Optimierte Messenger-Applikation zur Notfallkommunikation via LoRaWAN-DTN

Denis Orlov[1], Franz Kuntke[2], Christian Reuter[2]

Abstract: Die vorliegende Arbeit präsentiert die Entwicklung einer Messenger-App mit Schwerpunkt auf Benutzerfreundlichkeit, für die Nutzung mit einem bestehenden LoRaWAN-DTN-Backend. Die App ermöglicht den Austausch von Nachrichten mit anderen Personen über ein vorhandenes Kommunikationssystem auf LoRaWAN-Basis. Das grundlegende Softwaregerüst wurde mithilfe agiler Softwareentwicklungsmethoden als Progressive-Web-App entwickelt und iterativ verbessert. Das Ergebnis ist eine plattformübergreifende App für Desktop-PCs und Android-Smartphones. Die App bietet grundlegende Messenger-Funktionen wie Kontaktverwaltung, Chatverlauf-Speicher und Benachrichtigungen. Zusätzlich enthält die App erweiterte Funktionen wie einen leicht zugänglichen SOS-Button, um Notfallnachrichten schnell absetzen zu können. Ziel der Entwicklung war es, die Gebrauchstauglichkeit gegenüber einem ersten Prototyp zu verbessern. Die App soll effektive Kommunikation zwischen Helfern und Betroffenen ermöglichen, während und nach Krisenereignissen wie beispielsweise der europäischen Flutkatastrophe 2021. In folgenden Arbeiten soll das System unter Nutzung dieser App im Einsatz getestet werden.

Keywords: Notfall-Messenger, Usability, LoRaWAN Multi-Hop, Disruption Tolerant Networking (DTN), Bundle Protocol 7 (BP7)

1 Einleitung

Internetbasierte Kommunikation hat sich zu einem unverzichtbaren Bestandteil der modernen Gesellschaft entwickelt. Allerdings haben in der Vergangenheit Krisenereignisse wie die europäische Flutkatastrophe 2021 gezeigt, dass Kommunikationsinfrastrukturen auch verletzlich sind und deren großflächige Reparatur besonders in dünn besiedelten Regionen einige Zeit in Anspruch nehmen kann. Gerade während solcher Krisen, aber auch im Nachgang besteht jedoch ein Bedarf an Kommunikation zwischen Helfern und Betroffenen. Als ein Lösungsansatz wurde bereits in einer vorhergehenden Arbeit [KBR23] der Ansatz eines Internet-unabhängigen Kommunikationsnetzwerkes speziell für landwirtschaftliche Betriebe vorgestellt, das auf herkömmliche LoRaWAN-Gateways setzt und autark betrieben werden kann. Es wurde dadurch gezeigt, dass die Sterntopologie von LoRaWAN mit Hilfe eines Disruption

[1] Technische Universität Darmstadt (TUDa), Wissenschaft und Technik für Frieden und Sicherheit (PEASEC), Pankratiusstr. 2, 64298 Darmstadt, https://orcid.org/0000-0003-4134-5630
[2] Technische Universität Darmstadt (TUDa), Wissenschaft und Technik für Frieden und Sicherheit (PEASEC), Pankratiusstr. 2, 64298 Darmstadt, <nachname>@peasec.tu-darmstadt.de, https://orcid.org/0000-0002-7656-5919, https://orcid.org/0000-0003-1920-038X

Tolerant Network (DTN)-Ansatzes in ein Multi-Hop-fähiges Netzwerk umgewandelt werden kann. Die Gebrauchstauglichkeit des ersten Prototypen stand dabei nicht im Vordergrund und lässt entsprechend zu wünschen übrig.

In der vorliegenden Arbeit wird die Entwicklung einer neuen Messenger-App vorgestellt, mit einem Fokus auf hohe Gebrauchstauglichkeit. Die entstandene App ist in der Lage, sich mit dem bestehenden LoRaWAN-DTN-Backend zu verbinden und Nachrichten im Notfallkommunikationsnetz auszutauschen, durch Senden und Empfangen. Ziel war es, die grundlegenden Funktionen eines Messengers, wie z.B. eine Kontaktverwaltung, ein Chatverlauf-Speicher und Benachrichtigungen eingegangener Nachrichten umzusetzen und dabei eine hohe Zugänglichkeit durch eine einfache und ansprechende Oberfläche zu erhalten. Ebenso wurden Zusatzfunktionen eingebaut, wie ein einfach erreichbarer SOS-Button um schnell Notfallnachrichten absetzen zu können.

Im Folgenden werden zunächst Grundlagen über die Backend-Infrastruktur dargestellt, und anschließend auf das Konzept und Umsetzung der App eingegangen. Es folgt eine erste heuristische Usability-Analyse der Entwicklung nach den 10 Kriterien von Nielson.

2 Grundlagen: Notfallkommunikation via LoRaWAN-Backend

Im geplanten Einsatzszenario besteht ein Kommunikations-Backend aus den folgenden Komponenten (siehe Abbildung 1): Ein LoRaWAN-Gateway und ein verbundener (Mini-)Server, auf dem eine ChirpStack-Instanz mit der Erweiterung *Spatz* läuft. Das Interface von *Spatz* ist im lokalen Netzwerk beispielsweise per Wi-Fi erreichbar und ist entsprechend per Client Applikation auf einem Smartphone nutzbar. Das Backend ist für das Routing der Nachrichtenpakete zuständig und nutzt dabei einen Disruption Tolerant Networking (DTN) Ansatz. Hierbei werden Pakete zu logischen Einheiten gebündelt, und in einer *Store-Carry-Forward* Methode weitergeleitet. Die Routing-Algorithmen und Clients wurden unter Annahme entwickelt, dass Geräte auch zeitweise oder permanent nicht verfügbar sein können und müssen entsprechend mit solchen Situationen umgehen. Details zum Backend sind in einer Vorarbeit veröffentlicht [KBR23].

Für die Messenger-App ist das clientseitige Interface von *Spatz* selbst wichtig. Dieses ist per WebSockets erreichbar und nimmt Nachrichten entgegen. Die Nachrichtenpakete

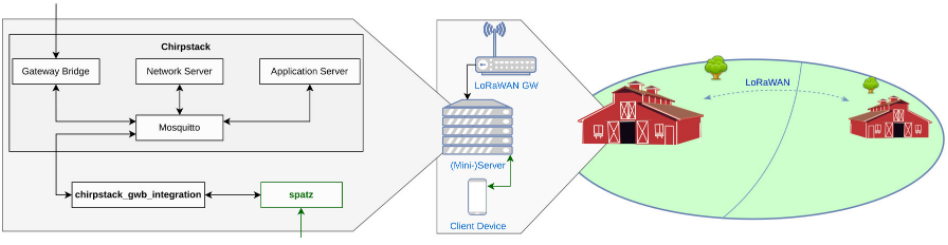

Abbildung 1: Schaubild der zugrundeliegenden Architektur (Quelle: [KBR23])

entsprechen dabei dem Standard Bundle Protocol 7 (BP7) [BFB22] und werden an das Backend in serialisierter Form übertragen, als Concise Binary Object Bundle (CBOR) [BH20]. Ebenso werden eingehende Nachrichten bei aktiver WebSocket-Verbindung an den Client in Form von CBOR-kodierten BP7 Paketen geschickt. Eine bestehende Referenzimplementierung für den Client basierend auf VueJS/JavaScript existiert bereits. Da das zugrundeliegende Konzept des Notfallkommunikationssystem auf DTN basiert, ist prinzipiell eine spätere Erweiterung denkbar, dass Smartphones um die Funktion von *Data Mules* erweitert werden. Dies würde ermöglichen, dass *Netzwerk/-Funklücken* über den Transport von Daten auf Smartphones manuell geschlossen werden können, insofern die Informationen eine hohe Latenz verkraften und entsprechend nicht zeitkritisch sind.

3 Konzept & Umsetzung

Das Konzept orientierte sich im Aufbau an bestehenden Messengern (siehe Abbildung 2) und wurde als Progressive Web App mit Ionic/Angular [WU18] umgesetzt, um mit einer Code-Basis[3] die beiden Ziel-Plattformen Android und PC (Browser) abzudecken – das

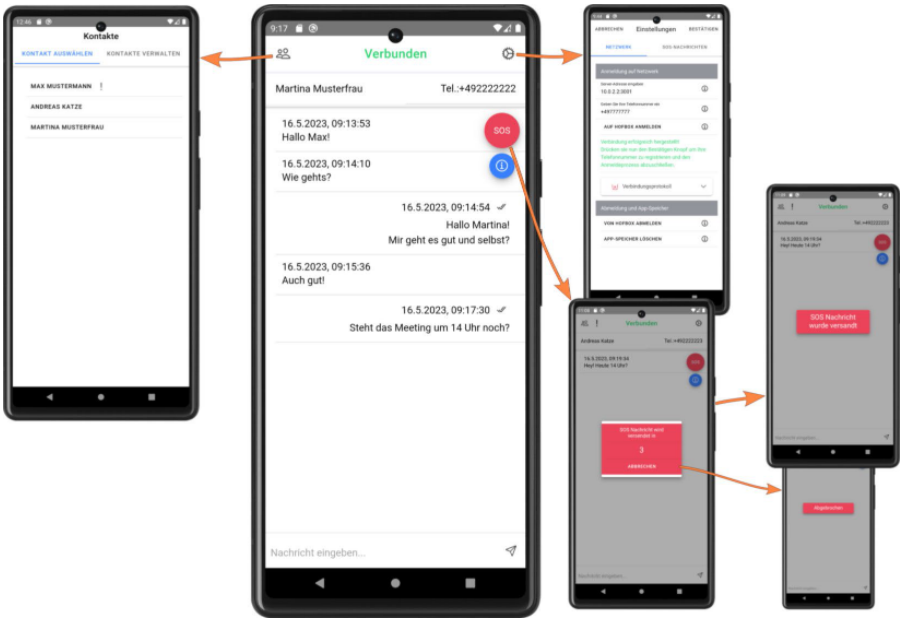

Abbildung 2: Die Oberfläche besteht aus 3 Bereichen: Eine Kontaktübersicht (links), eine Chatansicht (mittig) und eine Einstellungsansicht (oben rechts). In der Chatansicht wird ein SOS-Button angezeigt, der bei Betätigung nach Ablauf von 5 Sekunden (unten rechts) definierten Kontakten eine Notfallnachricht schickt.

[3] https://github.com/PEASEC/LoRaWAN-DTN-Messenger

Framework verspricht hier später auch native Clients für weitere Plattformen wie Apple iOS und Desktop-Systeme (Apple macOS, Microsoft Windows und GNU/Linux) zu erstellen. Das zentrale Element stellt die Chatansicht dar, in dem der Nachrichtenverlauf mit einem einzelnen Kontakt chronologisch aufgelistet wird. Ein Eingabefeld ermöglicht die Eingabe und das Abschicken einer neuen Nachricht. Ein textueller Indikator am oberen Rand zeigt dabei stets den aktuellen Verbindungsstatus zum Backend an („Verbunden“, „Nicht verbunden“). Symbole an den Nachrichten selbst zeigen dabei den Status an, ob das Backend die jeweilige (eigene) Nachricht in Empfang genommen hat. Somit kann die Nutzer:in sehen, ob das Backend die Nachrichten erhalten hat. Eine dedizierte Bestätigung des Empfangs einer Nachricht beim Gegenüber ist aufgrund von System-Limitierungen im ersten Schritt nicht vorgesehen, könnte jedoch nachträglich hinzugefügt werden. Weiterhin ist in der Chat-Ansicht ein SOS-Button enthalten, der einen Countdown startet, nach dessen Ablauf eine vorgefertigte Nachricht inkl. (bei Verfügbarkeit) aktueller Geo-Koordinate (Längen-/Breitengrad) an einen hinterlegten Notfall-Kontakt versandt wird. Beim Empfang einer Nachricht mit Geo-Koordinate wird daraus automatisch ein Link generiert, der die Position in OpenStreetMap im Browser darstellt, was beim Empfänger Internetverbindung voraussetzt. Alternativ kann die Koordinate in ein offline-fähiges Kartenprogramm kopiert werden.

Zwei weitere Ansichten ermöglichen das Management von Kontakten, sowie die Konfiguration von Parameter, wie Adresse des Backend im lokalen Netzwerk und Text der SOS-Nachricht.

4 Heuristische Analyse der Gebrauchstauglichkeit

Eine erste Evaluation der Applikation ist mit Hilfe einer heuristischen Evaluation nach den 10 Design-Prinzipien von Nielsen [Ni93] erfolgt, um eine erste Einschätzung der Oberfläche und dessen Limitierungen zu erhalten. In Tabelle 1 sind die einzelnen Kriterien gelistet mit einer Zusammenfassung der Auswertung

Design-Prinzip	Zusammenfassung der Auswertung
1. Sichtbarer System-Status	Benutzer:in wird immer über notwendige, kontextspezifische Statusinformationen und dessen Änderungen informiert. Limitation des Systems: Benutzer:in wird nicht benachrichtigt, ob eine Nachricht das Ziel erreicht hat, oder gar gelesen wurde.
2. Übereinstimmung zwischen System und Realität	Der Begriff „Hofbox“ und das dahinterstehende Konzept ist der einzig sichere Fall von Fachjargon, der für nicht mit dem Projekt vertrauten Personen unklar ist.
3. Möglichkeit der freien Navigation	Das Versenden von Nachrichten, das Löschen des App-Speichers und das Entfernen eines Kontakts kann nicht widerrufen werden. Beim Löschen des App-Speichers und

Design-Prinzip	Zusammenfassung der Auswertung
	beim Entfernen eines Kontakts ist eine erneute Bestätigung erforderlich.
4. Konsistenz und Standardisierung	Die Beschreibung von Aktionen weist in der App eine hohe Konsistenz auf.
5. Fehlervermeidung	Mit dem Versenden von Nachrichten existiert eine Aktion die unwiderrufliche Konsequenzen hat, jedoch nicht von der Benutzer:in erneut bestätigt werden muss. Ansonsten werden mögliche fehlerhafte Eingaben abgefangen und Aktionen die zu Fehlern führen könnten, entsprechend verhindert und die Benutzer:in darüber in Kenntnis gesetzt.
6. Wiedererkennung geht vor Erinnerung	Im Chat ist der ausgewählte Kontakt direkt sichtbar. Darüber hinaus ist keine Erinnerung notwendig.
7. Flexibilität und Effizienz	Personalisierung der Benutzeroberfläche und die Abkürzung von Aktionen sind bisher nicht vorgesehen. Einzige Effizienzmaßnahme: Die Web-Oberfläche erlaubt es mit Strg+Enter eine Nachricht zu verschicken, um nicht den Senden-Knopf per Mausklick betätigen zu müssen.
8. Ästhetisches und minimalistisches Design	Die App besitzt ein minimalistisches Design. Eine logische Unstimmigkeit besteht in der Einstellungsansicht: Der „App-Speicher löschen“-Button wird im Netzwerksegment der Einstellungs-Ansicht angeführt, beeinflusst jedoch mehr als die Netzwerk-Einstellungen.
9. Information und Korrekturangebote bei Fehlern	Für fehlerhafte Eingaben wird jeweils der Fehler als auch ein Lösungsvorschlag präsentiert. Bei Aktionen die zu Fehlern führen könnten wird zwar der Fehler angemerkt, jedoch nur ein indirekter Lösungsvorschlag gegeben.
10. Hilfe und Dokumentation	Eine separate Dokumentation ist für die App nicht vorgesehen. Die App selbst verfügt an vielen Stellen über Informationssymbole, welche die Funktionalitäten der App erklären. Beispielsweise ist unterhalb des SOS Knopfes ein Informationssymbol zu sehen, welches bei Betätigung die SOS Funktionalität in einer Textbox beschreibt. Weiterhin kann in der Einstellungs-Ansicht beobachtet werden, dass für jedes Eingabefeld und jede Aktion ein Informationssymbol zur Verfügung steht, welche eine Textbox öffnet mit den jeweils notwendigen Informationen.

Tabelle 1: Ergebnisse der App-Analyse hinsichtlich der 10 Prinzipien nach Nielsen [Ni93].

5 Fazit

Die entwickelte App ermöglicht die Kommunikation mit einem modifizierten LoRaWAN-Netzwerk basierend auf Disruption Tolerant Networking. Sie kann mit Bundle Protocol 7 Pakete umgehen und bietet eine einfache Oberfläche, die sich an geläufigen Messengern orientiert. Das Ziel war es den Zugang zu dem Kommunikationsbackend möglichst einfach und nutzerfreundlich zu gestalten. Eine heuristische Analyse hinsichtlich der 10 Prinzipien nach Nielsen [Ni93] bescheinigt eine gute Gebrauchstauglichkeit. Die Oberfläche der App orientiert sich an gängigen Messengern und bietet Hilfstexte für besondere Funktionen und Einstellungen an. Wichtige Komfortfeatures sind soweit bekannt eingebaut, beispielsweise das automatische Neu-Verbinden mit dem Backend, sobald das Gerät wieder im korrekten Netzwerk (lokales Wi-Fi) eingewählt ist. Ebenso werden Nachrichten zwischengespeichert, solange noch keine Verbindung besteht. Die Integration von SOS-Nachrichten soll eine schnelle und einfache Kommunikation eines Notfalls ermöglichen. Mit Hilfe der Kriterien von Nielsen wurden einige Design-Probleme frühzeitig erkannt und umgebaut, so dass derzeit die heuristische Evaluation wenig Verbesserungspotenzial offenbart. Tests mit potenziellen Anwendern sollen in Zukunft die Gebrauchstauglichkeit der Applikation evaluieren und Änderungswünsche am gesamten System erarbeiten.

Bibliografie

[BFB22] Burleigh, S; Fall, K; Birrane, E. J.: Bundle Protocol Version 7 (Request for Comments Nr. RFC 9171): Internet Engineering Task Force, 2022

[BH20] Bormann, C.; Hoffman, P. E.: Concise Binary Object Representation (CBOR) (Request for Comments Nr. RFC 8949): Internet Engineering Task Force, 2020

[KBR23] Kuntke, F.; Baumgartner, L.; Reuter, C.: Rural Communication in Outage Scenarios: Disruption-Tolerant Networking via LoRaWAN Setups, In: Proc. of 20th Global Information Systems for Crisis Response and Management Conf. (ISCRAM), 2023.

[Ni93] Nielsen, J.: Chapter 5: Usability Heuristics. In: Usability engineering. Academic Press, Boston, ISBN 978-0-12-518405-2, S. 115–164, 1993.

[WU18] Waranashiwar, J.; Ukey, M.: Ionic Framework with Angular for Hybrid App Development. In: International Journal of New Technology and Research, Bd. 4, Nr. 5, 2018.