



MCastor**2.0**

System Architecture Specification

MultiCastor 2.0

V 1.0 beta 7

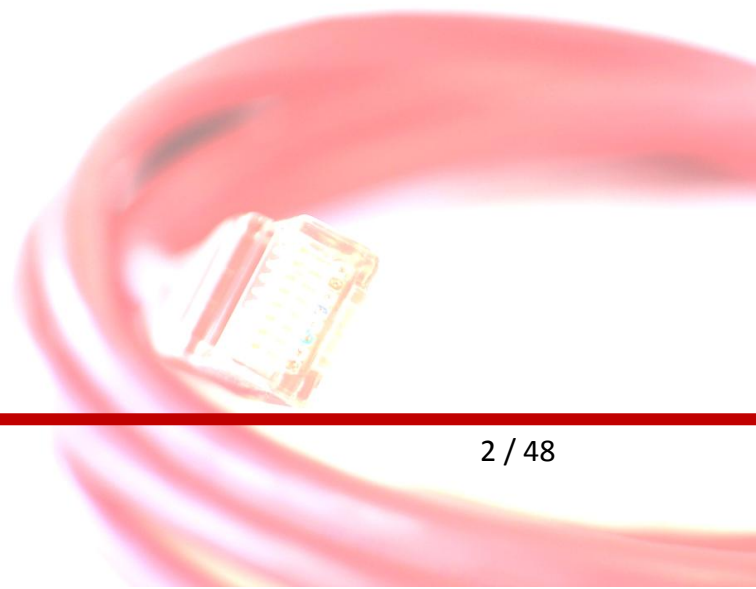
Projekt: MultiCastor 2.0

Auftraggeber: Rentschler & Stuckert
Rotebühlplatz 41/1
70178 Stuttgart

Auftragnehmer: TIT10AID - Team 4 - MCastor2.0

Fabian Fäßler
Filip Haase
Matthis Hauschild
Sebastian Koralewski
Jonas Traub
Christopher Westphal

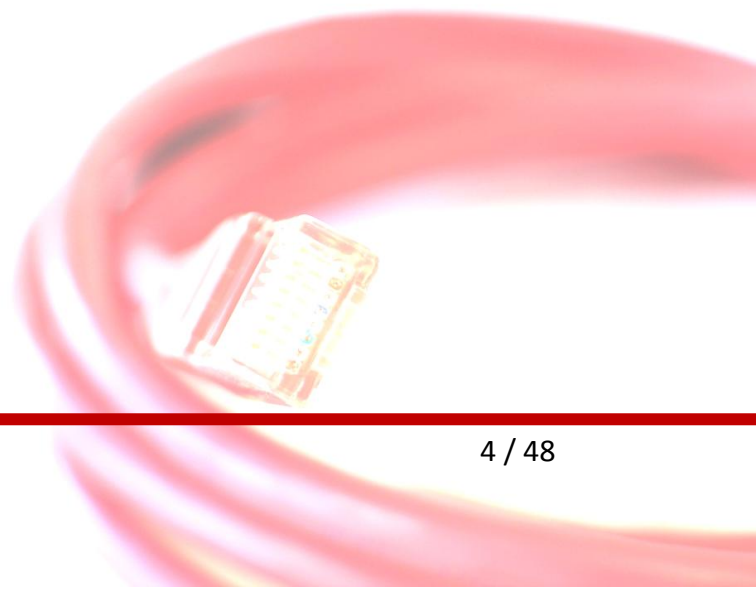
Rotebühlplatz 41 – Raum 0.10
70178 Stuttgart



Inhaltsverzeichnis

1. Einführung	5
2. Systemübersicht	6
2.1. Systemumgebung	6
2.2. Hardwareumgebung	6
2.3. Softwareumgebung	6
3. Architekturkonzept	7
3.1. (erweiterte) Architekturübersicht	7
3.1.1. Bestehende Programmstruktur	7
3.1.2. Funktionale Erweiterung um das MMR-Protokoll	9
3.1.3. Änderungen an der Konfiguration	9
3.1.4. Änderungen an der Benutzerschnittstelle	10
3.1.5. Programmtests mit STAF/STAX	10
3.2. Änderungen an der grafischen Oberfläche	11
3.2.1. Einführung	11
3.2.2. Aufbau Allgemein	11
3.2.3. Fenstertitel	12
3.2.4. Menü	13
3.2.5. Tableiste	15
3.2.6. Multicast-Steuerung	15
3.2.7. Multicast Konfiguration	17
3.2.8. Multicast Auswertung	18
3.2.9. Multicast-Anzeige	19
3.3. Änderungen an der Konfiguration	20
3.3.1. Programmkonfiguration	21
3.3.2. Userkonfiguration	21
3.3.3. Sprachkonfigurationsdateien	23
3.3.4. Manual-PDF	23
4. Integration eines neuen Protokolls – MMRP	24
4.1. Einführung in das Multiple MAC Registration Protokolls	24
4.1.1. Registrierung von Multicast-Pfaden	24

4.1.2.	Reservierung von Multicast-Pfaden	25
4.1.3.	Deregistrierung von Multicast-Pfaden	25
4.2.	Die Funktion des MMR-Protokolls.....	26
4.2.1.	Nachrichtentypen beim MMR-Protokoll.....	26
4.2.2.	Beschreibung der Zustandsautomaten	27
4.2.3.	Kommunikation – Verbindung vom Empfänger zu Sender herstellen	27
4.2.4.	Kommunikation – Verbindung vom Empfänger zum Sender trennen.....	28
4.3.	Systemdesign und Programmstruktur.....	28
4.3.1.	Zustandsautomat – Registrar state machine	29
4.3.2.	Zustandsautomat – Applicant state machine	31
4.3.3.	Zustandsautomat - LeaveAll state machine	34
4.3.4.	Zustandsautomat - Periodic Transmission state machine	35
4.3.5.	Das MMRP-Paket.....	37
4.4.	Systemdesign und Programmstruktur.....	39
5.	Integration in das Testframework STAF/STAX	40
5.1.	Einführung	40
5.2.	Technische Dokumentation.....	40
5.3.	Systemdesign und Programmstruktur.....	43
6.	Programmreaktion im Fehlerfall	44
6.1.	Grundlegendes	44
6.2.	Existierendes Problem Handling.....	45
6.3.	Gewolltes Verhalten	46
	Dokumentversionen	47



1. Einführung

Dieses Dokument dient der Spezifikation der Architektur des MultiCastors Version 2.0.

Da die V2.0 komplett auf V1.0 aufsetzt, bleibt die Basisarchitekturstruktur identisch. Deshalb ist das System in Untersysteme (Subsysteme) unterteilt und die Klassen, die die Subsysteme bilden, sind ebenfalls in diesem Dokument spezifiziert. Die Aufteilung wurde so durchgeführt, um Kohäsion zu maximieren und Kopplung zu minimieren.¹

Die "starke Fokussierung auf Modularität"² der Subsysteme erlaubt eine getrennte Entwicklung von neuen Modulen, wodurch die Arbeit im Team gut aufgeteilt werden kann.

Bei dem System kommt das Architekturmuster des Model View Controller zum Einsatz. Dieses beinhaltet die Strukturierung in die drei Einheiten Datenmodell (engl. model), Präsentation (engl. view) und Programmsteuerung (engl. controller). Ziel des Musters ist ein flexibler Programmentwurf, der eine spätere Änderung oder Erweiterung erleichtert und die Wiederverwendbarkeit der einzelnen Komponenten erhöht.³

Die Anbindung an das Testframework STAF/STAX, welche in V2.0 hinzukommt, ist eine eigene Säule in der Entwicklung, da dazu der MultiCastor nicht verändert werden muss, somit kann die Entwicklung komplett getrennt erfolgen.

Das MMR-Protokoll hingegen wird modular in den bestehenden MultiCastor integriert werden.

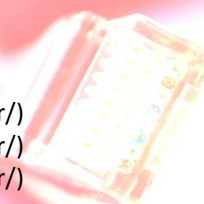
Einzig für die GUI müssen nennenswerte Eingriffe in bereits bestehende Module vorgenommen werden.

Zur Sicherstellung der Multibetriebssystemfähigkeit, wird der MultiCastor ständig auf Windows, Linux und Mac getestet. Um dem Anwender die unterschiedlichen „Look and Feel“ dieser Betriebssysteme näher zu bringen, haben wir uns entschlossen, bei den Screenshots in unseren Dokumenten verschiedene Systeme abzudecken.

¹ Quelle: SAS Multicastor V1.0 (<http://sourceforge.net/projects/multicastor/>)

² Quelle: SAS Multicastor V1.0 (<http://sourceforge.net/projects/multicastor/>)

³ Quelle: SAS Multicastor V1.0 (<http://sourceforge.net/projects/multicastor/>)



2. Systemübersicht

2.1. Systemumgebung

Der MultiCastor2.0 stellt einige Anforderungen an die Hardware- und Softwareumgebung. Bei der Entwicklung wurde darauf geachtet, dass der MultiCastor sowohl auf Windows als auch auf unixbasierten Betriebssystemen einwandfrei funktioniert. Außerdem muss das System über eine intakte Netzwerkverbindung verfügen, um Tests durchzuführen. Ausnahme stellt hierbei der Loopback-Modus dar, bei dem Sender und Empfänger auf dem gleichen Host ausgeführt werden.

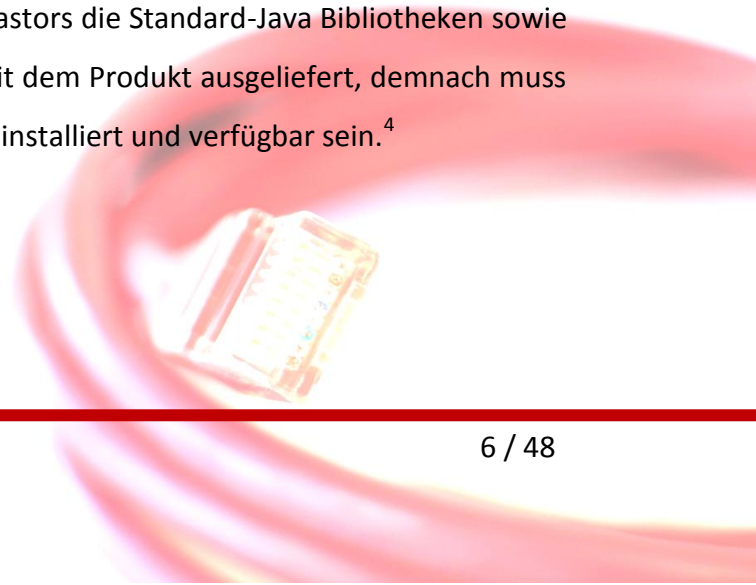
2.2. Hardwareumgebung

Die Hardware muss so gewählt werden, dass eine Java-Laufzeitumgebung auf Windows oder einem unixbasierten Betriebssystem installiert und dauerhaft betrieben werden kann. Die Hardware muss über eine Netzwerkschnittstelle verfügen, auf die über eine Softwareschnittstelle zugegriffen werden kann (Ausnahme stellt auch hier wieder der Loopback-Modus dar). Um MMRP-Anfragen bearbeiten zu können, wird ein MMRP-fähiger Switch benötigt.

2.3. Softwareumgebung

Der MultiCastor wurde und wird komplett in der Programmiersprache Java V1.6 umgesetzt. Es werden bei der Weiterentwicklung des MultiCastors die Standard-Java Bibliotheken sowie die JPCAP-Bibliothek verwendet. Letztere wird mit dem Produkt ausgeliefert, demnach muss nur eine JRE der Version 1.6 auf dem Hostsystem installiert und verfügbar sein.⁴

⁴ <http://www.java.com/de/download/help/sysreq.xml>



3. Architekturkonzept

3.1. (erweiterte) Architekturübersicht

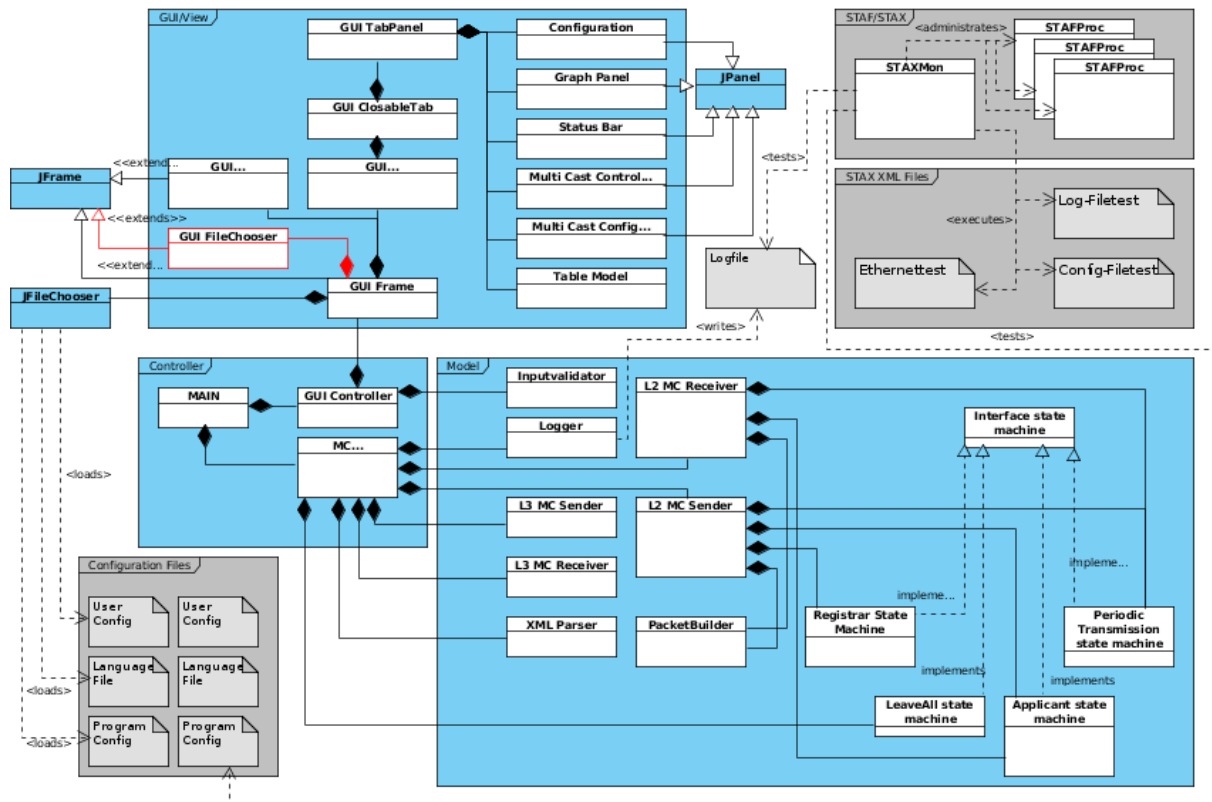


Abbildung 1: Architekturübersicht

3.1.1. Bestehende Programmstruktur

Die in Abbildung 1 gezeigte Programmstruktur setzt auf der Bestehenden des Multicastor 1.0 auf. Diese Struktur ist nach dem MVC-Modell designed. Damit wird Wartbarkeit, Testbarkeit und Möglichkeit zur Modularisierung gegeben.

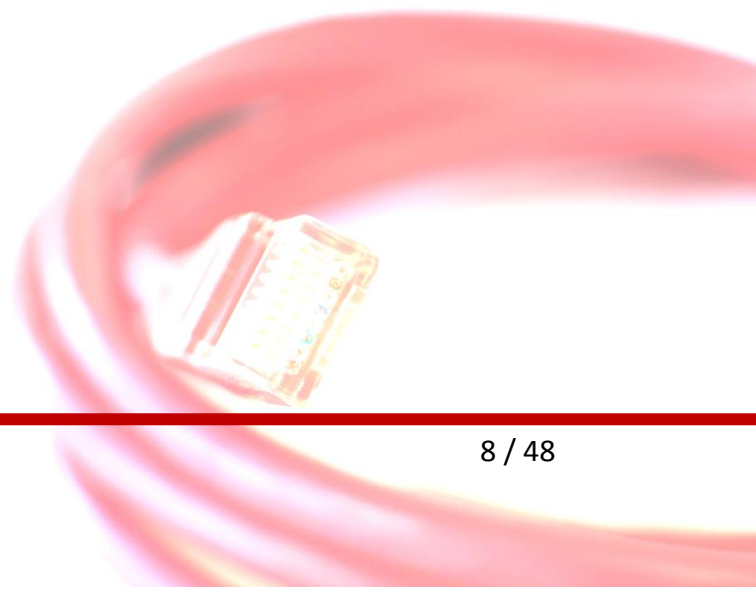
Der Programmstart erfolgt über die **Controller**. Dabei wird die *MAIN* Klasse gestartet, diese instanziiert immer den *MC Controller*, der alle Referenzen für die Logik hält. Wenn das Programm mit GUI gestartet wird, dann wird ebenfalls der *GUI Controller* von der *MAIN* Klasse instanziiert, der die Referenzen zu allen GUI Komponenten hält. Hier sieht man den Vorteil des MVC-Modells. Wenn der Multicastor ohne GUI gestartet wird, wird einfach auf die GUI Controller Klasse verzichtet.

Eine weitere Besonderheit sieht man, wenn man den Inputvalidator betrachtet. Diese Klasse ist dafür verantwortlich Eingaben (bisher IPv4 und IPv6 Adressen) zu validieren. Da diese Klasse zwar Programmlogik enthält, aber nur von der GUI genutzt wird, wird eine Referenz auf ein Objekt des *Inputvalidators* vom GUI Controller gehalten.

In Blick auf die **Model** Schicht sieht man die übernommen Klassen aus der bisherigen Struktur: XML Parser, Logger, Packetbuilder. Der *XML Parser* wird allgemein genutzt um XML zu parsen. Dies ist für die Config-Files relevant, da diese im XML-Format vorliegen. Die *Logger* Klasse dient allgemein zum Loggen der Ereignisse. Die *Packetbuilder* Klasse wird genutzt, um die Pakete, die versendet werden, protokollkonform zu erstellen.

Außerdem werden die bisherigen Sender/Receiver Klassen leicht abgeändert als L3 Sender/Receiver geführt (L3 für Layer 3 also IPv4 und IPv6).

Des Weiteren wurde die **View** Schicht aus dem bisherigen System übernommen und abgeändert. Allgemein gibt es hier aus der bisherigen Struktur das GUI Frame (die Klasse für das Hauptfenster des Multicastors) und die einzelnen Komponenten Configuration (Panel in der die Konfiguration vorgenommen wird), Graph Panel (der die Graphen darstellt), StatusBar (zeigt die Statusinformationen an) und MC_Control_Panel (für die Buttons zum Steuern).



3.1.2. Funktionale Erweiterung um das MMR-Protokoll

Eine zusätzliche Funktion, die im MultiCastor2.0 implementiert werden soll, ist die Unterstützung des MMRP-Protokolls. Um multicastfähige Switches mit dem MultiCastor2.0 zu testen, müssen Sender und Empfänger lediglich MMRP-Pakete senden bzw. empfangen. Die Verarbeitung der Pakete übernimmt der Switch.

Der MultiCastor2.0 soll jedoch auch in der Lage sein eine Back-to-Back Funktionalität zu unterstützen. Diese Anforderung verlangt, dass die einzelnen "state machines" in den MultiCastor2.0 implementiert werden müssen. Je nachdem, ob ein Sender oder Receiver angelegt wird, benötigt man verschiedene Zustandsautomaten. Der Receiver benötigt eine PeriodicTransmission state machine und eine Application-Only machine. Der Sender benötigt eine Registrar state machine, Applicant state machine und eine PeriodicTransmission state machine. Die LeaveAll state machine wird im MultiCastor Controller instanziiert, da sie generell alle Sender und Receiver gleichzeitig ansprechen muss. Die einzelnen Funktionen und Aufgaben der Zustandsautomaten werden im Abschnitt MMRP erläutert.

3.1.3. Änderungen an der Konfiguration

Wie in Abbildung 1 zu erkennen, wird der GUIFileChooser im MultiCastor2.0 nicht mehr enthalten sein. Diese Klasse stellte eine Erweiterung des JFileChoosers dar, welche die Auswahlcheckboxboxen für das partielle/inkrementelle Laden und Speichern enthielt. Da diese Funktionalitäten ohnehin unvollständig implementiert waren und die Ursache für Bugs darstellten, wurde beschlossen, folgende neue Struktur zu verwenden:

Um Darstellungsfehler bei der Dateiauswahl künftig zu vermeiden (insbesondere in Hinblick auf die Benutzung verschiedener Betriebssysteme), wird in der kommenden Version zunächst die Dateiauswahl mit einem (Standard) JFileChooser erfolgen. Dieser wiederum greift auf die Konfigurationsdateien im Dateisystem zu.

Die Funktionalität für inkrementelles sowie partielles Laden und Speichern wird nun in der Klasse GUIConfigChooser enthalten sein. Eine Instanz, dieser von JFrame erbenden Klasse, wird dem Benutzer nach der Dateiauswahl angezeigt.

Sowohl die Instanz des JFileChoosers als auch die des GUIConfigChoosers wird von einer JFrame-Instanz erzeugt und verwaltet.

3.1.4. Änderungen an der Benutzerschnittstelle

Wie in der obigen Abbildung zu erkennen, werden zwei neue Klassen ("GUI Closable Tab" und "GUI Draggable TabPane") implementiert, die weitere Funktionalitäten zur bereits bestehen Tab-Leiste hinzufügen und somit die Bedienung benutzerfreundlicher gestalten. Die Klasse "GUI Closable Tab" ermöglicht das Schließen jedes einzelnen Tabs. Mit Hilfe der Klasse "GUI Draggable TabPane" wird das Verschieben und Vertauschen einzelner Tabs ermöglicht.

Für die weiteren Änderungen an der Benutzerschnittstelle werden bereits vorhandene Klassen angepasst bzw. erweitert. In der Klasse "Multicast Control Panel" werden einige Änderungen vorgenommen, welche die Bedienung der Multicast-Ströme einfacher und intuitiver gestalten.

Die Klasse "Multicast Config Panel" wird vor allem in der Hinsicht erweitert, dass das MMR-Protokoll implementiert werden kann. Dies erfordert unter anderem die Eingabemöglichkeit einer MAC-Adresse statt einer - wie bisher nur möglich - IP-Adresse.

Des Weiteren wird die Eingabe so modifiziert, dass der MultiCastor2.0 automatisch erkennt, ob eine IPv4 oder eine IPv6 Adresse eingegeben wird, dies ermittelt das Programm im "GUI Controller", so dass sich der Benutzer bei der Eingabe nicht mehr darum kümmern muss.

Die Klasse "Graph Panel" wird auf Empfängerseite um einen so genannten Receive-Counter erweitert.

3.1.5. Programmtests mit STAF/STAX

Wie in Abbildung 1 zu sehen, besteht der STAF/STAX Teil aus mehreren Komponenten. STAFProc (STAF Process) sind Prozesse, die lokal oder im Netzwerk auf anderen Rechnern ausgeführt werden können. Wie im Abschnitt STAF/STAX noch genauer beschrieben, führt der STAXMon (STAX Monitor) anhand der STAX XML-Dateien auf den STAFProc Maschinen STAF Befehle aus. Wie beispielsweise das Lesen bzw. Testen der Config- sowie Log-Files und startet den MultiCastor2.0 ohne GUI. Es werden verschiedene STAX XML-Dateien, mit denen die verschiedenen I/O Kanäle vom MultiCastor2.0 getestet werden können, zur Verfügung gestellt.

3.2. Änderungen an der grafischen Oberfläche

3.2.1. Einführung

Im Vergleich zur Version 1.0 des MultiCastors, wird in der Version 2.0 besonderer Wert auf eine logischere und benutzerfreundlichere Bedienung gelegt. Einerseits wurde das Feedback der Benutzer analysiert und ausgewertet, um die Steuerung spürbar zu vereinfachen und das Arbeiten mit dem MultiCastor2.0 angenehmer zu gestalten. Andererseits wurden auch eigene Ideen und Verbesserungen eingebracht, die ebenfalls zu einer angenehmeren Bedienung beitragen.

Elemente der Benutzeroberfläche, die sich bewährt haben, werden in die Version 2.0 des MultiCastors übernommen.

3.2.2. Aufbau Allgemein

Die grafische Benutzeroberfläche baut auf einem normalen Frame auf und ist vom Betriebssystem abhängig. Das bedeutet, dass das Fensterdesign sich dem Design des verwendeten Betriebssystems anpasst. Die grobe Struktur und Anordnung der GUI-Elemente lehnt sich hauptsächlich an die Version 1.0 an und wurde an entsprechenden Stellen abgeändert oder weiterentwickelt.

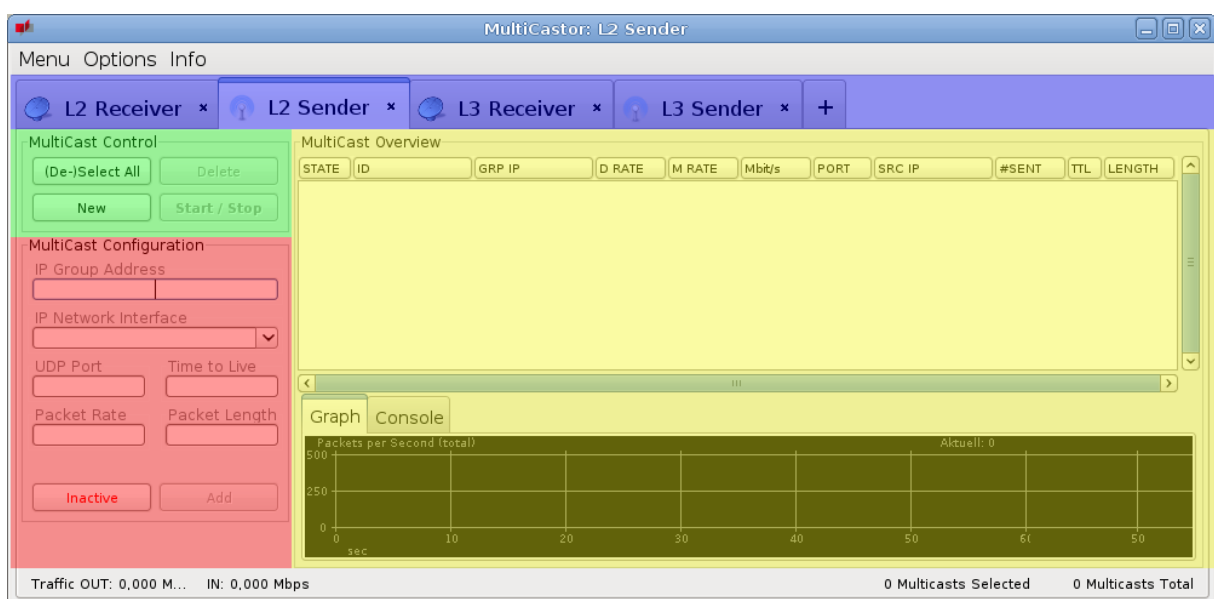


Abbildung 2: Neuer GUI-Aufbau

Das Frame beinhaltet mehrere Panels, welche die Steuerung logisch gruppieren und sinnvoll zusammenfassen (vier Hauptbereiche). Diese Bereiche sind in der obigen Abbildung farbig (blau, grün, gelb und rot) markiert und werden in den Folgeabschnitten noch genauer betrachtet.

3.2.3. Fenstertitel

Bisher existierte das Problem, dass man zwar mehrere Instanzen des MultiCastors starten konnte, jedoch der Fenstertitel immer gleich lautete. Sobald der Benutzer mehrere Instanzen geöffnet hatte, wurde es schnell unübersichtlich. Mit der Version 2.0 wird dieser Umstand beseitigt, indem sich der Fenstertitel automatisch danach richtet, in welchem "Tab" man sich befindet.

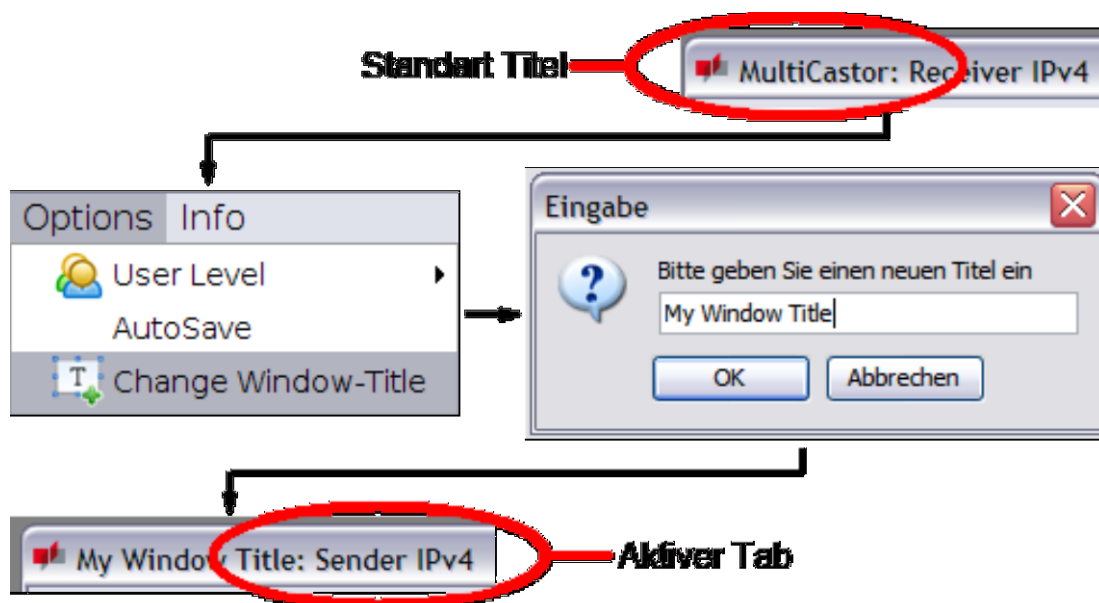


Abbildung 3: Unterschiedliche Fenstertitel

Eine Besonderheit stellt eine neue Funktion dar, die es dem Benutzer ermöglicht, sogar eigene individuelle Fenstertitel anzugeben. Dazu öffnet man das Options-Menü und klickt auf "Change Window-Title" (siehe Abbildung).

3.2.4. Menü

Das Menü besteht aus den drei Reitern "Menu", "Options" und "Info".

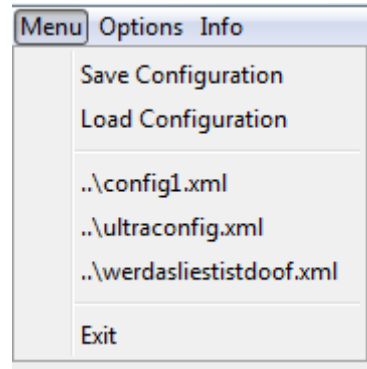


Abbildung 4: Menü⁵

Im Menü können Konfigurationsdateien geladen und gespeichert werden. Die Funktion, dass die drei zuletzt gespeicherten Konfigurationsdateien in einer Schnellauswahl angezeigt werden, wurde in der Version 2.0 beibehalten. Dadurch können Konfigurationsdateien mit einem Klick und ohne zusätzlichen Dateiauswahldialog geladen werden.

Wie aus den meisten anderen Programmen bekannt, kann auch der MultiCastor2.0 über das Menü ordnungsgemäß beendet werden.

Unter "Options" kann der Benutzer weitere Einstellungen vornehmen. Unter anderem wird dem Benutzer ermöglicht, einen eigenen Fenstertitel anzugeben (siehe Beschreibung in vorherigem Abschnitt).

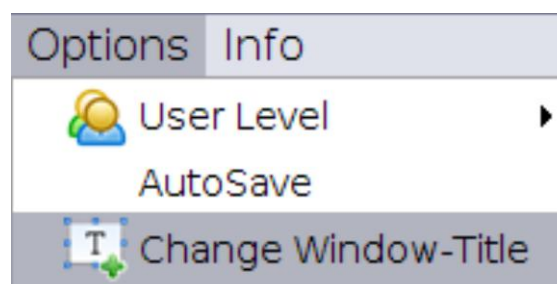


Abbildung 5: Optionen

Das "User-Level" ist eine Einstellmöglichkeit, die aus dem MultiCastor1.0 übernommen wurde. Hier kann der Benutzer die GUI nach seinen Bedürfnissen anpassen (Beginner/Expert). In der folgenden Tabelle⁶ ist zu erkennen, welche Elemente der GUI

⁵ Quelle: TIT09AIB_System_Architecture_Specification – Abbildung 3 Menu

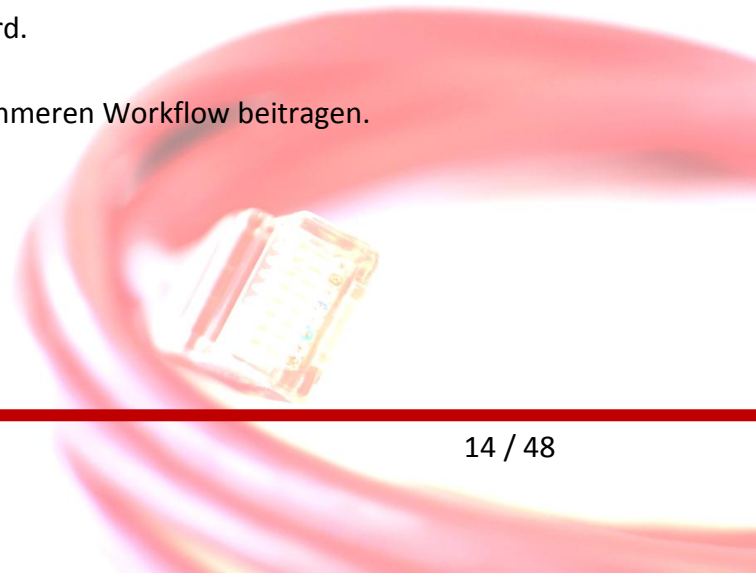
⁶ Quelle: TIT09AIB_System_Architecture_Specification – Tabelle 3 Einstellung Benutzererfahrung

sichtbar sind und welche nicht (grün = sichtbar; rot = nicht sichtbar; gelb = nicht editierbare Standardwerte):

Beginner	Expert
Steuerung – Start Button	Steuerung – Start Button
Steuerung – Stop Button	Steuerung – Stop Button
Steuerung – New Button	Steuerung – New Button
Steuerung – Select All Button	Steuerung – Select All Button
Steuerung – Delesect All Button	Steuerung – Delesect All Button
Steuerung – Delete Button	Steuerung – Delete Button
Konfiguration – Group IP	Konfiguration – Group IP
Konfiguration – Source IP	Konfiguration – Source IP
Konfiguration – Port	Konfiguration – Port
Konfiguration – Packet Length	Konfiguration – Packet Length
Konfiguration – TTL	Konfiguration – TTL
Konfiguration – Paket Rate	Konfiguration – Paket Rate
Konfiguration – Active	Konfiguration – Active
Konfiguration – Enter	Konfiguration – Enter
Spalte - State	Spalte – State
Spalte - ID	Spalte – ID
Spalte - Group IP Address	Spalte - Group IP Address
Spalte - Source IP Address	Spalte - Source IP Address
Spalte - Port	Spalte – Port
Spalte - Desired Packet Rate	Spalte - Desired Packet Rate
Spalte - Measured Packet Rate	Spalte - Measured Packet Rate
Spalte - Time To Live	Spalte - Time To Live
Spalte - Packet Length (bit)	Spalte - Packet Length (bit)
Spalte - Total Packets Sent	Spalte - Total Packets Sent
Spalte - Number of Interruptions	Spalte - Number of Interruptions
Spalte - Average Interruption Time	Spalte - Average Interruption Time
Spalte - Packet Loss per Second	Spalte - Packet Loss per Second

Bei der Option "AutoSave" kann der Benutzer festlegen, ob das automatische Speichern im Hintergrund aktiviert werden soll. Automatisches Speichern bedeutet in diesem Fall, dass bei jeder Änderung der User-Konfiguration – sei es ein Löschen, Einfügen oder Updaten – die neu entstandene Konfiguration in eine XML-Datei gespeichert wird, die beim nächsten starten des MultiCastors automatisch geladen wird.

Dies kann in bestimmten Fällen zu einem angenehmeren Workflow beitragen.



3.2.5. Tableiste

Die Tableiste wurde beim MultiCastor2.0 umgebaut und komfortabler gestaltet.



Abbildung 6: Neue Tableiste

Die alten IPv4 -und IPv6-Tabs werden in der Version 2.0 vollständig durch Sender -und Receiver-Tabs (jeweils Layer-2 und -3) ersetzt. Hintergrund ist hierbei, dass die Version 2.0 das MMR-Protokoll unterstützt.

Wie auf der obigen Abbildung ersichtlich, können Tabs jederzeit ausgeblendet werden, indem man auf das "X" hinter dem Namen des Tabs klickt. Bei Bedarf können Tabs natürlich auch wieder eingeblendet werden. Dazu klickt man auf das "+" am Ende der Tab-Auflistung (siehe roter Pfeil in der obigen Abbildung).

3.2.6. Multicast-Steuerung

Mit Hilfe der Multicast-Steuerung werden die Sender und Empfänger gesteuert und verwaltet. Die Steuerung kann für beide – Sender als auch Empfänger – verwendet werden.

Im Vergleich zur Version 1.0 des MultiCastors wird in der Version 2.0 ein besonderer Wert auf eine benutzerfreundlichere Steuerung gelegt. Die Menge der Buttons wird reduziert, indem Buttons mit ähnlichen Funktionalitäten zusammengelegt werden. Des Weiteren wird die Anordnung der Buttons logischer und intuitiver gestaltet.

So wird beispielsweise der "Start / Stop"-Button optisch hervorgehoben und rechts unten platziert, da Buttons dieser Art (z.B. "Weiter", "Start", ...) sich in den meisten anderen Programmen ebenfalls rechts unten etabliert haben. Der Benutzer findet sich also schneller zurecht.

Da das Auswählen von Einträgen und das Löschen oft direkt nacheinander ausgeführt werden, werden beide Buttons direkt nebeneinander angeordnet. Der Benutzer agiert hierbei also lediglich in der ersten "Buttonzeile".

Des Weiteren werden die Buttons "Select All" und "Deselect All" zusammengelegt, da das Programm beim Klick auf den Button "(De-)Select All" selbstständig entscheiden kann, ob es Sinn ergibt, alles auszuwählen oder ob die bestehende Auswahl zu entfernen ist.

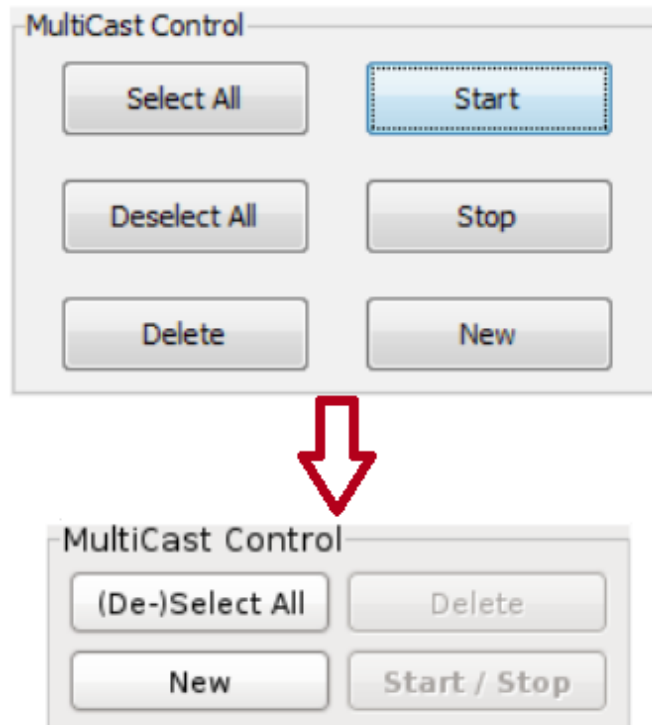


Abbildung 7: Vergleich alte vs. neue MC-Steuerung

Im Folgenden werden die Funktionen der Multicast-Steuerung beschrieben:

- "(De-)Select All": Markiert oder entfernt die Markierung aller vorhandenen Einträge in der Multicast-Übersicht
- "Delete": Löscht die ausgewählten Einträge in der Multicast-Übersicht
- "New": Löscht alle Eingaben aus dem oben genannten Multicast-Konfigurationspanel und hilft dem Benutzer durch die Verwendung von Tooltips bei der Eingabe der neuen Werte
- "Start / Stop": Je nachdem, ob ein ausgewählter Eintrag bereits aktiviert oder deaktiviert ist, wird der Eintrag aktiviert oder deaktiviert. Wird ein Eintrag aktiviert, bedeutet dies, dass der entsprechende Sender / Empfänger gestartet wird. Wird ein Eintrag deaktiviert, bedeutet dies, dass beim Sender das Senden deaktiviert wird. Beim Empfänger werden alle markierten und abonnierten Multicasts abbestellt.

3.2.7. Multicast Konfiguration

Das Konfigurationspanel wird aus Version 1.0 übernommen. Es wird jedoch erweitert, so dass es auch die Eingabe einer MAC-Adresse bei Bedarf verlangt, da diese für das MMR-Protokoll benötigt wird.

Das Tool prüft bereits bei der Eingabe, ob korrekte Werte in die entsprechenden Felder eingegeben wurden. Sind die Werte korrekt, wird das Feld grün umrandet; andernfalls rot.

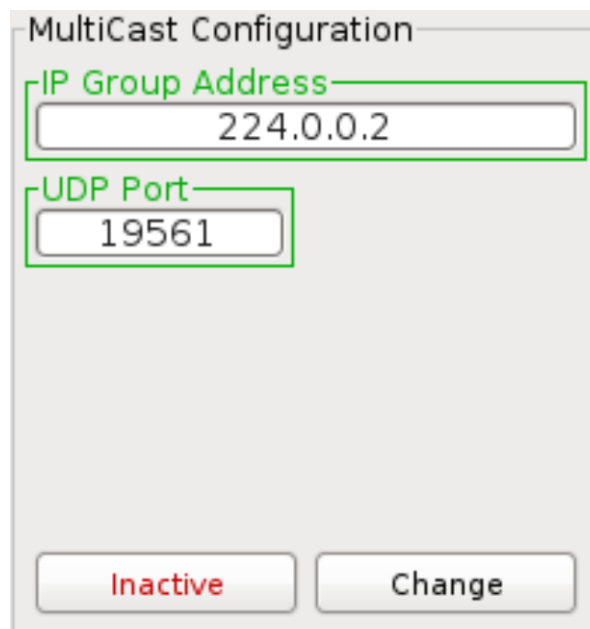


Abbildung 8: Multicast Empfänger (Layer-3)

Jedes Textfeld besitzt außerdem noch einen eigenen so genannten „Tooltip“, der eingeblendet wird, wenn der Benutzer länger als eine Sekunde mit dem Mauszeiger über dem entsprechenden Textfeld bleibt. Dieser "Tooltip" liefert Informationen über akzeptierte Eingabewerte.

3.2.8. Multicast Auswertung

Die Auswertung der Multicasts wird in zwei verschiedenen Bereichen jeweils beim Sender und Empfänger angezeigt. Die Anzeige der Auswertung kann vollständig und ohne größere Änderungen aus der Version 1.0 übernommen werden.

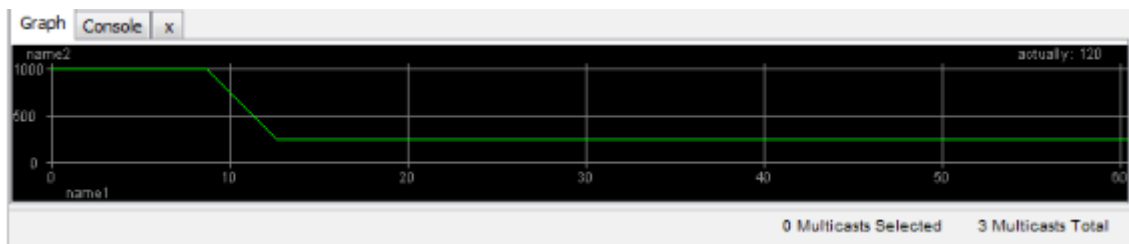


Abbildung 9: Ansicht des Graphen

Der Sinn des Graphen ist es, die gewonnenen Werte optisch aufzubereiten und übersichtlich darzustellen. Dazu wird eine Graphenklasse erstellt, die einen Graphen zeichnet. Die Klasse erweitert die Java Klasse JPanel. Das Panel ist in die grafische Benutzeroberfläche integriert.

Für den Graphen wird es keine Scrollleisten geben, der Graph muss also bei jedem Fenster-Resize neu und in der Größe angepasst gezeichnet werden.

Folgende Graphen werden implementiert:

Funktion	Graph	Beschreibung
Sender	Total Packet Rate	Stellt die Paketrage über Zeit dar
	Jitter	Stellt die Jitter-Werte über Zeit dar
Empfänger	Lost Packets	Stellt die verloren gegangenen Pakete über Zeit dar
	Measured Packet Rate	Stellt die gemessene Paket Rate über Zeit dar

3.2.9. Multicast-Anzeige

Die Anzeige der Multicast Empfänger und Sender wird vollständig aus der Version 1.0 übernommen, da sie sich in dieser Form bewährt hat. Die tabellarische Ansicht wird jedoch noch um eine Spalte "MAC Address" erweitert, da die Version 2.0 des MultiCastors das MMR-Protokoll unterstützt.

MultiCast Overview										
STATE	ID	GRP IP	D RATE	M RATE	Mbit/s	PORT	LOSS/S	AVG INT	#INT	SRC
<input type="checkbox"/>	-1	224.0.0.1	0	0	0,000	19561	-1	0	0	0 UNDEFINED
<input type="checkbox"/>	-1	224.0.0.1	0	0	0,000	18512	-1	0	0	0 UNDEFINED

Abbildung 10: Multicast-Anzeige

Die Einträge in der Multicast Anzeige können wahlweise direkt über die jeweilige Zeile oder über die Multicast Steuerung angesprochen werden.

Sender und Empfänger können demnach über die Checkbox am Anfang der Zeile gestartet oder gestoppt werden. Ist ein Empfänger oder Sender gestoppt, können die Attribute bearbeitet werden, indem die entsprechende Zeile ausgewählt wird. In der Multicast-Konfiguration können dann die Werte geändert und aktualisiert werden.

Tabellenspalten Sender	Tabellenspalten Empfänger
State	State
ID	ID
Group IP Address	Group IP Address
Source IP Address	Source IP Address
Port	Port
Desired Packet Rate	Desired Packet Rate
Measured Packet Rate	Measured Packet Rate
Time To Live	Number of Interruptions
Packet Length (bit)	Average Interruption Time
Total Packets Sent	Packet Loss per Second

„Wie in der obigen Tabelle zu sehen, sind die Tabellenspalten für Sender und Empfänger sehr ähnlich. Daher wird auch hier ein gemeinsames Tabellenmodell verwendet, welches jeweils für Sender / Receiver und Layer-2 / Layer-3 unterschieden und angepasst wird.“

Die Tabelle bietet die Möglichkeit, mehrere Einträge zu markieren und zu bearbeiten. Die Einträge können auf drei unterschiedliche Weisen markiert werden.

Die einzelne Auswahl eines Senders oder Empfängers erlaubt die Änderung aller Parameter über das Konfigurationspanel.

Eine Mehrfachauswahl von Sendern oder Empfängern erlaubt ebenfalls eine Änderung aller Parameter mit Ausnahme der Multicast Adresse, der Source Adresse und des Ports, da diese nicht bei mehreren Sendern gleich sein können.

Keine Auswahl führt dazu, dass der User im Konfigurationspanel neue Multicasts hinzufügen kann.

Bei Einfach- und Mehrfachauswahl ist es zudem möglich, die Sender und Empfänger über das Control Panel zu starten, zu stoppen und zu löschen.“⁷

3.3. Änderungen an der Konfiguration

Die Konfiguration im MultiCastor2.0 soll modularer gestaltet werden. Um dies zu erreichen, wird es folgende Konfigurationsdateien geben:

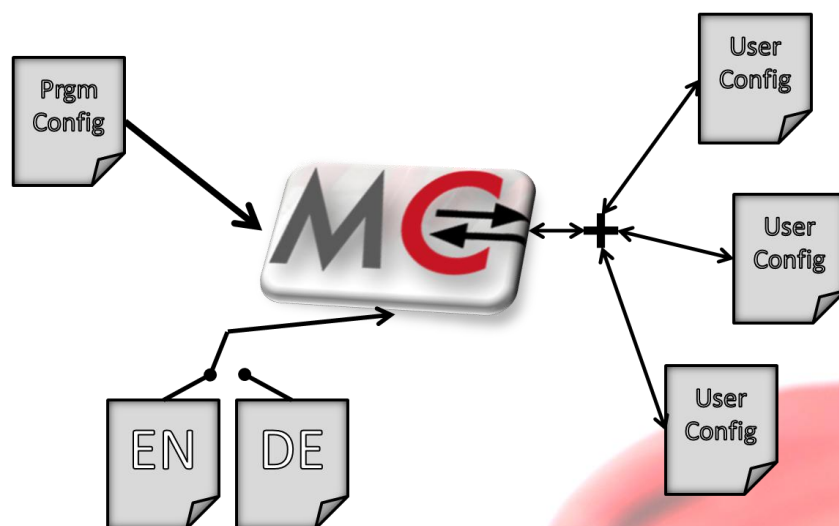


Abbildung 11: verschiedene Konfigurationsdateien

⁷ Quelle: TIT09AIB_System_Architecture_Specification – 2.2.4 Multicast Anzeige & Tabelle 1

3.3.1. Programmkonfiguration

Die Programmkonfiguration wird in einer XML⁸-Datei gespeichert. Sie umfasst folgende Punkte:

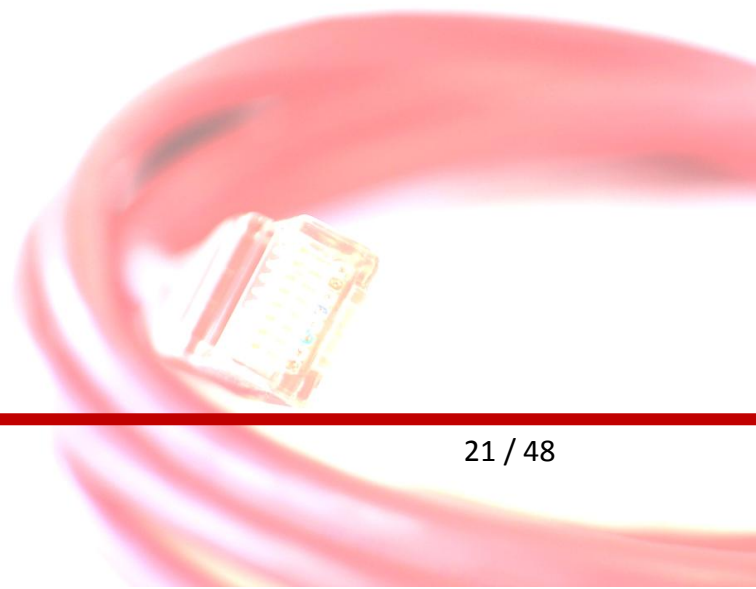
- ausgewählte Sprache [en / de / ...]
- automatisches Speichern [an / aus]
- ausgewähltes Userlevel [Beginner / Expert]
- Fensternamen [Multicastor / eigene Eingabe]
- standardmäßig aktiver Tab [L2 Sender / L2 Receiver / L3 Sender / L3 Receiver]
- Standartwerte für das Hinzufügen eines neuen Senders / Receivers
 - L2 Sender Standartwerte
 - L2 Receiver Standartwerte
 - L3 Sender Standartwerte
 - L3 Receiver Standartwerte
- Reihenfolge der Spalten

3.3.2. Userkonfiguration

Auch Userkonfigurationen werden in XML-Dateien gespeichert. Diese umfassen folgende Punkte:

- L2 Sender
 - Multicast-MAC-Adresse
 - Netzwerkinterface
 - Paketlänge
 - gewünschte Datenrate
- L2 Receiver

⁸ siehe Glossar



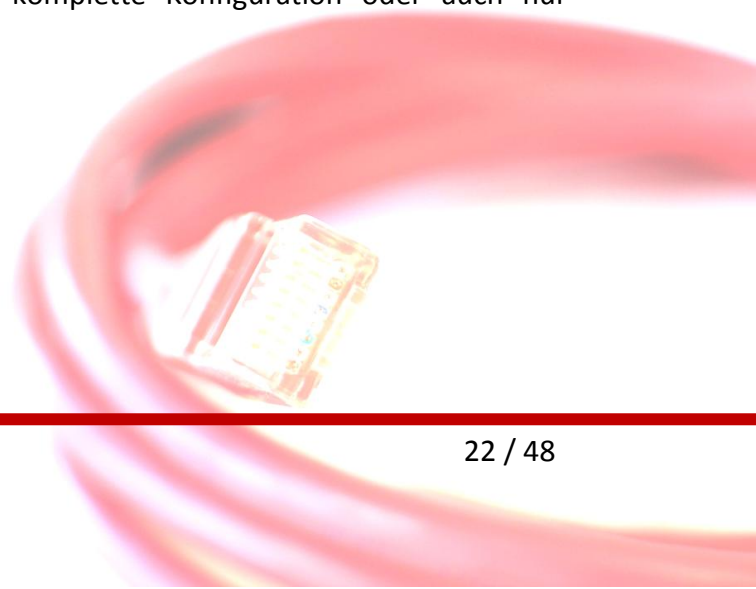
- Multicast-MAC-Adresse
 - Netzwerkinterface
- L3 Sender
 - Multicast-IP-Adresse
 - Netzwerkinterface
 - Port
 - Paketlänge
 - TTL⁹
 - gewünschte Datenrate
- L3 Receiver
 - Multicast-IP-Adresse
 - Port
- PC-Kennung und Speicherzeitpunkt

Es soll ein inkrementelles und partielles Laden sowie ein partielles Speichern ermöglicht werden.

Beim Laden bedeutet dies, dass der Nutzer entweder eine ganze Konfigurationsdatei laden, oder aber auch nur Teile der Datei auswählen kann. Des Weiteren kann er inkrementell aus mehreren Dateien hintereinander laden, ohne dass vorher geladene Konfigurationen verloren gehen.

Ebenso kann der Nutzer beim Speichern die komplette Konfiguration oder auch nur Teilespeichern.

⁹ siehe Glossar



3.3.3. Sprachkonfigurationsdateien

Es wird für jede Sprache jeweils eine Sprachdatei geben. Es werden eine englische und eine deutsche Sprachdatei zu Verfügung gestellt. Der Nutzer hat die Möglichkeit, weitere Sprachdateien zu erstellen und im MultiCastor zu laden.

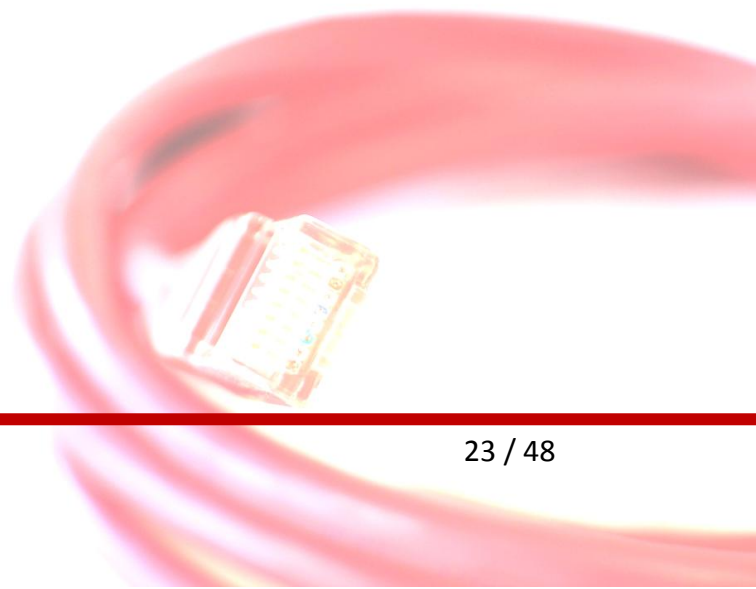
Wenn eine neue Sprache eingestellt wird, wird eine Warnung ausgegeben, wenn gerade ein Multicast-Strom aktiv ist, denn um die Sprache zu ändern, muss der Multicastor neugestartet werden.

Wie eine Sprachdatei zu erstellen ist, ist dem mitgelieferten Manual zu entnehmen.

3.3.4. Manual-PDF

Die Beschreibung, sowie die Konfiguration des MultiCastors ist dem Manual-PDF zu entnehmen, welches direkt aus dem MultiCastor (unter Info->Help) geöffnet werden kann.

Hinsichtlich der Multilingualität wird das Manual von uns in englischer und deutscher Ausführung zur Verfügung gestellt. Der MultiCastor öffnet anhand der eingestellten Sprache das entsprechende PDF. Wenn eine Sprache eingestellt ist, zu der es kein Manual gibt, wird das englische Manual angezeigt.



4. Integration eines neuen Protokolls – MMRP

In diesem Abschnitt wird das MMR-Protokoll dargestellt. Es wird erklärt, welche Aufgaben das Protokoll zu erledigen hat, wie die Kommunikation abläuft, wie die einzelnen Switches aufgebaut werden müssen, um MMRP zu unterstützen und wie der Aufbau der Zustandsautomaten und des MMRP-Paketes aussieht.

4.1. Einführung in das Multiple MAC Registration Protokolls

Mit Hilfe des MMR-Protokolls sollen die Switche im Netzwerk selbstständig Multicast betreiben. Da Switche nur auf der Layer-2-Ebene kommunizieren können, muss das Protokoll auch dort anknüpfen. Die folgenden Absätze werden darstellen, welche Aufgabe MMRP bewältigen muss, um Multicast zu betreiben.

4.1.1. Registrierung von Multicast-Pfaden

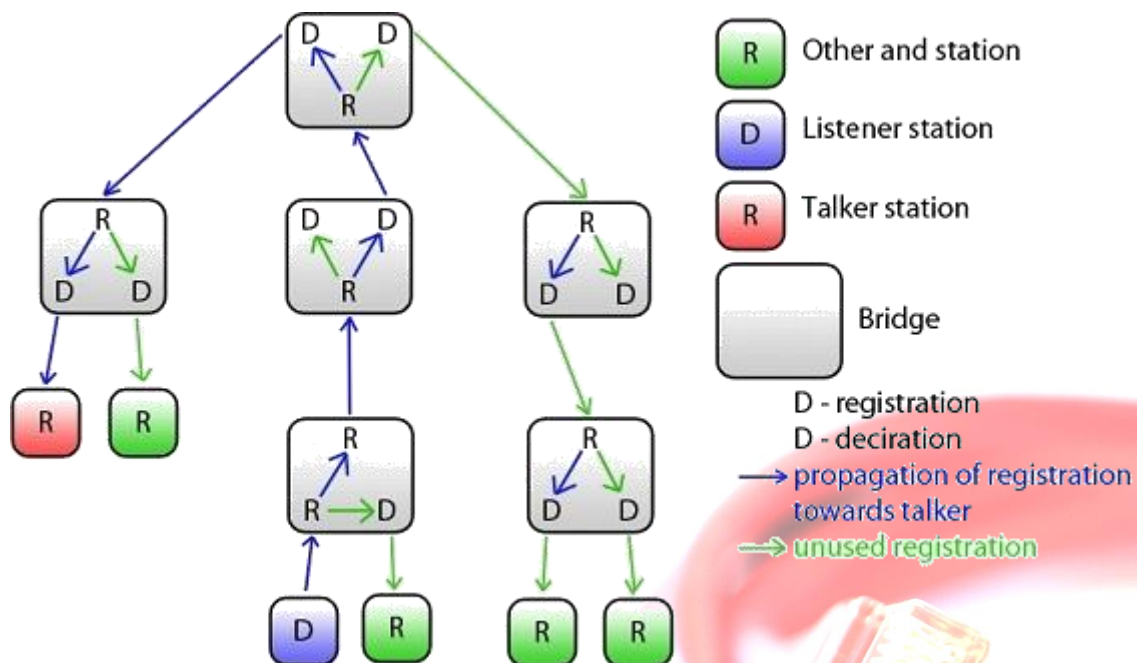


Abbildung 12: Registrierung von MC-Pfaden

Jeder einzelne Switch muss in der Lage sein, den Empfänger mit dem Sender verbinden zu können, unabhängig davon, wie viele Zwischenstationen vorhanden sind. Erst bei einem registrierten Pfad vom Empfänger zum Sender kann ein erfolgreiches Multicasting stattfinden.

4.1.2. Reservierung von Multicast-Pfaden

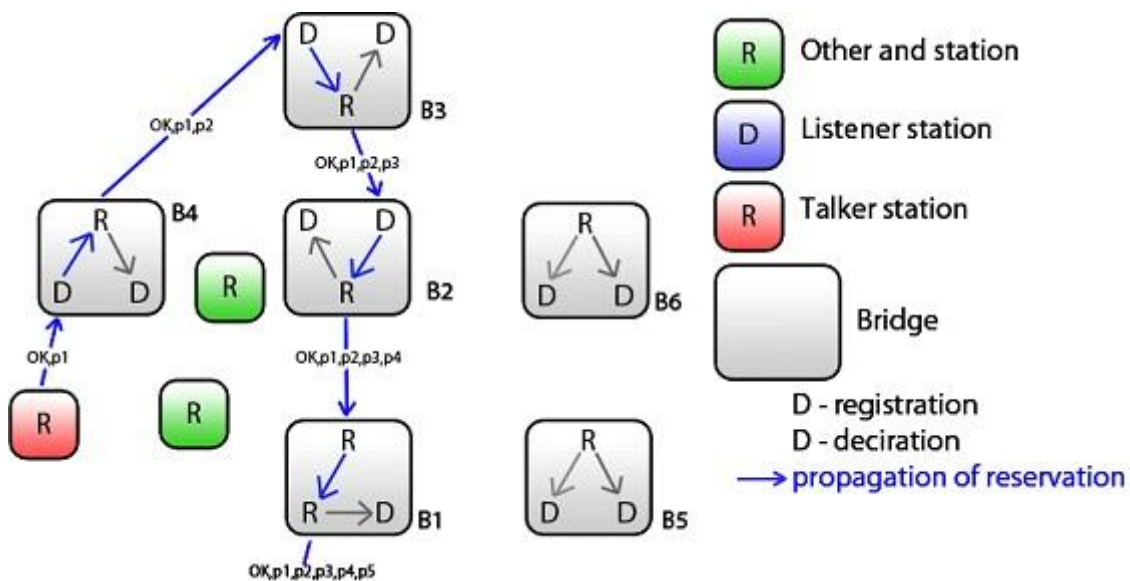


Abbildung 13: Reservierung von MC-Pfaden

Eine erfolgreiche Registrierung alleine, ermöglicht noch keinen Multicast. Der Empfänger muss seine angeforderten Daten erhalten können. Dieser Vorgang wird als Reservierung der Multicast-Pfade bezeichnet. Die Switches haben die Ports gespeichert, durch die die Empfänger verbunden sind. Beim Senden der Multicast-Pakete werden die reservierten Pfade genutzt, um die Pakete zu den Empfängern zu übertragen.

4.1.3. Deregistrierung von Multicast-Pfaden

Die Switches müssen ebenfalls in der Lage sein, die registrierten Pfade wieder freizugeben. Dies kann zwei verschiedene Gründe haben:

- Der Empfänger möchte die Daten nicht mehr empfangen.
- Der Sender löst seine Multicast Gruppe auf.

Das Freigeben von Multicast-Pfade bezeichnet man auch als *Source-Pruning*.

4.2. Die Funktion des MMR-Protokolls

Hier wird beschrieben, wie MMRP funktioniert. Es wird betrachtet, wie ein Switch aufgebaut sein muss, um MMRP zu unterstützen, wie die Zustandsautomaten untereinander kommunizieren und wie die Kommunikation abläuft, wenn man einen Pfad registrieren, reservieren oder freigeben will.

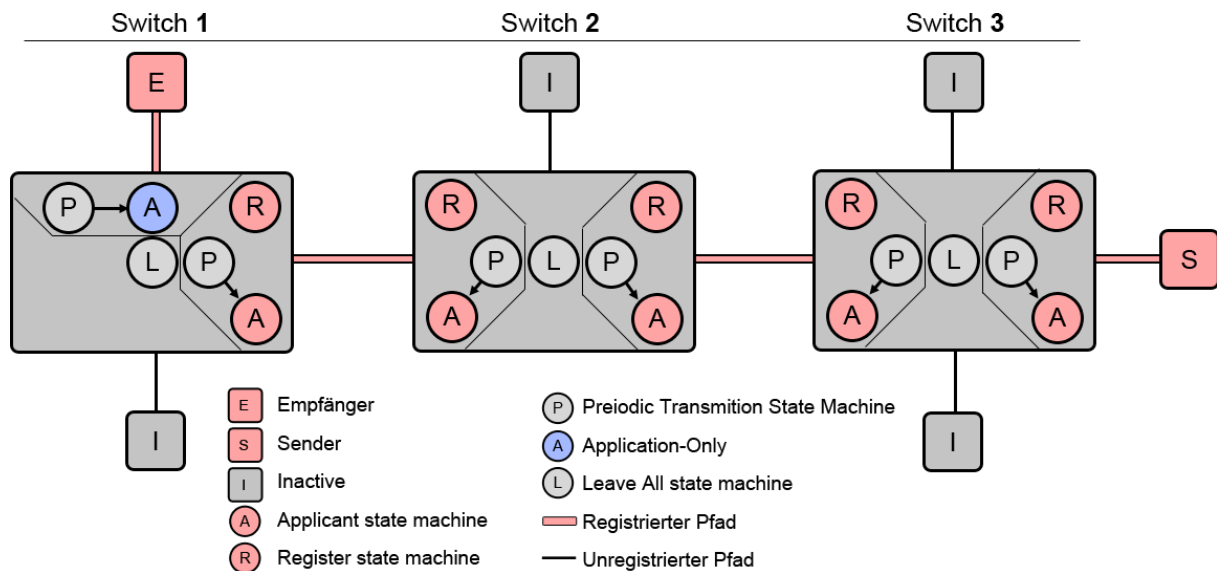


Abbildung 14: MMRP-Kommunikation zwischen mehreren Switches

4.2.1. Nachrichtentypen beim MMR-Protokoll

Es gibt 6 Nachrichtentypen, die im MMRP definiert sind. Jeder Nachrichtentyp erfüllt seine eigene Funktion.

1. JoinIn (*Applicant* deklariert und registriert)
2. In (*Applicant* **nicht** deklariert und registriert)
3. JoinEmpty (*Applicant* deklariert und **nicht** registriert)
4. Empty (*Applicant* **nicht** deklariert und **nicht** registriert)
5. Leave (*Applicant* deregistrieren)
6. LeaveAll (Alle *Applicants* deregistrieren)

4.2.2. Beschreibung der Zustandsautomaten

Registrar state machine: Die *Registrar state machine* verwaltet die registrierten MAC-Adressen. Pro Multicast-Strom, der gesendet wird, benötigt man eine *Registrar state machine*. Ein Switch, der einen Multicast Strom nur empfängt, benötigt keine *Registrar state machine* zu initialisieren.

Applicant state machine: Die *Applicant state machine* wird genutzt, um die Kommunikation im Netzwerk herzustellen. Die *Applicant state machine* versendet Nachrichten ins Netzwerk und stellt die Kommunikation zwischen *Applicant state machine* und/oder *Registrar state machine* her. Pro Multicast-Strom, der gesendet bzw. empfangen wird, benötigt man eine *Applicant state machine*.

Application-Only: Als *Application-Only* bezeichnet man eine *Applicant state machine*, die nur Multicastströme empfängt, sie jedoch nicht sendet.

LeaveAll state machine: Die *LeaveAll state machine* versendet in bestimmten Zeitpunkten LeaveAll Nachrichten, um unbenutzte Multicastströme zu entfernen. Jeder Switch benötigt nur eine *LeaveAll state machine*.

PeriodicTransmission state machine: Eine *PeriodicTransmission state machine* sendet an die dazu gehörigen *Applicant state machine* Nachrichten, dass die *Applicant state machine* aktive JoinIN Nachrichten verschicken soll. Dies soll einerseits zur Überprüfung dienen, ob der Multicaststrom vorhanden ist und andererseits ist bei Netzwerkänderungen garantiert, dass die Verbindung schnell hergestellt wird.

4.2.3. Kommunikation – Verbindung vom Empfänger zu Sender herstellen

1. *Application-Only* vom Switch 1 schickt eine JoinIn Nachricht an die *Applicant state machine* am Port 2 in Switch 1.
2. Die *Registrar state machine* am Port 2 registriert die MAC-Adresse vom Host und teilt der *Applicant state machine* mit, dass eine MAC-Adresse registriert wurde.

3. Die *Applicant state machine* sendet daraufhin eine JoinIn Nachricht an den *Application-Only* zurück.
4. Der *Application-Only* sendet als Bestätigung eine JoinIn Nachricht zurück.
5. Die *Applicant state machine* führt dieselbe Art von Bestätigung nochmals aus. Die erste Verbindung wurde erfolgreich hergestellt.
6. Um Switch 1 mit Switch 2 zu verbinden, müssen die zwei *Applicant state machines* ebenfalls eine Verbindung herstellen. Die Verbindungsherstellung funktioniert genauso wie in Schritt 1 bis 5 beschrieben.
7. Dieser Vorgang zieht sich durch bis die *Applicant state machine* vom Sender verbunden ist. Damit ist die Verbindung vom Empfänger zum Sender hergestellt.

4.2.4. Kommunikation – Verbindung vom Empfänger zum Sender trennen

1. *Application-Only* vom Switch 1 schickt eine Leave Nachricht an die *Applicant state machine* am Port 2 in Switch 1.
2. Da keine JoinIn Nachricht vom *Application-Only* mehr kommt, deregistriert die *Registrar state machine* die MAC-Adresse. Daraufhin schickt die *Applicant state machine* ebenfalls eine Leave-Nachricht an Switch 2.
3. Dieser Vorgang wiederholt sich, bis sie beim Sender ankommt. Die Verbindung wurde erfolgreich aufgelöst.

4.3. Systemdesign und Programmstruktur

In diesem Abschnitt werden die einzelnen Zustandsautomaten dargestellt. Es wird erläutert, wie sie aufgebaut sind und wie sie auf bestimmte Events reagieren. Des Weiteren wird der Aufbau des MMRP-Pakets dargestellt. Hier wird verdeutlicht, wie sich der Aufbau einer JoinIn Nachricht von beispielsweise einer JoinEmpty Nachricht unterscheidet.

4.3.1. Zustandsautomat – Registrar state machine

Zustandsautomat:

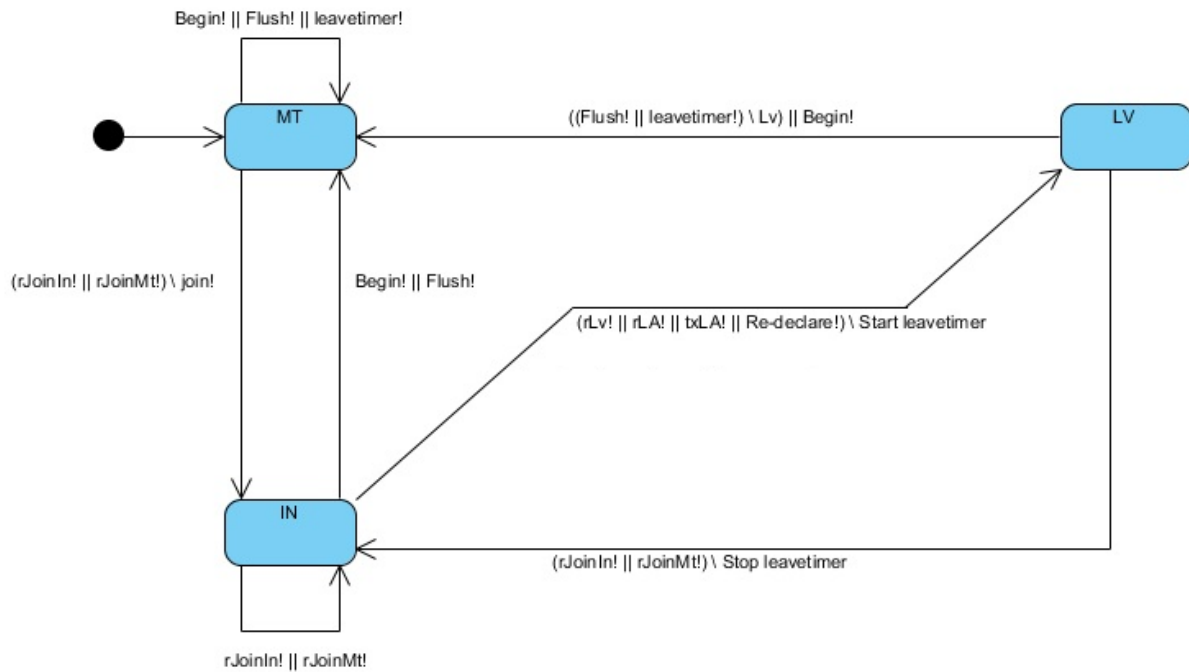


Abbildung 15: Zustandsautomat Registrar state machine

Zustandsübergangstabelle:

		STATE		
		IN	LV	MT
EVENT	Begin!	MT	MT	MT
	rNew!	New IN	New Stop leavetimer IN	New IN
	rJoinIn! rJoinMt!	IN	Stop leavetimer IN	Join IN
	rLv! rLA! txLA! Re-declare!	Start leavetimer LV	-x-	-x-
	Flush!	MT	Lv MT	MT
	leavetimer!	-x-	Lv MT	MT

Abbildung 16: Registrar state machine¹⁰

¹⁰ Quelle: Analyse des MMRP-Protokolls nach IEEE 802.3ak einschließlich Erstellung einer zustandsorientierten Simulation des Protokolls auf Basis von OMNeT++ sowie weiterer Tools für Testzwecke von Johannes Jochen

Zustände:

Zustand	Bedeutung
IN	MAC-Adresse wurde gespeichert
LV	Registrar hat eine Leave Nachricht erhalten. Wenn keine JoinIn Nachricht kommt, wechselt der Automat in den MT Zustand
MT	Registrar ist leer

Events:

Event	Bedeutung
Begin!	Event zum Starten und initialisieren des Zustandsautomaten.
rNew!	Event zum schnellen Registrieren von Attributen. (Findet keine Verwendung in MMRP.)
rJoinIn!	Empfang einer JoinIn-Nachricht über das Netzwerk
rJoinMt!	Empfang einer JoinMT-Nachricht über das Netzwerk. (Wird von der Applicant state machine generiert. Der Unterschied zur JoinIn-Nachricht wird dort erklärt)
rLv!	Empfang einer Leave-Nachricht über das Netzwerk. Dient zur Deregistrierung
rLA!	Empfang einer LeaveAll-Nachricht über das Netzwerk. Zur Deregistrierung durch den garbage collector.
txLA!	Event für die Übertragungsmöglichkeit und einer LeaveAll-Nachricht tritt ein. Hier keine größere Änderung als bei einem normalen rLA!-Event
Re-declare!	Event, das einen Port-Rollen-Wechsel anzeigt. (Von Designated-Port zu Root-Port oder Alternate-Port)
Flush!	Event, das einen Port-Rollen-Wechsel anzeigt. (Von Root-Port oder Alternate-Port zu Designated-Port.)
leavetimer!	Der leavetimer ist abgelaufen und generiert dieses Event, um anzuzeigen, dass der Time-Out zum Reregistrieren vorbei ist.

Actions:

Action	Bedeutung
New	Im MMRP wird die Action nie ausgeführt.
Join	Sendet eine Join Nachricht an die anknüpfende Applicant state machine.
Lv	Sendet eine Leave Nachricht an die anknüpfende Applicant state machine
Start leavetimer	Startet den Timer, der definiert, wie lange ein Empfänger Zeit hat sich zu registrieren, nachdem die Verbindung getrennt wurde.
Stop leavetimer	Der Timer ist abgelaufen und die registrierte MAC-Adresse wird deregistriert.

Anmerkung:

- Weiße Zustände stellen einen Application-Only Automaten dar.
- Hellgelbe Zustände fallen weg, wenn es sich um ein Point-to-Poin-Netzwerk handelt.

Zustände:

Zustand	Bedeutung
VO	<i>Very anxious Observer</i> - Zustand, in dem sich der Automat zu Beginn beendet. Dieser wird auch beim Verlassen der Registrierung wieder erreicht.
VP	<i>Very anxious Passive</i> - Der Automat kommt in den Zustand, da er aufgefordert wurde, die Adresse zu deklarieren. (Dies geschieht durch die internen Join! -Nachricht, die beispielsweise der Registrar aussendet.)
VN	<i>Very anxious New</i> - Nicht weiter beschrieben; MMRP macht keinen Gebrauch des New! -Kommandos, welcher zu diesem Zustand führt
AN	<i>Anxious New</i> - siehe: VN
AA	<i>Anxious Active</i> - Die Applikation beendet sich bei der Deklaration der Adresse. Sie hat bereits eine Join-Nachricht versendet und sendet beim nächsten tx!-Event die zweite Join-Nachricht aus.
QA	<i>Quiet Active</i> - Zustand, in dem die Applikation die Anzahl der nötigen Join-Nachrichten abgesetzt hat. Dies ist der Zustand nach erfolgter Deklaration der Adresse.
LA	<i>Leaving Active</i> - Die Applikation war bereits bei der Deklaration der Adresse und hat eine interne Lv-Nachricht empfangen. Sie sendet beim auftretenden tx!-Event eine Lv-Nachricht über das Netzwerk aus.
AO	<i>Anxious Observer</i> - Die Applikation deklariert selbst keine Adressen, hat aber eine JoinIN-Nachricht (durch das Netzwerk) erhalten.
QO	<i>Quiet Observer</i> - Die Applikation deklariert selbst keine Adressen, hat aber eine JoinIN - Nachricht (durch das Netzwerk) erhalten.
AP	<i>Anxious Passive</i> - Die Applikation hat eine JoinIN-Nachricht empfangen. Anschließend wird sie aufgefordert, diese Adresse im Netzwerk zu deklarieren. (VO->AO->AP)
QP	<i>Quiet Passive</i> - Die Applikation hat zwei JoinIN-Nachrichten empfangen, um in den QO - Zustand zu kommen. Nun wird sie intern aufgefordert, selbst die Adresse zu deklarieren. In der Zwischenzeit sind zwei JoinIN-Nachrichten bei ihr eingegangen, deswegen muss sie selbst keine Nachricht versenden und kann im Zustand QP verbleiben.
LO	<i>Leaving Observer</i> - Die Applikation deklariert diese Adresse nicht. Sie hat aber eine Lv-, LeaveALL- oder Re-Declare-Nachricht empfangen. Beim nächsten tx!-Event versendet sie eine Empty -Nachricht.

Events:

Event	Bedeutung
Begin!	Event zur Initialisierung des Automaten.
New!	Event zum schnellen Registrieren von Attributen. (Findet keine Verwendung in MMRP.)
Join!	Internes Event; Aufforderung zur Deklaration der Adresse.
Lv!	Internes Event; Aufforderung die Deklaration zu deregistrieren.
rNew!	Externe New-Message. (Findet keine Verwendung in MMRP.)
rJoinIn!	Externes Event; Die Application state machine des verbundenen Switches wurde aufgefordert, die Adresse zu deklarieren. Sie hat eine Join-Nachricht abgesetzt, die sagt, dass ihr eigener Zustandsautomat sich im Zustand IN beendet. (Oder, dass sie eine Applicant-Only state machine ist.)
rIn!	Externes Event; der Automat hat den Zustand eines verbundenen Registrars durch einen verbundenen Applicant state machine empfangen. Er befand sich im Zustand IN.
rJoinMt!	Externes Event; Join-Nachricht wurde empfangen mit der Information, dass sich der entsprechende Registrar nicht im Zustand IN befindet.
rMt!	Externes Event; Die Nachricht informiert darüber, dass sich der entsprechende Registrar nicht im Zustand IN beendet.
rLv!	Externes Event; Die Nachricht informiert darüber, dass sich der entsprechende Registrar nicht im Zustand IN beendet.
rLA!	Dieses Event kann sowohl von einer externen als auch von einer internen Quelle stammen. Zeigt an, dass eine LeaveAll-Nachricht empfangen wurde. Eventuelle Deklarationen müssen erneut deklariert werden, um einen Abbau der Deklaration zu vermeiden.
Re-declare	Internes Event; Erneutes Deklarieren durch Wechsel der Port-Rolle nötig.
periodic!	Internes periodisches Event, das durch die Periodic state machine erzeugt wird.
tx!	Event, das anzeigt, dass nun eine Übertragungsmöglichkeit existiert.
txLA!	Event für die Übertragungsmöglichkeit mit zeitgleicher LeaveAll-Nachricht.
txLAF	Event für die Übertragungsmöglichkeit mit zeitgleicher LeaveAll -Nachricht und fehlendem Platz für weitere Adressen.

Actions:

Action	Bedeutung
s	Sendet eine IN- oder Empty-Nachricht in Abhängigkeit der zugehörigen Registrar state machine aus
[s]	Sendet eine IN- oder Empty-Nachricht in Abhängigkeit der zugehörigen Registrar state machine aus. Diese Aktion kann ausgeführt werden, wenn sie zu Code-Optimierungszwecken gebraucht wird.
sj	Sendet eine JoinIN- oder JoinMt-Nachricht in Abhängigkeit der zugehörigen Registrar state machine aus

[sJ]	Sendet eine JoinIN- oder JoinMt-Nachricht in Abhängigkeit der zugehörigen Registrar state machine aus. Diese Aktion kann ausgeführt werden, wenn sie zu Code-Optimierungszwecken gebraucht wird.
sL	Sendet eine Lv-Nachricht aus
sN	Wird nicht benötigt, da MMRP New!-Events nicht verwendet

4.3.3. Zustandsautomat - LeaveAll state machine

Zustandsautomat:

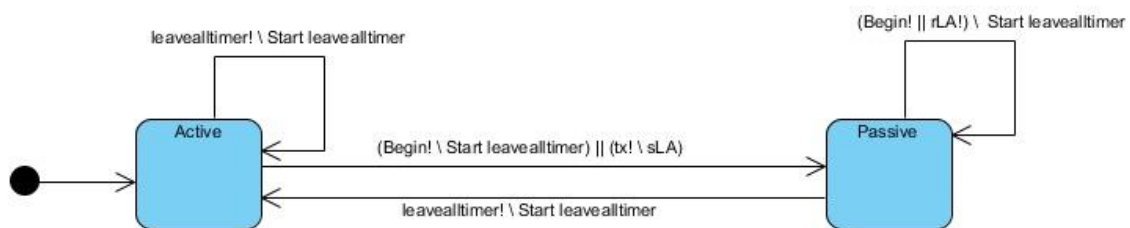


Abbildung 19: Zustandsautomat - LeaveAll state machine

Zustandsübergangstabelle:

		STATE	
		Active	Passive
EVENT	Begin!	Start leavealltimer Passive	Start leavealltimer Passive
	tx!	sLA Passive	-x-
	rLA!	Start leavealltimer Passive	Start leavealltimer Passive
	leavealltimer!	Start leavealltimer Active	Start leavealltimer Active

Abbildung 20: LeaveAll state machine¹²

¹²Quelle: Analyse des MMRP-Protokolls nach IEEE 802.3ak einschließlich Erstellung einer zustandsorientierten Simulation des Protokolls auf Basis von OMNeT++ sowie weiterer Tools für Testzwecke von Johannes Jochen

Zustände:

Zustand	Bedeutung
Active	Automat ist bereit eine LeaveAll-Nachricht zu verschicken
Passive	Automat ist inaktiv

Events:

Event	Bedeutung
Begin!	Event zur Initialisierung des Automaten
tx!	Event, das anzeigt, dass nun eine Übertragungsmöglichkeit existiert
rLA!	Eine Externe LeaveAll-Nachricht wurde empfangen
leavealltimer!	Event verursacht, dass der Automat eine LeaveAll-Nachricht verschickt.

Actions:

Action	Bedeutung
sLA	Eine LeaveAll Nachricht wird ins Netz geschickt
Start	Nachdem der Timer abgelaufen ist, wird eine LeaveAll Nachricht verschickt
leavealltimer	

4.3.4. Zustandsautomat - Periodic Transmission state machine

Zustandsautomat:

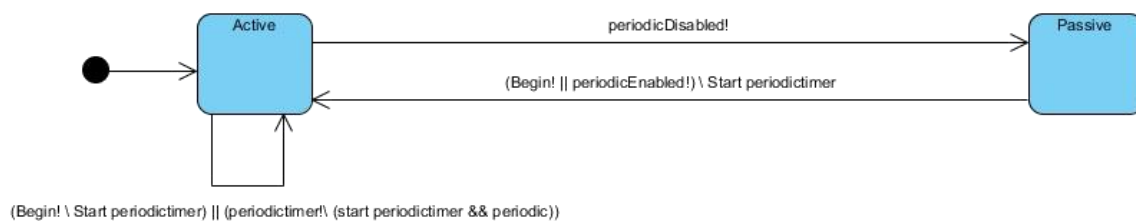


Abbildung 21: Zustandsautomat - PeriodicTransmission state machine

Zustandsübergangstabelle:

		STATE	
		Active	Passive
EVENT	Begin!	Start periodictimer Active	Start periodictimer Active
	periodicEnabled!	-x-	Start periodictimer Active
	periodicDisabled!	Passive	-x-
	periodictimer!	Start periodictimer periodic Active	-x-

Abbildung 22: Periodic Transmission state machine¹³

Zustände:

Zustand	Bedeutung
Active	Automat ist aktiv
Passive	Automat ist inaktiv

Events:

Event	Bedeutung
Begin!	Der Automat wird initialisiert.
periodicEnabled!	Periodic Transmission state machine wurde aktiviert.
periodicDisabled!	Periodic Transmission state machine wurde deaktiviert.
periodictimer!	Automat schickt ein internes periodic-Event an die Applicant state machine.

Actions:

Action	Bedeutung
Start periodictimer	Nachdem der Timer abgelaufen ist, wird eine periodic Nachricht abgeschickt.
periodic	Sendet eine periodic Nachricht an die Applicant state machine, damit sie eine JoinIn Nachricht schickt.

¹³Quelle: <http://standards.ieee.org/getieee802/download/802.1ak-2007.pdf> Seite 58

4.3.5. Das MMRP-Paket

Generell werden im Netzwerk Pakete im IEEE 802.3-Frame Format verschickt. Dieses Format ist folgendermaßen aufgebaut:

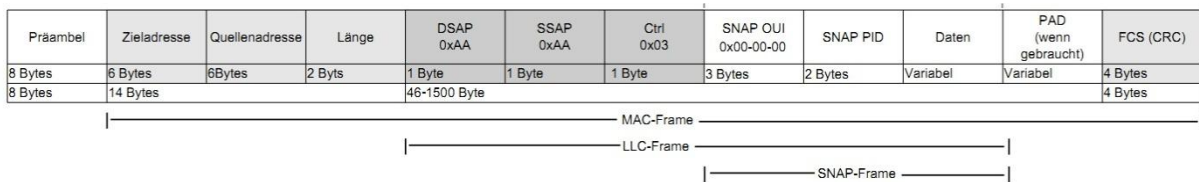


Abbildung 23: Paketaufbau nach IEEE 802.3¹⁴

Erläuterungen zum IEEE 802.3-Frame:

Frame Abschnitt	Bedeutung
Präambel	Dient zur zeitlichen Abstimmung der Kommunizierenden (Quelle: Wikipedia)
Zieladresse	MAC-Adresse vom Empfänger
Quellenadresse	MAC-Adresse vom Sender
Länge	Größe des Pakets
DSAP	Destination Service Access Point, enthält das Individual oder das Group Bit
SSAP	Source Service Access Point; Bit sagt aus, ob es ein Command oder eine Response Nachricht ist
Ctrl	Dienst zur Erkennung, ob ein SNAP-Frame enthalten ist
SNAP OUI	Enthält den Wert 0x000 zur Erkennung, welcher Protokolltyp enthalten ist
SNAP PID	Enthält den Protokolltyp
Daten	protokollspezifische Daten
PAD	Wird genutzt, um das Paket auf 64 Byte zu bringen, wenn das Paket kleiner ist
FCS(CRC)	Feld, dass zur Fehlererkennung dient

¹⁴Quelle: Analyse des MMRP-Protokolls nach IEEE 802.3ak einschließlich Erstellung einer zustandsorientierten Simulation des Protokolls auf Basis von OMNeT++ sowie weiterer Tools für Testzwecke von Johannes Jochen

Das MMRP Paket ist im IEEE 802.3-Frame eingebettet. Genau definiert ist das MMRP Paket im SNAP-Frame. Es setzt sich folgendermaßen zusammen:

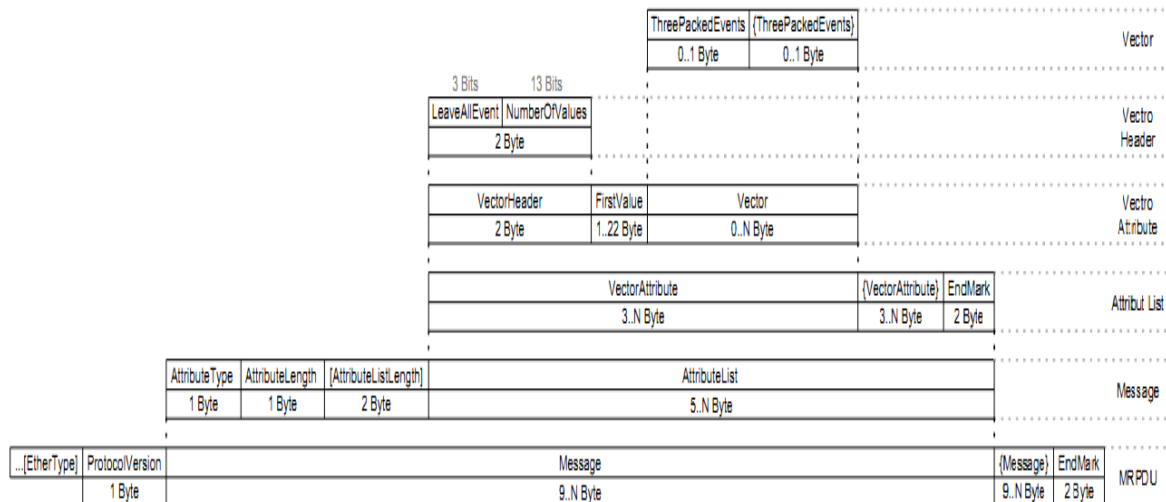


Abbildung 24: Aufbau MMR-Paket¹⁵

Erläuterungen zum MMRP Paket:

MMRP Abschnitt	Bedeutung
EtherType	Hier ist der Protokolltyp definiert. Im Falle von MMRP steht hier 0x88f6
ProtocolVersion	Hier steht die Protokollversion
Message	Hier steht die eigentliche Information. Der Messageabschnitt ist unterteilt in AttributeType, AttributeLength, AttributeListLength und AttributeList
AttributeType	Hier ist angegeben, welcher Typ von Attribut verschickt wird. Das MMRP Protokoll ist im Stadi zwei Typen zu verschicken. Typ1: Service Requirement Typ2: MAC-Adresse
AttributeLength	Gibt die Größe von Attribute an
AttributeList	Enthält eine bestimmte Anzahl von VectorAttributes
VectorAttribute	Setzt sich zusammen aus VectorHeader, FirstValue und Vector
VectorHeader	Enthält das LeaveAllEvent und NumberOfValues
LeaveAllEvent	Hier wird definiert, ob die Nachricht eine LeaveAll-Nachricht ist
NumberOfValues	Hier ist die Anzahl der Nachrichten definiert. Nachrichten wie beispielsweise JoinIn und Leave können zusammengefasst in einem Paket

¹⁵ Quelle: Analyse des MMRP-Protokolls nach IEEE 802.3ak einschließlich Erstellung einer zustandsorientierten Simulation des Protokolls auf Basis von OMNeT++ sowie weiterer Tools für Testzwecke von Johannes Jochen

	versendet werden. Die maximale Anzahl der zusammengefassten Nachrichten pro ThreePacketEvents beträgt drei
FirstValue	Gibt an, wieviele Nachrichten verschickt werden. Möglich sind die Werte 0 bis 3
Vector	Hier sind die zusammengefassten Nachrichten gespeichert
ThreePacketEvents	Ein ThreePacketEvent kann drei Nachrichten enthalten, die wie folgt gespeichert sind: $36 * \text{ErsteNachrichtTyp} + 6 * \text{ZweiteNachrichtTyp} + \text{DritteNachrichtTyp}$ Folgende Nachrichtentypen sind möglich: JoinIn, In, JoinMt, Empty, Leave
EndMark	Definiert das Ende vom Paket. Die zwei Bytes sind gefüllt mit Nullen

4.4. Systemdesign und Programmstruktur

<MOD-ID>	1. Registrar state machine
Abgedeckte Systemanforderungen	MMR-Protokoll-Unterstützung
Leistung:	Zustandsautomat, der die MAC-Adresse registrieren kann
Schnittstellen:	Filesystem, MCastor2.0 client
Externe Daten:	Logfile für den aktuellen Zustand
Ablageort:	/Multicastor
Offene Punkte	-

<MOD-ID>	2. Application state machine
Abgedeckte Systemanforderungen	MMR-Protokoll-Unterstützung
Leistung:	Zustandsautomat, der die Kommunikation im Netzwerk durchführen kann
Schnittstellen:	Filesystem, MCastor2.0 client
Externe Daten:	Logfile für den aktuellen Zustand
Ablageort:	/Multicastor
Offene Punkte	-

<MOD-ID>	3. LeaveAll state machine
Abgedeckte Systemanforderungen	MMR-Protokoll-Unterstützung
Leistung:	Zustandsautomat, der in bestimmten Zeitintervallen LeaveAll-Messages abschicken kann
Schnittstellen:	Filesystem, MCastor2.0 client
Externe Daten:	Logfile für den aktuellen Zustand
Ablageort:	/Multicastor

Offene Punkte	-
<MOD-ID>	4. Periodic Transmission state machine
Abgedeckte Systemanforderungen	MMR-Protokoll-Unterstützung
Leistung:	Zustandsautomat, der die Application state machine auffordert, JoinIn-Nachrichten zu verschicken
Schnittstellen:	Filesystem, MCastor2.0 client
Externe Daten:	Logfile für den aktuellen Zustand
Ablageort:	/Multicastor
Offene Punkte	-
<MOD-ID>	5. MMRP Paket
Abgedeckte Systemanforderungen	MMR-Protokoll-Unterstützung
Leistung:	Switch soll mit den Paketen Multicasting auf Layer-2-Ebene betreiben
Schnittstellen:	Netzwerkkarte
Externe Daten:	-
Ablageort:	-
Offene Punkte	-

5. Integration in das Testframework STAF/STAX

5.1. Einführung

Software Testing Automation Framework (STAF) ist ein OpenSource Framework für automatisierte Software Tests, basierend auf wiederverwendbaren Komponenten - Services genannt - und mehreren Systemen. STAF eXecution Engine (STAX) ist ein STAF Service, der das Ausführen und Testen von Software mit XML und Python (genauer Jython) erleichtert.

5.2. Technische Dokumentation

STAX ist ein STAF Service. Über ihn kann man bequem STAF-Tests in XML anlegen. Auf den Test Clients muss nur STAFProc gestartet und in der STAF.cfg dem Server (STAX Monitor) Security Level 5 gegeben werden. Nur auf dem Server muss STAF mit dem Service STAX

installiert werden. STAX verwendet die normalen STAF Funktionen und daher muss auf den Clients kein STAX installiert werden.

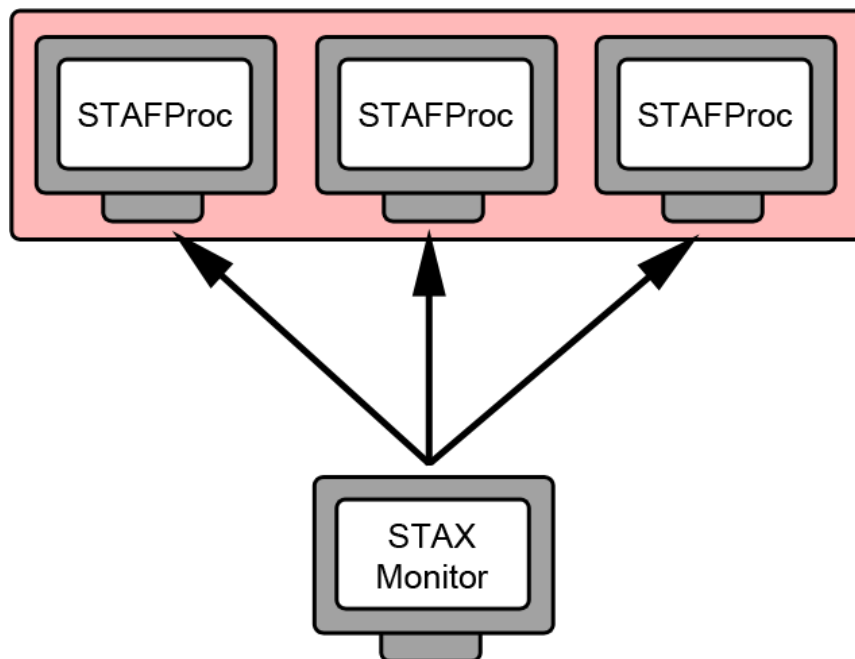


Abbildung 25: STAF - STAX Zusammenspiel

STAX liefert einen in Java geschriebenen STAX Monitor GUI (STAXMon.jar), mit dem man bequem die Tests (XML-Dateien) laden und ausführen kann. Hier ist eine kleines simples Beispiel STAX Skript:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE stax SYSTEM "stax.dtd">
<stax>
  <defaultcall function="main"/>
  <function name="main">
    <sequence>
      <stafcmd>
        <location>'local'</location>
        <service>'delay'</service>
        <request>'delay 50000'</request>
      </stafcmd>
      <if expr="RC != 0">
        <message>
          'Error: RC = %s' % (RC)
        </message>
      <else>
        <message>
          'Success = %s' % (STAXResult)
        </message>
      </else>
    </if>
  </sequence>
</function>
</stax>
```

Hier wird eine *main* Funktion definiert, die mit dem defaultcall aufgerufen wird. In der main Funktion wird eine Sequenz gestartet, die einen STAF Command/Service "delay" mit 50 Sekunden ausführt. Wenn der delay fehlschlägt, wird eine Error Message angezeigt.

Im folgenden Bild kann man den STAX Job Monitor sehen.

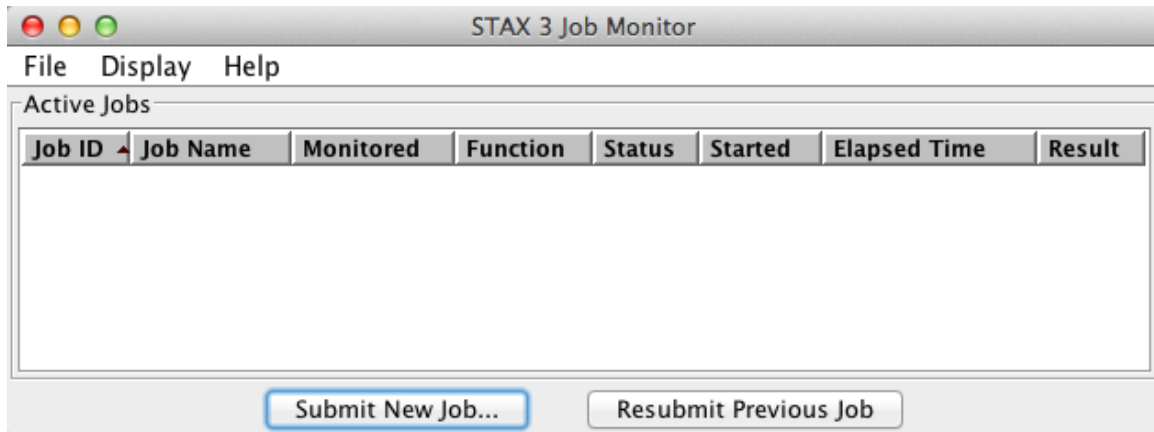


Abbildung 26: STAX Job Monitor

Hier können wir nun neue Jobs hinzufügen.

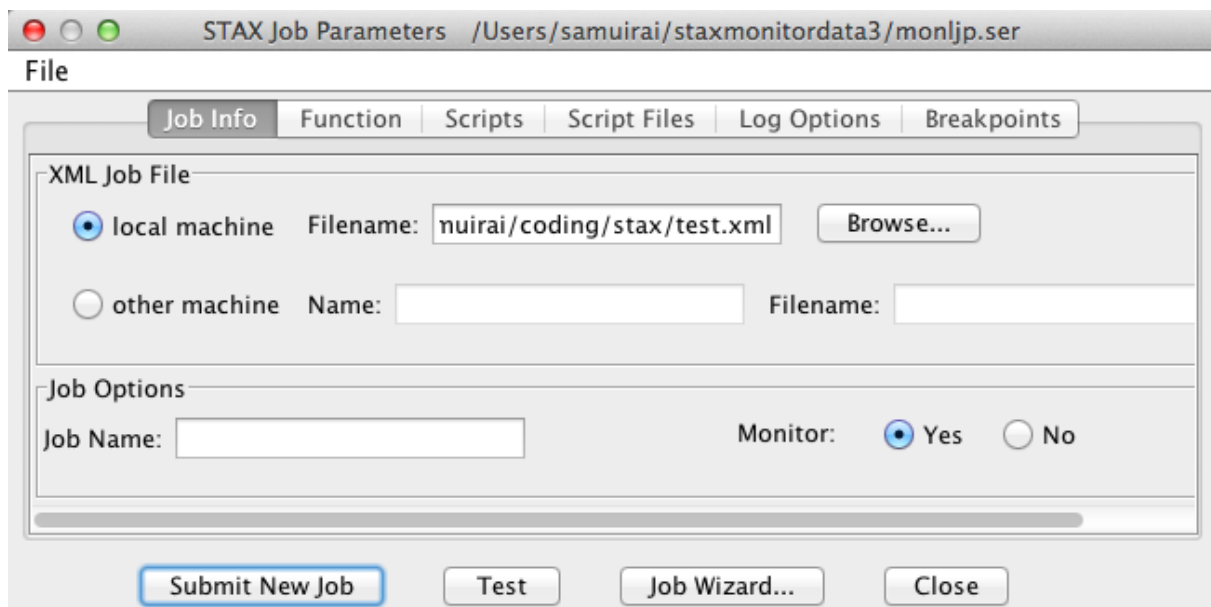


Abbildung 27: STAX Job Monitor: Job hinzufügen

Nun wird die *.xml-Datei ausgewählt und der Job kann gestartet werden.

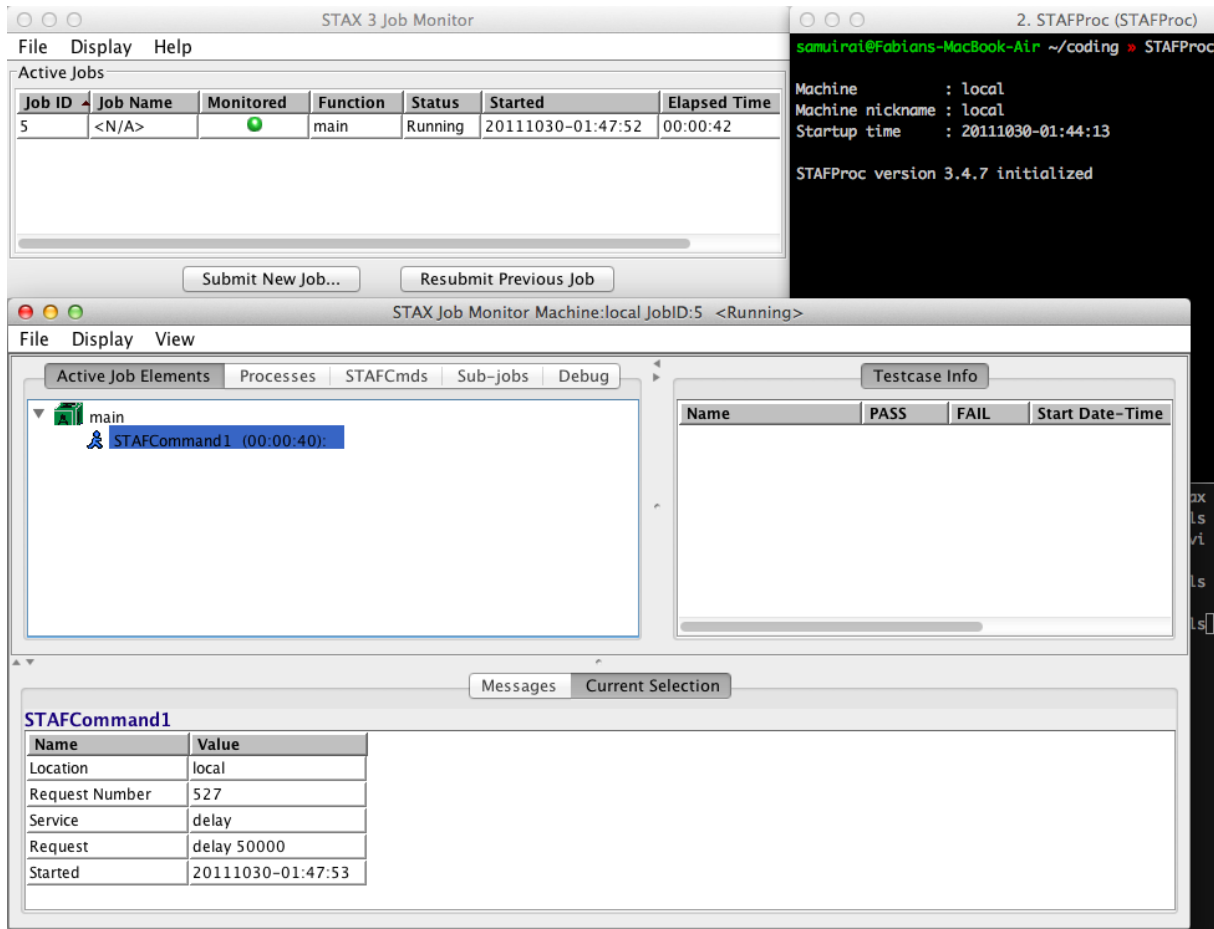


Abbildung 28: STAX Job Monitor: Job starten

STAX liefert viele Informationen, welche man über die XML Datei definieren kann und dabei dank Jython auf die kompletten Java und Python Libraries zurückgreifen kann.

5.3. Systemdesign und Programmstruktur

Um die MultiCastor2.0 Blackbox Tests durchzuführen, werden für die einzelnen Bereiche STAX-XML-Skripte erstellt. Die nun folgenden Module beschreiben die benötigten Skripte näher und beziehen sich auf die Anforderungsnummern aus dem Business Case.

<MOD-ID>	/LF81.0/ Aufsetzen von Config-Filetests
Abgedeckte Systemanforderungen	Installiertes STAF/STAX
Leistung:	Testet das korrekte Laden/Verwenden der MultiCastor2.0 Konfigurationsdateien
Schnittstellen:	Filesystem, MultiCastor2.0 CLI
Externe Daten:	vordefinierte Konfigurationsdateien für die Tests
Ablageort:	/stax
Offene Punkte	-

<MOD-ID>	/LF81.1/ Aufsetzen von Log-Filetests
Abgedeckte Systemanforderungen	Installiertes STAF/STAX
Leistung:	Testet, ob die Logfiles die erwarteten Daten beinhalten
Schnittstellen:	Filesystem, MultiCastor2.0 CLI
Externe Daten:	erwartete Werte definieren
Ablageort:	/stax
Offene Punkte	-

<MOD-ID>	/LF81.2/ Aufsetzen von Ethernetttests
Abgedeckte Systemanforderungen	Installiertes STAF/STAX
Leistung:	Testet, ob die Grundfunktionalitäten des MultiCastors2.0 – die Netzwerk/Ethernet Aufgaben - korrekt erfüllt werden
Schnittstellen:	Ethernet, MultiCastor2.0 CLI
Externe Daten:	
Ablageort:	/stax
Offene Punkte	STAF Service für Ethernet sniffing. Eventuell externes Programm „wireshark“ nutzen.

6. Programmreaktion im Fehlerfall

In diesem Abschnitt wird die Programmreaktion im Fehlerfall behandelt. Darunter fällt das Verhalten des MCastor2.0, wenn nicht zu vermeidende Fehler im Umfeld passieren. Die wohl größte Gefahr für den MCastor2.0 besteht darin, dass es Verbindungsprobleme (Netzwerkabrisse) geben kann.

6.1. Grundlegendes

Es ist immer möglich, dass ein Sender/Receiver die LAN oder Internetverbindung verliert. Dies kann beispielsweise durch ein Ausstecken des Kabels oder ein Ausfall von Knotenpunkten im Netzwerk geschehen. Dieser Fall muss deshalb beachtet und vom Programm behandelt werden.

6.2. Existierendes Problem Handling

Bisher wird das Senden und Empfangen durch die Klassen "MulticastSender" und "MulticastReceiver" übernommen. Diese beiden Klassen erben von der abstrakten Klasse "MulticastThreadSuper", welche das Interface Runnable implementiert. Das bedeutet, dass sowohl Sender als auch Receiver "Runnable" sind und die entscheidende Methode **run()** implementieren.

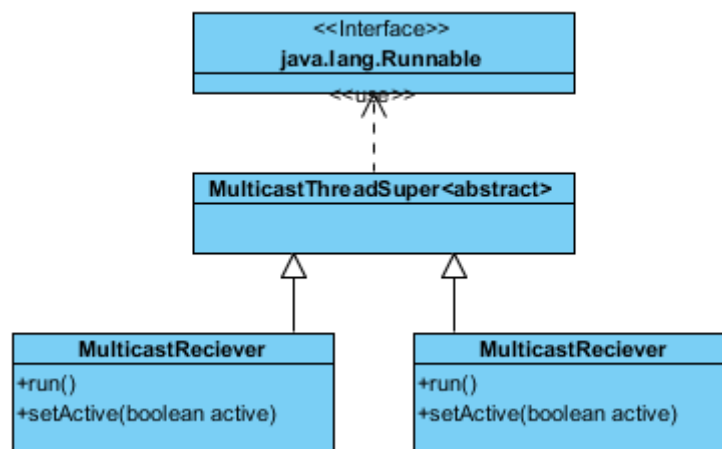


Abbildung 29: Klassendiagramm MulticastReceiver

Diese wird aufgerufen wenn der Thread (in unserem Fall der Sender/Receiver) aktiv wird. In dieser Methode gibt es jeweils eine while Schleife:

```

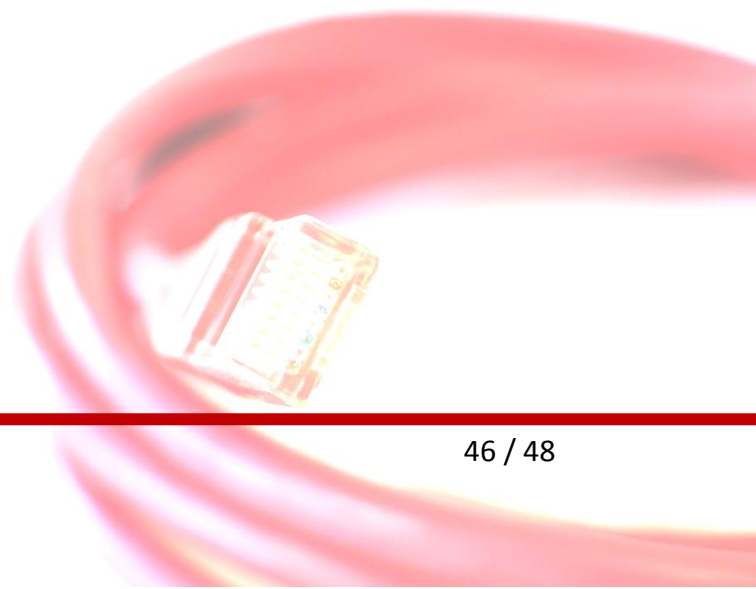
#Code zum Beenden des Sender/Receiver#
while(isSending){
    try {
        #Code vom Senden/Empfangen#
    } catch(IOException e) {    #...#
        #Logging des Fehlers#
    }
}

```

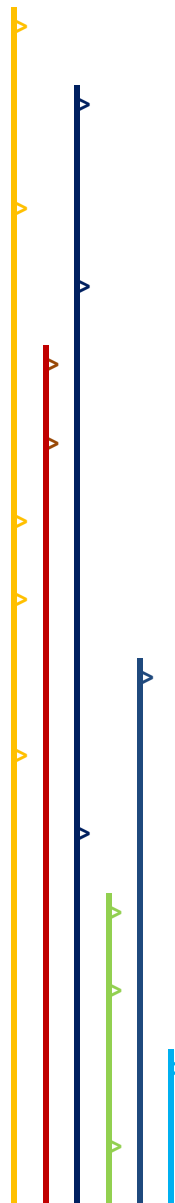
Diese Schleife sorgt dafür, dass der Sender/Receiver ständig aktiv bleibt und arbeitet, bis die boolean-Variable **isSending** auf false gesetzt wird. Dies kann von außen über die Methode **setActive(boolean)** passieren. Das bedeutet, dass der Sender/Receiver - solange er nicht von außen auf inaktiv gestellt wird - einfach weiter sendet bzw. probiert zu empfangen und dies lediglich loggt.

6.3. Gewolltes Verhalten

Im MCastor2.0 soll das bisherige Verhalten im Fehlerfall verbessert werden. Wenn ein Verbindungsproblem besteht, soll das ein Mal geloggt werden. Anschließend soll nicht weiter normal gesendet werden, sondern probiert werden, die Verbindung wieder aufzubauen. Dies soll für ca. 10-15 Sekunden geschehen. Dieses Zeitintervall ist wegen des MMRP Protokoll gewählt, da nach 10-15 Sekunden der Garbage Collector aktiv wird und "aufräumt" (, falls Sender/Receiver nicht mehr antwortet). Wenn in dieser Zeit die Verbindung nicht hergestellt werden konnte, soll der Benutzer darüber informiert werden und über einen Dialog entscheiden, ob er weiter probieren möchte, die Verbindung aufrechtzuerhalten oder den Sender/Receiver deaktivieren will.



Dokumentversionen



<u>18.10.2011 Filip Haase:</u>	Fachbegriffe Glossar: Dokument angelegt
<u>20.10.2011 Jonas Traub</u>	Sammeldokument für verschiedene Abschnitte angelegt (Bestimmte Abschnitte werden von den Autoren in dieses Dokument nach Fertigstellung eingefügt)
<u>25.10.2011 Fabian Fäßler:</u>	STAF/STAX spezifische Begriffe hinzugefügt
<u>27.10.2011 Filip Haase</u>	Abschnitt 6 – Programmverhalten im Fehlerfall eingefügt
<u>26.10.2011 Matthias Hauschild</u>	Abschnitt 1 – Einführung – Dokument angelegt und Beschreibung erstellt
<u>29.10.2011 Matthias Hauschild</u>	Ergänzung von Informationen zu MMRP, STAF/STAX, GUI/Usability
<u>26.10.2011 Filip Haase:</u>	Fachbegriffe zu Java hinzugefügt
<u>27.10.2011 Jonas Traub:</u>	Abkürzungen aus SRS Abschnitt Produktfunktionen eingefügt
<u>27.10.2011 Fabian Fäßler</u>	Abschnitt 5 – Staf/Stax – Testverfahren mit Staf/Stax beschrieben
<u>28.10.2011 Jonas Traub:</u>	Fehlende Beschreibungen zu Abkürzungen hinzugefügt
<u>01.11.2011 Matthias Hauschild</u>	Abschnitt 3.3 – Änderungen an der Konfiguration hinzugefügt
<u>01.11.2011 Jonas Traub</u>	Abschnitt 3.1 – erweiterte Architekturübersicht – Grafik erstellt, Dokument angelegt
<u>02.11.2011 Filip Haase / Matthias Hauschild / Fabian Fäßler / Sebastian Koralewski</u>	Erweiterung der Grafik um MMRP, GUI-Änderungen, Konfigurationskonzept und STAF/STAX Anbindung
<u>03.11.2011 Christopher Westphal</u>	Abschnitt 2 – Systemübersicht – erstellt
<u>04.11.2011 Fabian Fäßler</u>	Beschreibung zu STAF/STAX eingefügt

Lizenz/License:

© Fäßler, Haase, Hauschild, Koralewski, Traub, Westphal

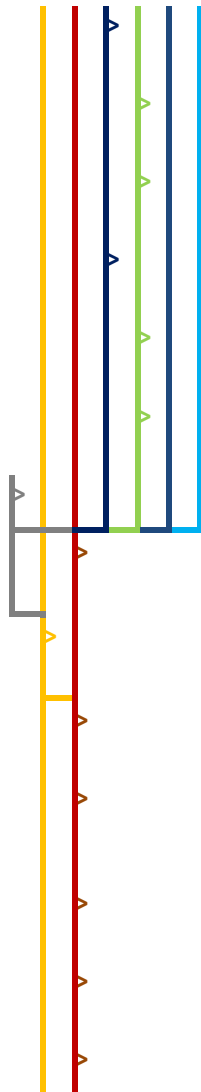
Dokumentversion/Document Version:

Titel: System Architecture Specification

Version: V 1.0 beta 7 (10. November 2011)

Autoren/Authors Projektteam/Project team:

- | | |
|------------------------|--|
| • Jonas Traub | (Projektleiter) |
| • Filip Haase | (Leading Engineer) |
| • Matthias Hauschild | (Documentation) |
| • Fabian Fäßler | (Engineer/Tester, Expert on STAF/STAX) |
| • Christopher Westphal | (Engineer/Tester, Expert on Usability) |
| • Sebastian Koralewski | (Engineer/Tester, Expert on MMRP) |



04.11.2011 Sebastian Koralewski

Abschnitt 4 – Integration eines neuen Protokolls – MMRP - hinzugefügt

05.11.2011 Filip Haase

Bestehende Programmstruktur beschrieben

06.11.2011 Christopher Westphal

Änderungen und Erweiterungen an GUI beschrieben

06.11.2011 Christopher Westphal

Abschnitt 3.2 - Änderungen an der grafischen Oberfläche eingefügt

06.11.2011 Matthias Hauschild / Jonas Traub

Konfigurationsdateien in Zusammenspiel mit dem Multicastor beschrieben

06.11.2011 Sebastian Koralewski

Erweiterung um das MMRP-Protokoll beschrieben

MCastor2.0 Document Template

01.11.2011 Matthias Hauschild

Dokumente zusammen Geführt zu SAS beta 1

01.11.2011 Matthias Hauschild

in MCastor2.0 Template eingefügt und Fehlerkorrektur

01.11.2011 Matthias Hauschild

Glossar als Anlage beigefügt

04.11.2011 Matthias Hauschild

Anpassung von Formulierungen und Formatierungen nach Gruppengespräch
Hinzufügen der Grafiken zu Statemachines von Sebastia Koralewski

07.11.2011 Jonas Traub

Verschiedene Änderungen

07.11.2011 Matthias Hauschild / Jonas Traub

Abschließende Formulierung, letzte Änderungen

10.11.2011 Jonas Traub

Dokument finalisiert

Lizenz/License:

© Fäßler, Haase, Hauschild, Koralewski, Traub, Westphal

Dokumentversion/Document Version:

Titel: System Architecture Specification

Version: V 1.0 beta 7 (10. November 2011)

Autoren/Authors Projektteam/Project team:

- Jonas Traub (Projektleiter)
- Filip Haase (Leading Engineer)
- Matthias Hauschild (Documentation)
- Fabian Fäßler (Engineer/Tester, Expert on STAF/STAX)
- Christopher Westphal (Engineer/Tester, Expert on Usability)
- Sebastian Koralewski (Engineer/Tester, Expert on MMRP)