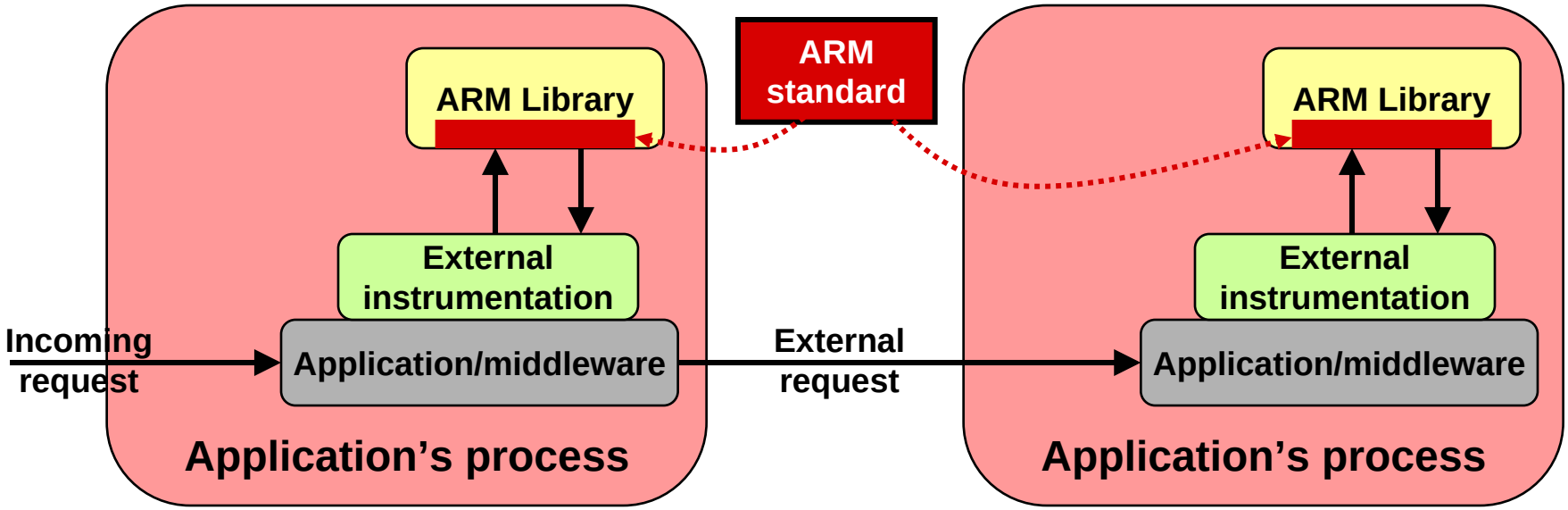
What ARM looks like to an application

- ARM is an interface to a dynamic/shared library
- Application loads the ARM library
- Application calls ARM while processing requests

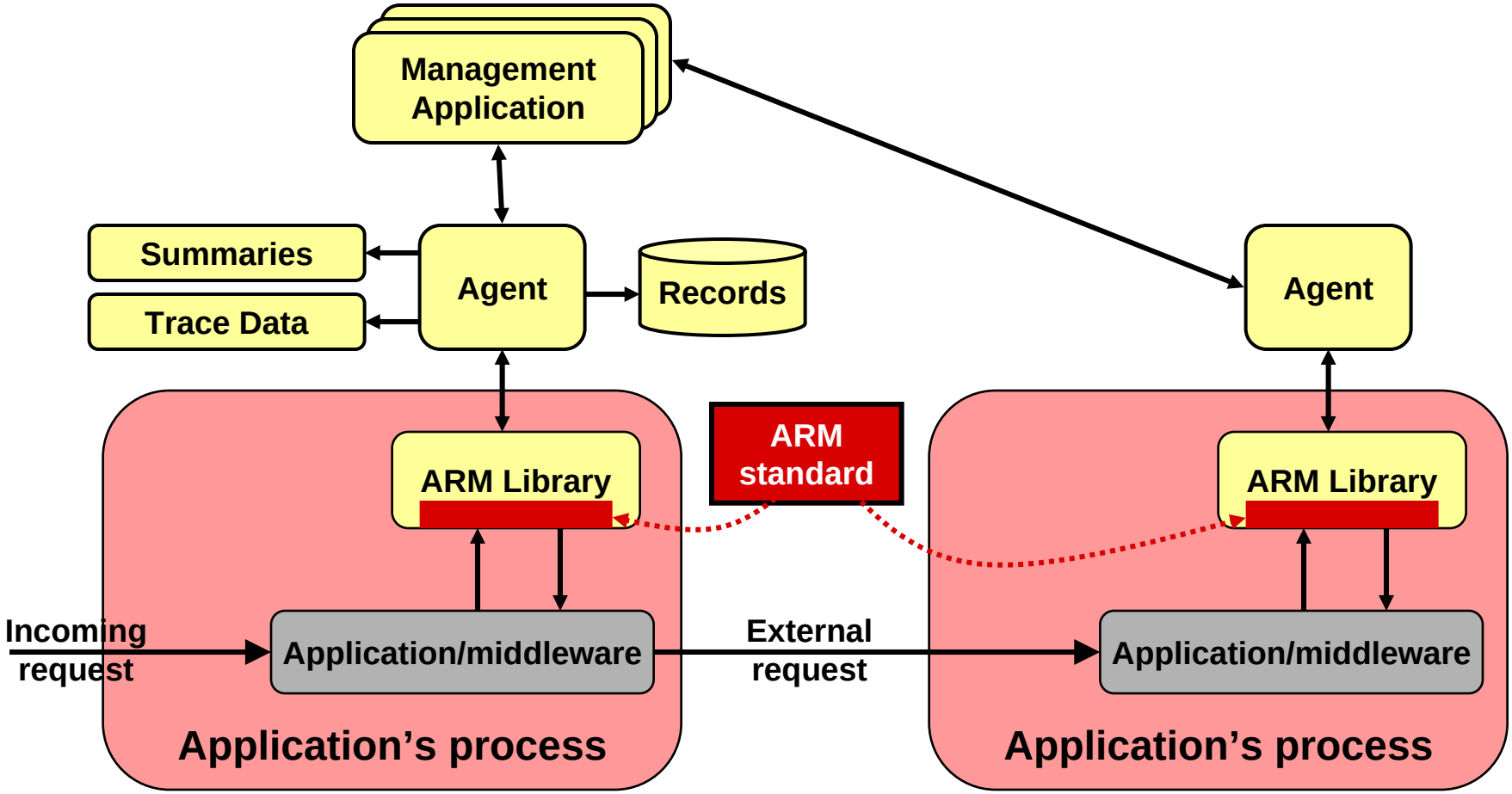


ARM is used with proxy instrumentation

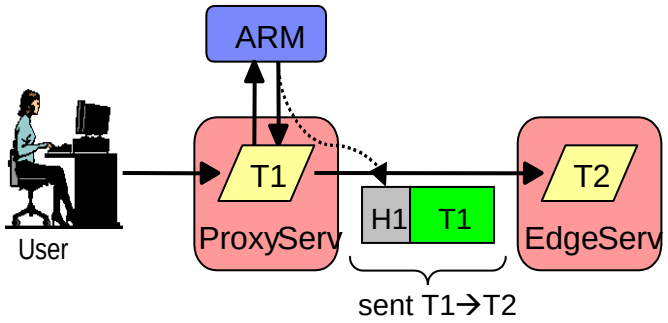
- Instrumentation uses existing hooks into the application
- Instrumentation loads the ARM library and makes calls to it
- Instrumentation "piggy-backs" correlation token to downstream calls



What goes on behind the scenes



Application Response Measurement – Overview



Application Response Measurement (ARM) is an Open Group Standard for measuring units of work, such as transactions & jobs.

When a unit of work, is initiated, the application calls the ARM interface.

The ARM library starts timing the unit of work and may assign attributes to this business transaction, such as a service class and accounting classes.

If the application requests it, the ARM library may create an ARM correlator token.

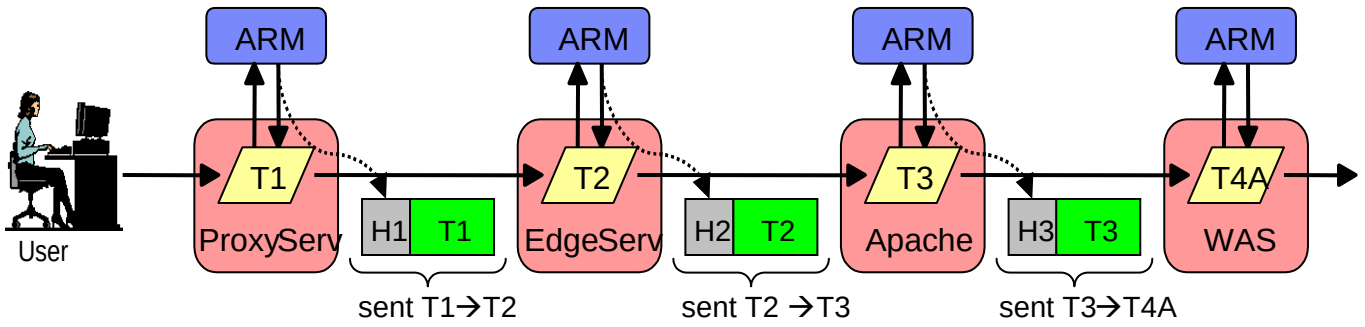
The application then passes the token to any applications it invokes.

Correlating Transactions End-to-End

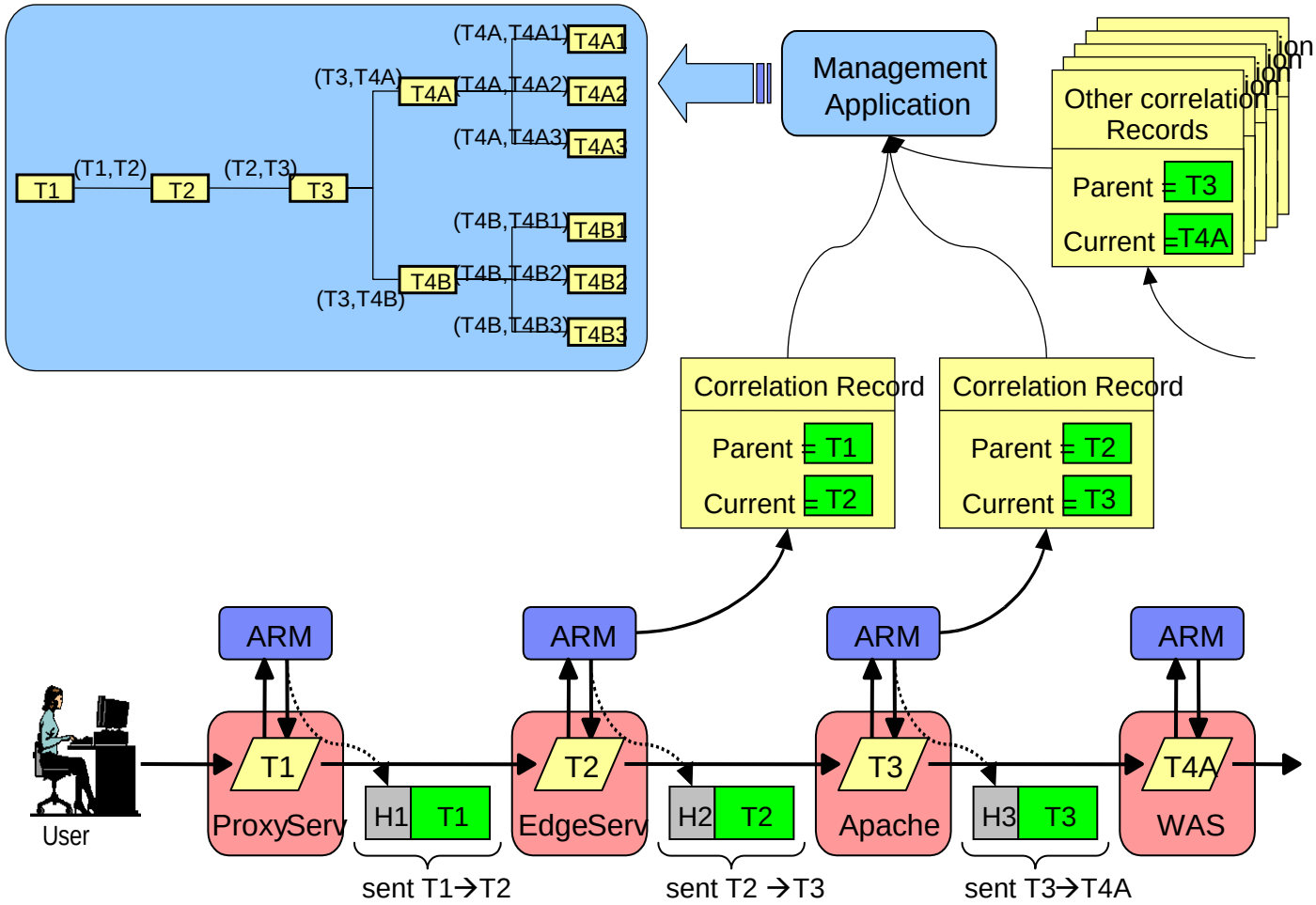
The next application in line calls ARM passing the initiator's token as its "parent". This allows the application to be associated as a 'child' application to the initiator's 'parent' role in all ARM calls.

The ARM library returns a correlator to this child application. This correlator token represents the child transaction.

This application passes the token to any applications it invokes and each called application, in turn, calls ARM to start timing the transaction and to create its own correlator, which it passes to applications it calls.



Analyzing ARM Data



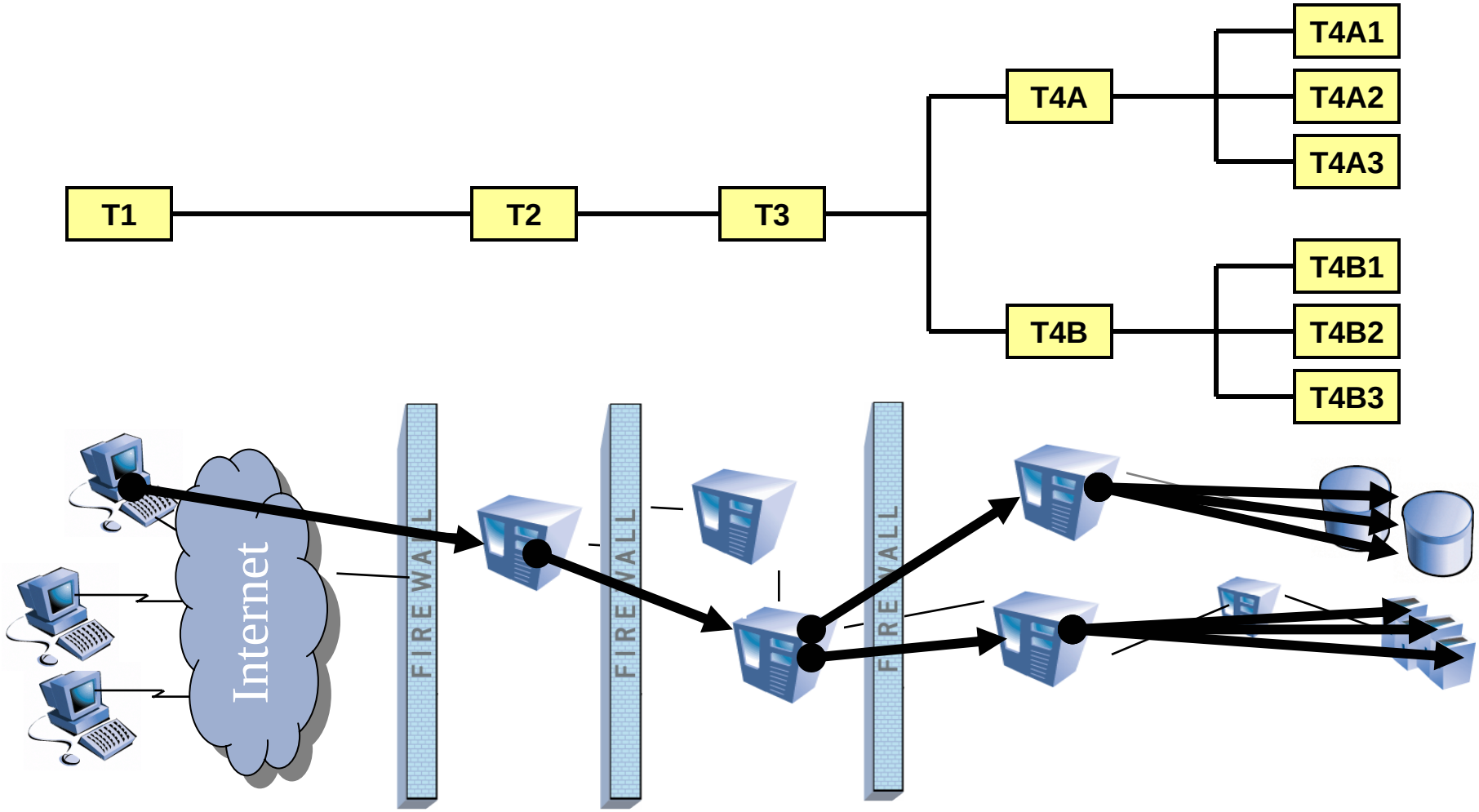
Management applications collect the correlator tokens from each application's ARM agent.

The correlator tokens pairs show all the parent/child transaction relationships.

Management applications create a relationship graph for the entire end-to-end flow.

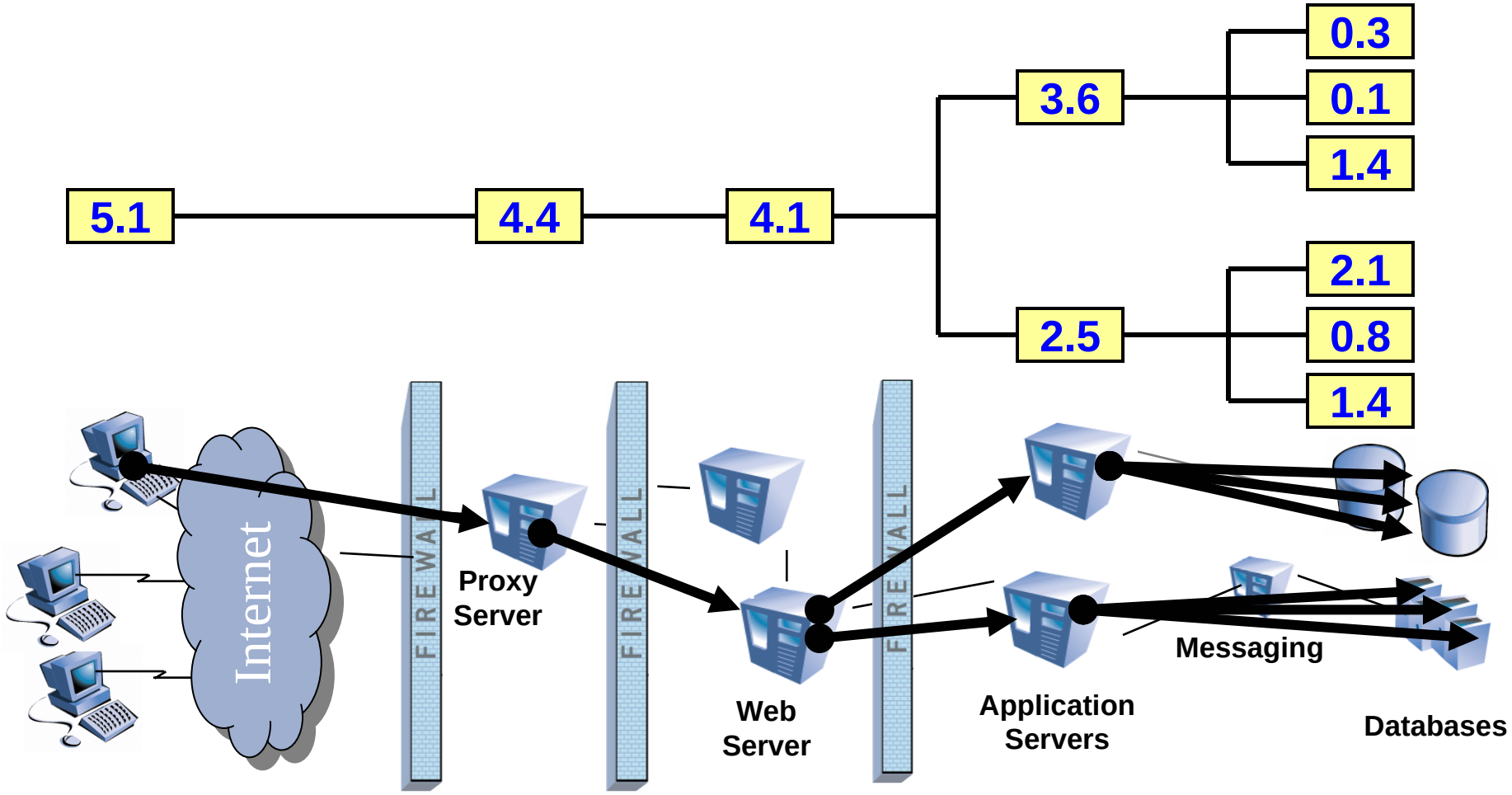
Model the end-to-end transaction

… Discover how resources support business services

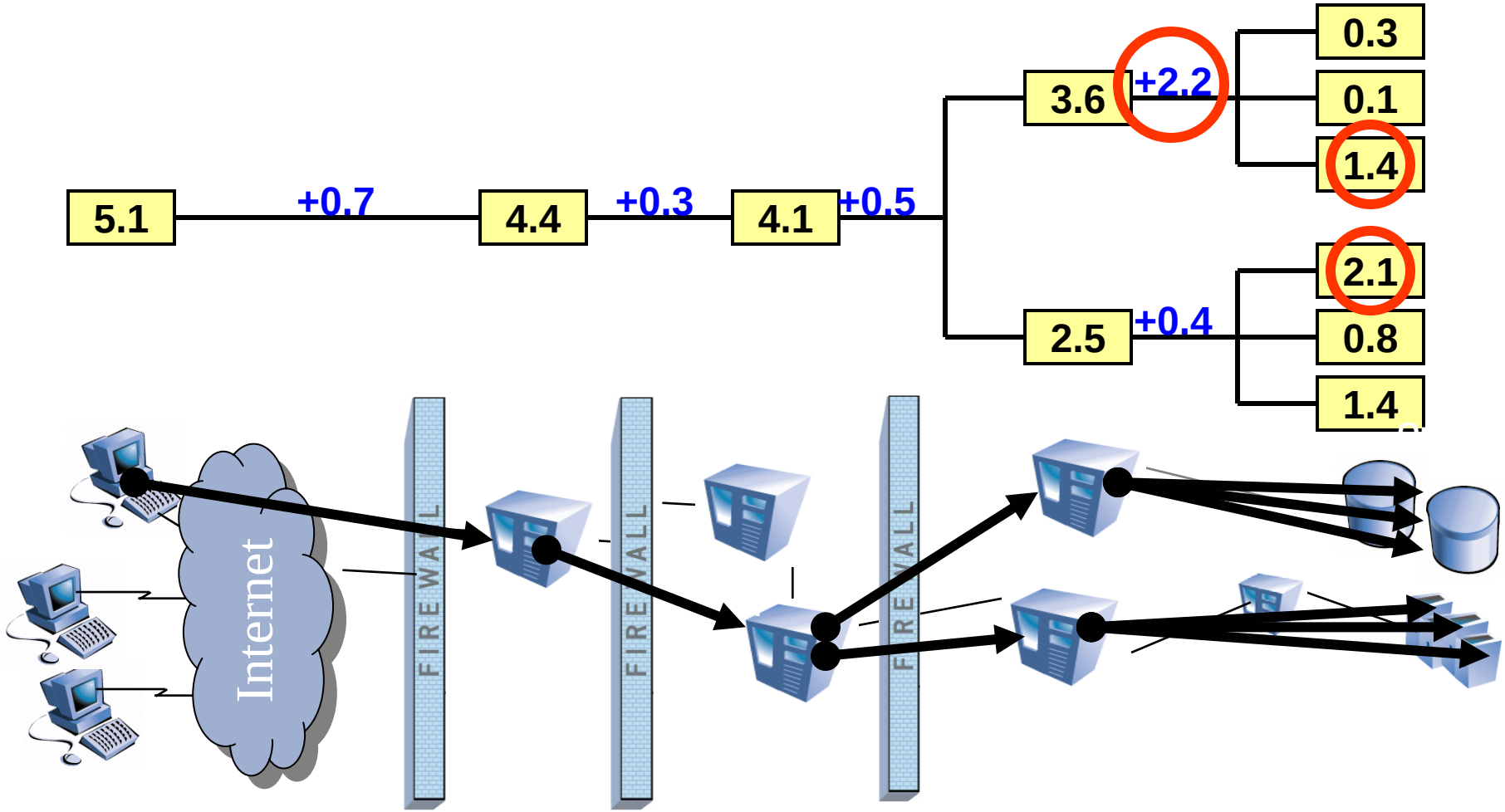


Measure response times to establish baseline

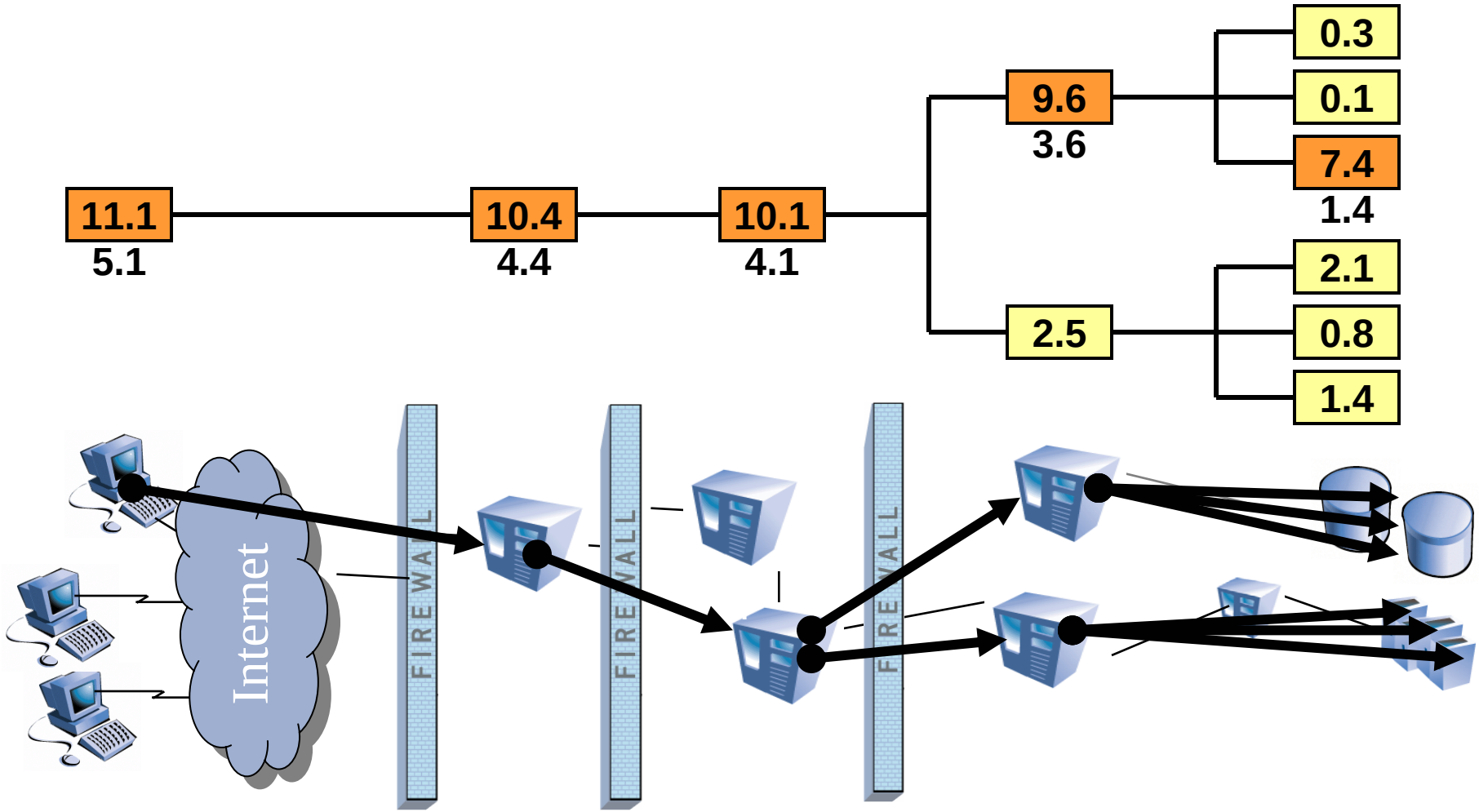
… Each time = local time + time waiting for downstream calls



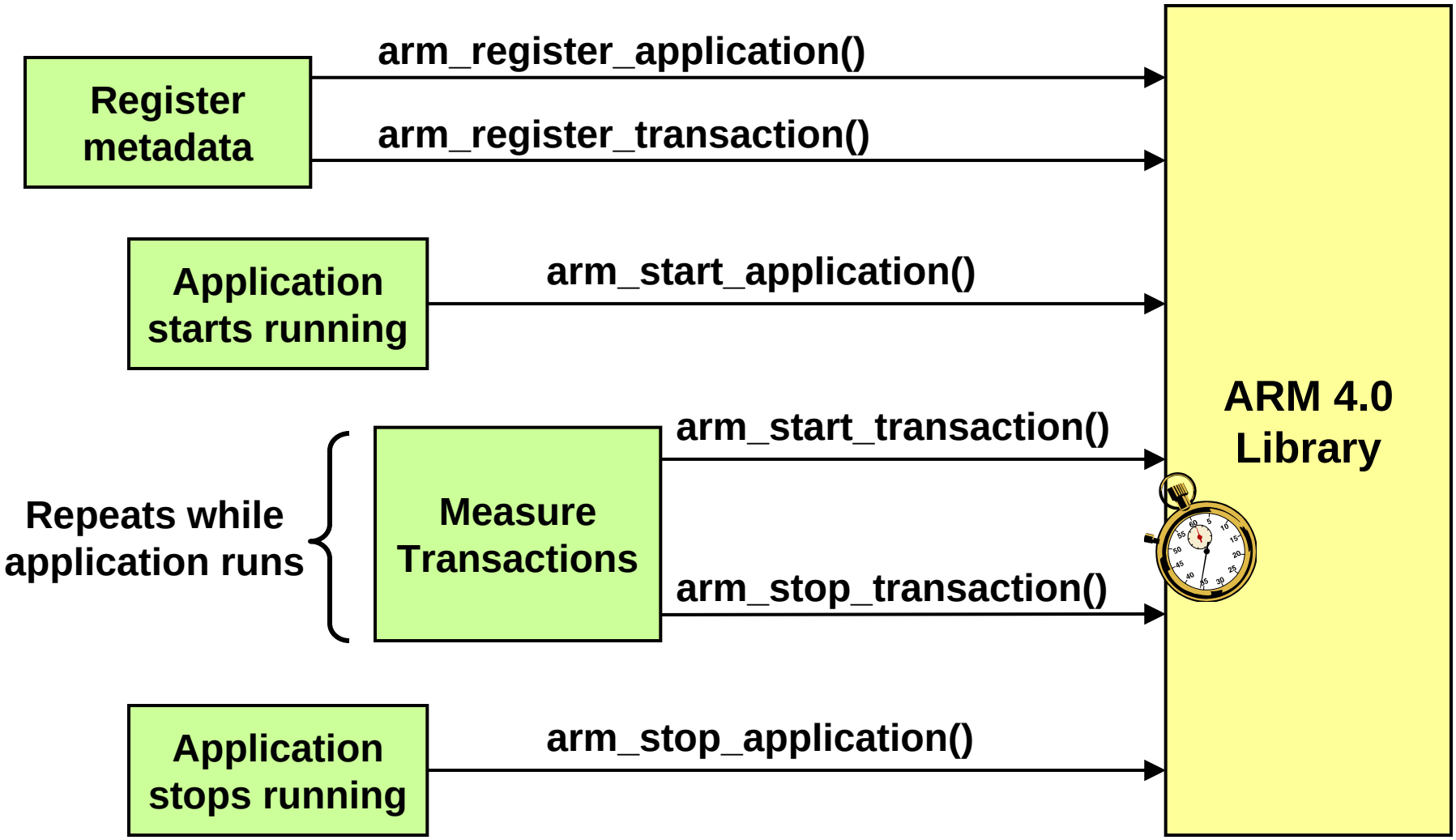
Tune application or capacity by finding transactions that most impact end user response time



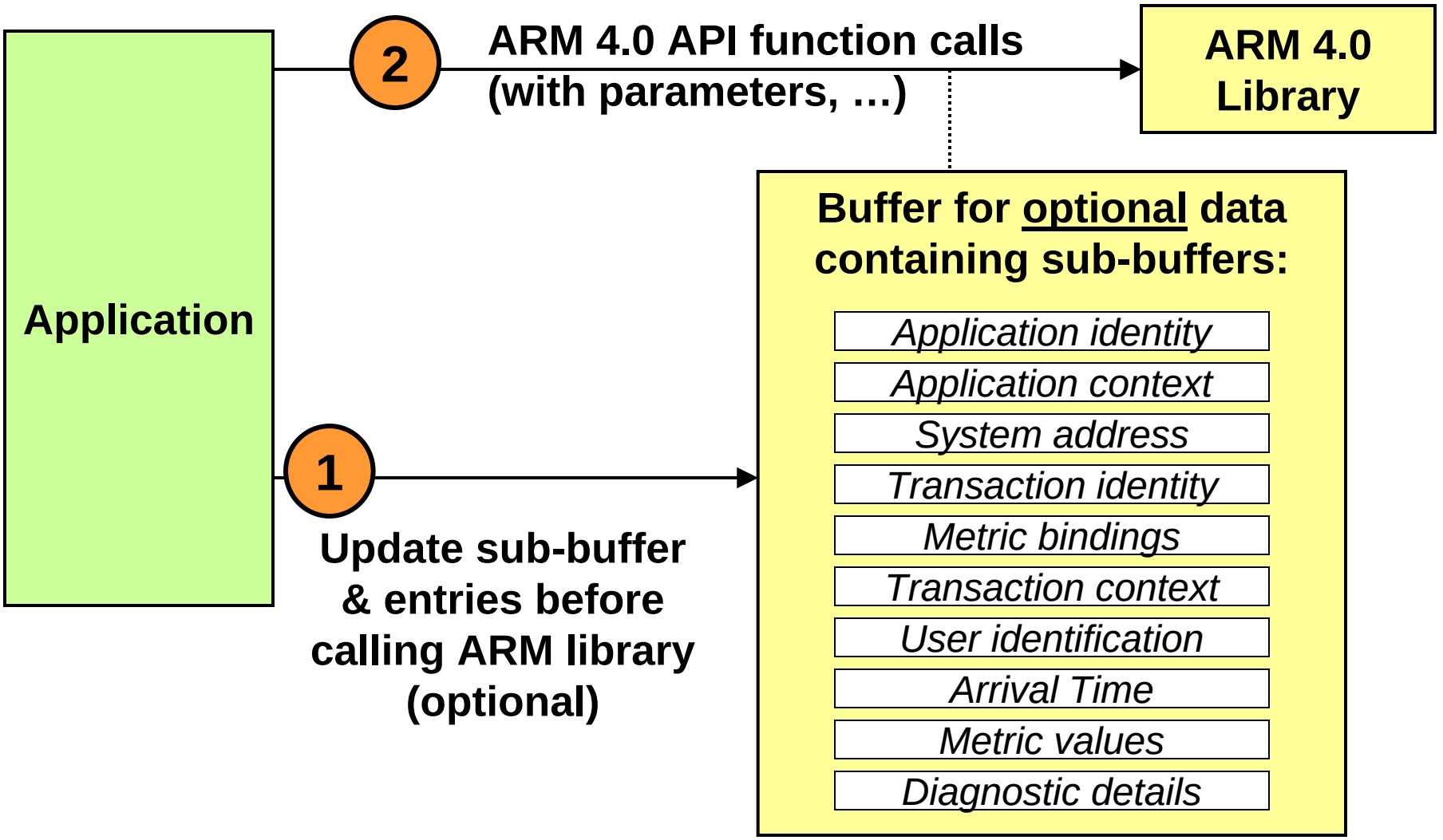
Compare to baseline to find anomalies



API Overview (C Bindings)



Function Call Pattern



Instrumentation Best Practices

- Identify transactions in a way that provides useful measurements
  … (see examples below)
- Call ARM start/stop functions close to the actual start/stop time
- Use correlators
  (receive, provide to ARM, request from ARM, send with remote calls)

Transaction Name	Transaction Interval Count	Response Time Elapsed	Response Time Mean
"Query_Individual_Acct"	500	1500 sec	3.00 sec
"Query_Business_Payable_Acct"	300	2550 sec	8.50 sec
"Query_Business_Receivable_Acct"	200	3000 sec	15.00 sec
(TOTAL) … or … "Query_Account"	1000	7050 sec	7.05 sec

What do you get with ARM?

- Specification
  - ➔ C bindings
  - ➔ Java bindings
- C Header and Java Interface files for compiling
- Software Developers Kit (SDK)
  - ➔ Freely downloadable from opengroup.org
  - ➔ Use for compiling, testing, small-scale pilots
  - ➔ Source code – use and/or change anyway you like
  - ➔ Used by developers of business applications and middleware to instrument their programs to call ARM
  - ➔ Used by developers of management solutions to provide standards-compliant libraries that implement ARM

ARM Evolution

- **ARM 1.0 (C APIs) – 1996**
  - ➔ **IBM and Hewlett-Packard collaborate to release the first version**
- **ARM 2.0 (C APIs) – 1997 & 1998**
  - ➔ **1997 – the ARM Working Group (~15 companies, including product vendors and end-users) adds transaction correlation and metrics**
  - ➔ **1998 – adopted by The Open Group as a formal standard**
- **ARM 3.0 (Java interfaces) – 2001**
  - ➔ **The Open Group adopts a Java version**
- **ARM 4.0 (C and Java) – 2003**
  - ➔ **Align the C and Java semantics**
  - ➔ **Better enable the use of ARM by middleware and proxy instrumentation**
  - ➔ **Richer notion of application and transaction identity & context**
  - ➔ **Capture a more accurate processing start time in certain situations**
  - ➔ **Associate threads to transactions**
  - ➔ **Indicate when a transaction is blocked on an external event**
  - ➔ **Applications can provide more diagnostic data, such as the SQL query text of a failing query**

Summary

- ARM helps:
  - ➔ Measure service levels
  - ➔ Analyze response time and status end-to-end
  - ➔ Capture who uses an application, and how much
  - ➔ Discover how IT resources support business services
- ARM is used to instrument many applications developed “in-house”
- ARM is being used with commercial middleware
  - ➔ IBM DB2 Universal Database
  - ➔ IBM WebSphere Application Server
  - ➔ SAS 8.2 and later
  - ➔ Siebel 7.7
  - ➔ Plug-ins for web servers (Apache, IBM, Microsoft Internet Information Services)