UNIVERSITÄT DUISBURG ESSEN

samba eXperience 2009

Monitoring Samba using the Application Response Measurement Standard (ARM)

Sven Kubiak
University of Duisburg-Essen

Agenda

- Introduction
- Current development status
- Requirements on measuring Samba
- Application Response Measurement (ARM)
- Developing the Prototype
- Results
- Lessons learned

Introduction

- Talk is based on my Bachelor Thesis at the Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen, Prof. Dr. Müller-Clostermann
  - In cooperation with ZRWest
- ZRWest is a data center for the "Deutsche Rentenversicherung" (German pension fund)
- Infrastructure with 2000+ Clients
  - Using Samba for their Branches
  - Mainly as File- and Print Server
- Work in progress!!

Introduction

- ZRWest required to monitor their Samba Servers
  - Service Level Agreement with Offices / End-Users
  - Helpdesk Support
  - Troubleshooting
- Questions that require answers (e.g.)
  - How long does it take to load the profile of a user?
  - How long does it take to mount a network drive?
  - How long does it take to mount a printer?
  - How long does it take to log on to a domain?

Current development status

- First step: Analyze and examine if it is possible to combine Samba and ARM
- Second step: Analyzing SMB and developing a prototype based on results of first step
- Third step: Implementation in test-environment
- Fourth step: Implementation in production-environment

Requirements on measuring Samba

- A monitoring solution for Samba which focus on measuring end-to-end transactions per Client
- Currently using existing monitoring solutions (e.g. ISIS – An Integration Samba Inspection Services – see sambaXP 08)
  - Problem: Monitoring is mainly based on performance analysis (CPU, RAM, etc.)
- Idea: Using the Application Response Measurement Standard (ARM) and its concepts to measure Samba features

Application Response Measurement

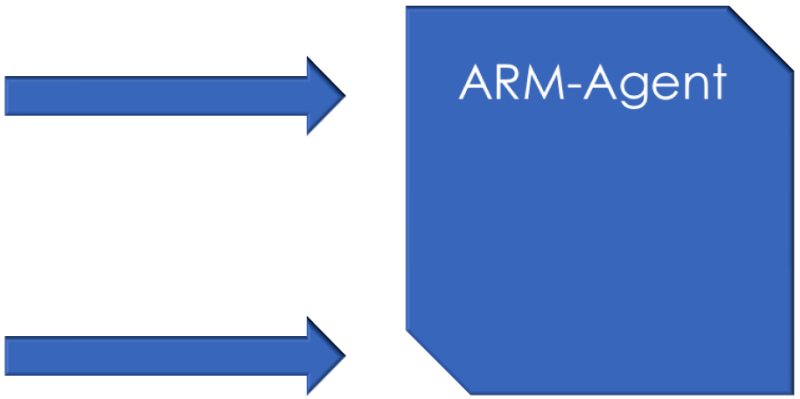
- ARM is an open standard published by the OpenGroup
- Was originally developed by IBM and Tivoli in 1996
  - Is included in IBM Websphere and IBM Application Server (e.g.)
- API for performance analysis written in C
  - Java (since Version 3.0)
  - Current stable version 4.1
  - Open-Source SDK available
- Used to gain an insight into the behavior of a (distributed) application
  - Are transactions executed, if not, what are the reasons?
  - How can one increase the performance of an application
  - How long is the response time?
  - Identifying bottlenecks in distributed Systems

Application Response Measurement

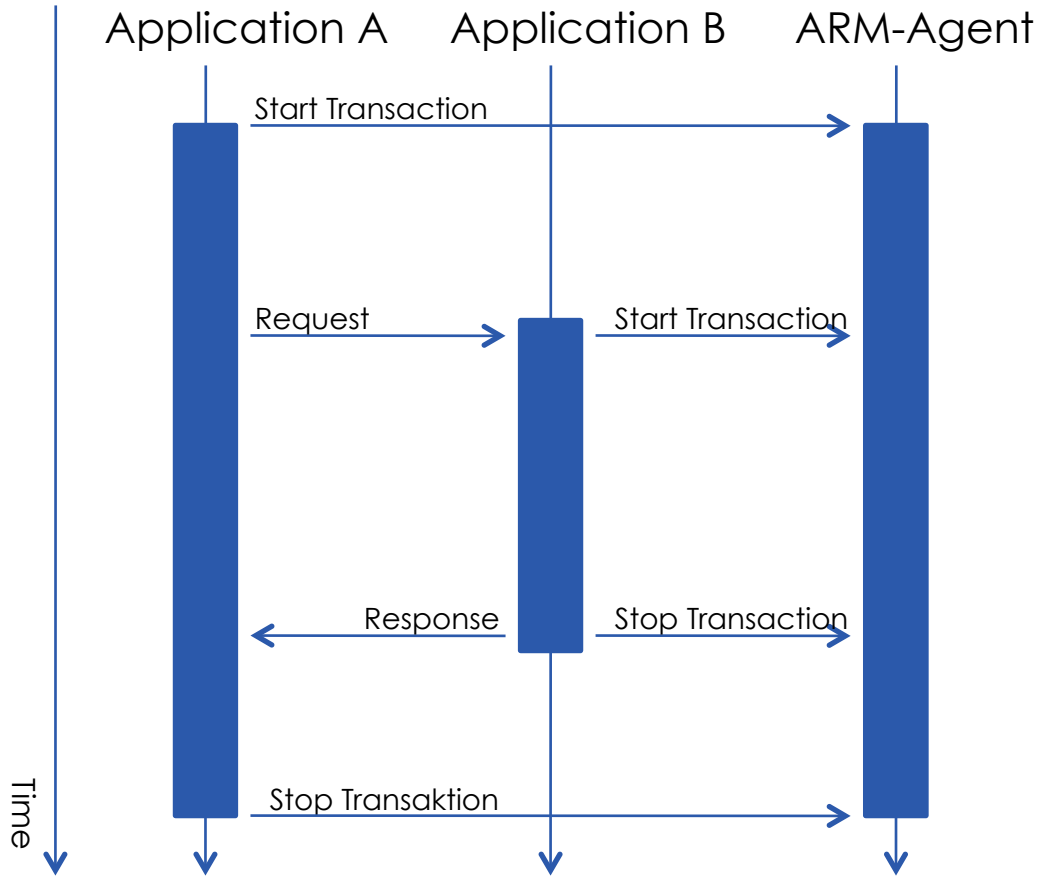
- Measurement of the runtime through the definition of transaction points
  - Simple start and stop calls in your application
- ARM-Calls are send to an ARM-Agent which does the analysis
  - Agent sets timestamps
  - Agent handle IDs
- Extend the information send to the agent
  - Metrics (e.g. counters)
  - Correlators (Parent-Child relationships)
  - etc.

Application Response Measurement (Very) simple Example

```
Load libarm

start_arm_transaction()

{Program}

stop_arm_transaction()
```



Application Response Measurement Distributed Systems Example

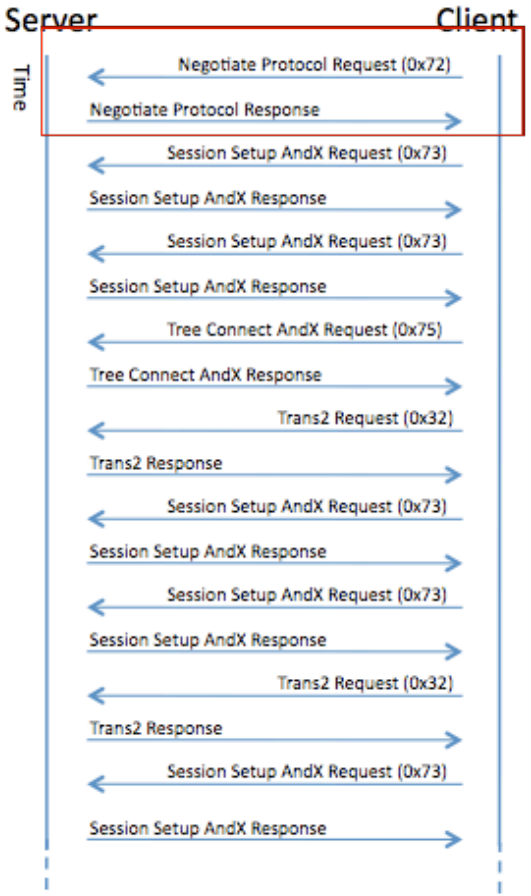


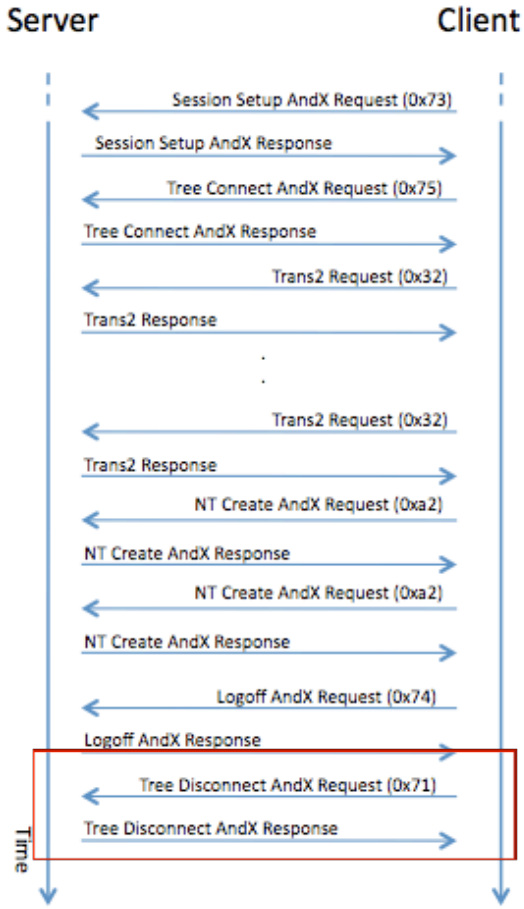
- Detailed measurement of distributed systems
- App. A = Parent transaction
- App. B = Child transaction of App. A
- Example
  - Runtime App. A: 2 Sec.
  - Runtime App. B: 5 Sec.
  - Total runtime: 7 Sec.

Developing the Prototype

- How to measure Samba features like mounting a network drive?
- Idea: Measuring SMB-Commands during their process time within Samba
  - Pro: Complete measurement of a SMB-Command from entering until leaving a SMB-Process
  - Contra: When accessing Samba and requesting a feature, a high number of SMB commands is exchanged (Overhead?)
- Identify specific SMB-Calls that mark the start and the end of a Samba feature
- Analysis of network traffic was required

Developing the Prototype
Network Analysis



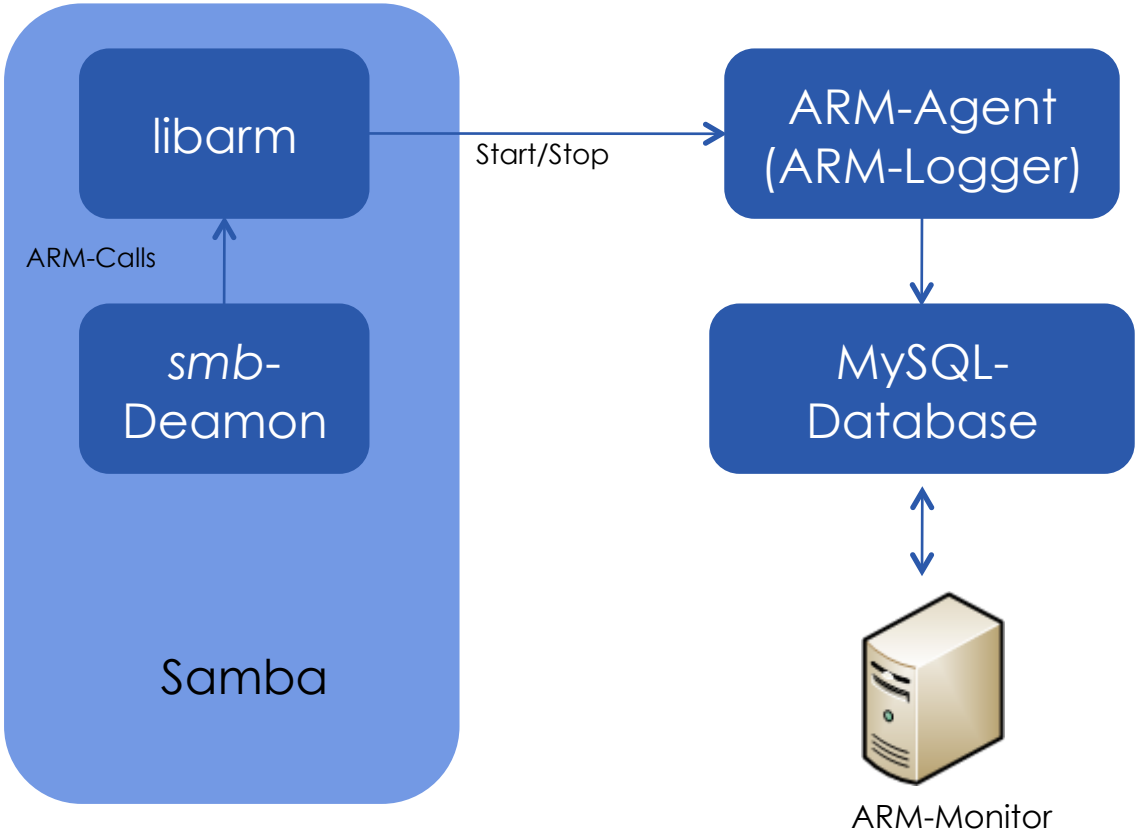


- Example: Mounting a network drive
- A **lot** of SMB Calls
- First idea was to measure all SMB Calls
  - Dropped that idea **very** fast!
- Start at *SMBNegprot*
- Stop at *SMBtdis*

Developing the Prototype
Extending Samba

- ARM-API-Calls had to be implemented in the „right place“ in the Samba source code
- Start with Samba main method
  - server.c (smbd_process method)
  - process.c (process_smb method)
- SMB Process starts a parent transaction
  - All further SMB Calls are handled as Child Transaction
- Additional information (e.g. Client IP address) had to be added to identify different Clients (more on that later)

Developing the Prototype Implementation



- Libarm knows about previous Client connections
- Check for a specific SMB Command in libarm (passed from smb-Deamon)
- Libarm decides when to send start/stop Calls to the ARM-Agent
- Why MySQL?
  - Storing data on a different Server
  - Independent from Analysis software

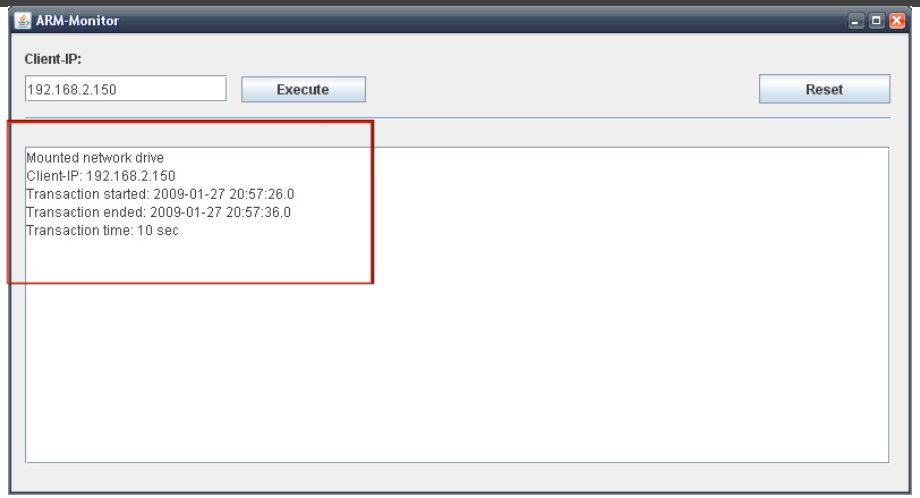
Developing the Prototype
Identifying different Clients

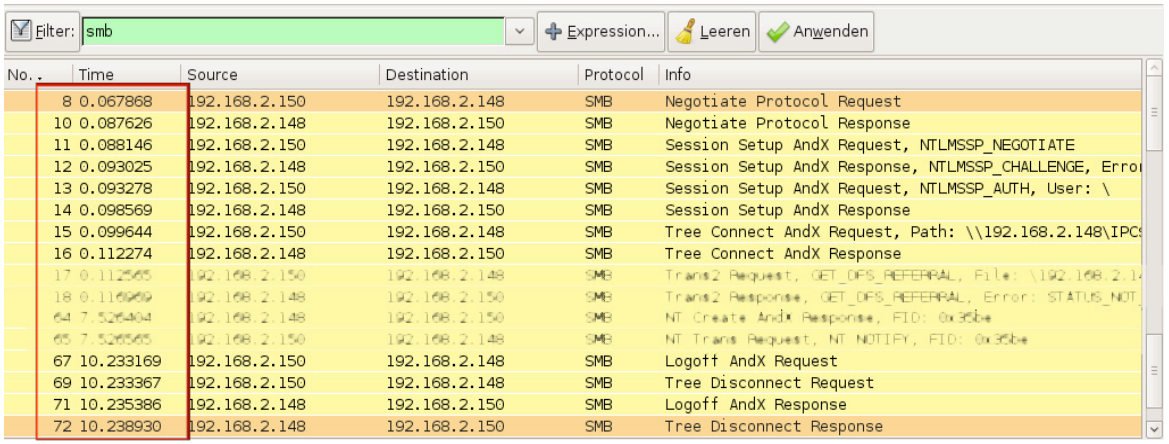
- ARM identifies each transaction with a unique ID
  - But how can one identify different Clients during analysis?
- Idea: grab as much information from Samba that we can
  - IP-Address
  - Username
  - PID
  - etc.
- Used for Prototype: IP-Address
  - Most suitable: **all** information

Results

- Prototype fulfilled our requirements
- Implementation requires little effort
  - Few lines of code and some modification for compiling
- Main Problem: Finding the correct SMB Calls which identify the start and the end of a specific Samba features
- ARM measurements correspond with network analysis
- More benefit comes when adding ARM-Calls to multiple Applications in your infrastructure

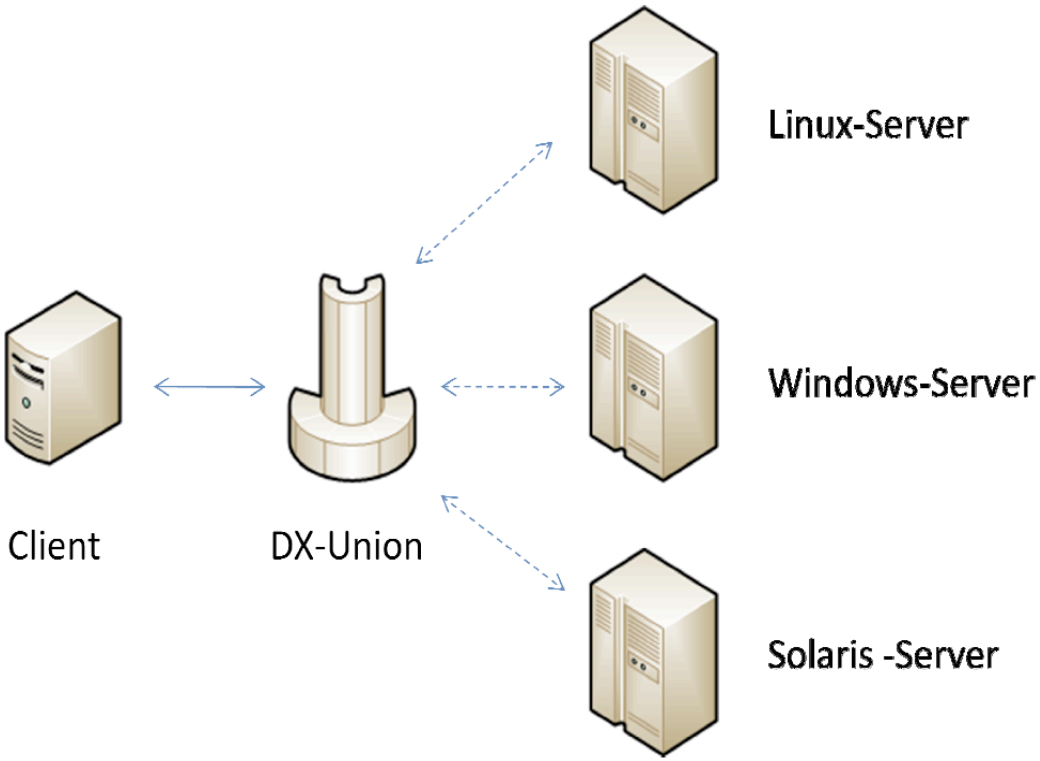
Results Measurements





- Java-GUI (Prototype) to access the database
- Query the database based on Client IP-Address
- Transaction time matches network traffic

Results
Benefit in distributed Systems



- DX-Union (e.g.)
  - Patch Management
  - License Management
  - Device Management

Lessons learned

- Fine-tuning required
  - Just analyzing SMB command is not sufficient enough for the measurement of a specific Samba features
  - A deeper look at the SMB Calls is required (e.g. Sub Commands)
- Further Analysis required
  - Scaling (Step three)
  - Overhead (Step three)

Thank you for your attention!

Questions?