UAVNet: Prototype of a Highly Adaptive and Mobile Wireless Mesh Network using Unmanned Aerial Vehicles (UAVs)

Simon Morgenthaler
University of Bern

Dez 19, 2011
Outline

- Introduction
- Related Work
  - Mikrokopter
  - OM1P Wireless mesh node
- Architecture and concepts
- Implementation
- Demo Video
- Evaluation
- Conclusion / Future work
- Discussion
Motivation

- First response scenarios (avalanche, flooding, earthquake, ...)
- Automatically deployable and adaptive communication infrastructure
- Nodes position themselves autonomically according to the communication needs
Proposed Solution

- Unmanned Aerial Vehicles (UAVs) with attached wireless mesh nodes building an IEEE 802.11s wireless mesh network
- UAV platform: Mikrokopter.de
- Open-Mesh OM1P Wireless mesh node
- ADAM (Administration and Deployment of Adhoc Mesh networks)
Problem Statement

> Concept how a flying WMN should be designed, developed, deployed and maintained
> Implementation of a working prototype to show the feasibility of a flying communication network
> Use COTS equipment, open source software
> Keep the system cheap, compatible and lightweight
> Use an iPad/iPhone to deploy and monitor the network
> Evaluate the built prototype and the concepts
Mikrokopter.de Platform

- 4 brushless motors, controlled by 4 brushless controllers
- FlightControl
- NaviControl
- MK3Mag (3-axis compass)
- GPS
- 3 gyroscopes
- 3-axis acceleration sensor
- Pressure/height sensor

- “Hold position” / “Waypoint-Flight” / “Coming Home”
Open-Mesh OM1P Wireless Mesh Node

- Atheros AR2315 Soc, 180 MHz MIPS CPU
- 32 MB RAM, 8 MB NAND storage
- Atheros IEEE 802.11b/g wireless interface
- Serial interface

- Small (9.5 cm x 7 cm x 2.5 cm)
- Leightweight (86 grams)
- Low energy consumption
- Powerable by LiPo battery
Architecture and Concept
Scenarios

> Airborne Relay

> Multi-hop Airborne Relay

> Area Coverage (not implemented yet, future work)
Airborne Relay Deployment

1. START_CONFIG
2. OWN_POSITION
3. OWN_POSITION
4. DATA
5. Flight of the UAV
Presence Announcement of the UAVs

- Periodically broadcasted UDP packet
- Contains the UAV hostname, scenario, UAV state, positioning mode, searching mode, positions of the clients
- Clients can detect a near UAV and know the selected scenario, current UAV state, etc...
- UAVs can “hear/see” each other
Searching Modes

> Notebooks send their GPS coordinates
> UAV must fly in transmission range of the second notebook
> How to find the second notebook?

> Manual searching
  — User defines approximate flight direction and maximal distance
> Autonomous searching
  — UAV flies on a spiral track around the first notebook
Autonomous Searching Mode
Positioning Modes

- Positioning between two notebooks

- Location based positioning mode
  - Exact center position between the GPS coordinates of the two notebooks

- Signal strength positioning mode
  - Extends the location based positioning mode
  - Goal: same signal strength to both notebooks
Notification Subscription Service

- Clients / monitoring devices can subscribe to the service
- UAV periodically sends notification messages to subscribed clients / monitoring devices
- Notification message contains position, altitude, battery level, heading, speed and the status of an UAV
Remote Control App (Adrian Hänni)

> Network configuration and deployment
  — Configure the scenario, positioning mode, searching mode
  — Start the autonomous deployment process

> Monitoring a deployed network
  — Display UAVs and clients on an interactive map

> Reviewing saved flights
Remote Control App on an iPad
Implementation

> Assembly of two UAVs (quadrocopters)
  — Software of the flight electronics untouched
> Extension, adaption, configuration of ADAM / network
  — Ath5k, 802.11s, interfaces, iw, hostapd
> UAVNet software
  — Controller on the mesh node
  — Client on the notebook / smartphone
  — Libraries
  — Communication protocol
> Evaluation
Serial Connection Mesh Node ↔ UAV

> Serial interface on the mesh node (UART), 3.3V
> Debug port on the NaviCtrl of the flight electronics, 5V
> Logic level converter, 3.3V ↔ 5V

> Sending commands and waypoints to the UAV
> Requesting status and navigation data from the UAV
> Reading periodically sent data
ADAM

- New Kernel 2.6.37.6
- Compat-wireless package
- Ath5k replaces madwifi
- IEEE 802.11s (mac80211, cfg80211, nl80211)
- Bridged network interfaces (mesh0, wlan0, br0)
- iw
- hostapd
UAVNet: Prototype of a Highly Adaptive and Mobile WMN using UAVs

**UAVNet Architecture**

- **UAV 1**:
  - Engines
  - GPS
  - Sensors
  - FlightControl
  - NaviControl
  - libuavint
  - libuavext
  - uavcontroller
    - ADAM
    - Linux 2.6.37.6
  - Meshnode 1

- **UAV 2**:
  - Engines
  - GPS
  - Sensors
  - FlightControl
  - NaviControl
  - libuavint
  - libuavext
  - uavcontroller
    - ADAM
    - Linux 2.6.37.6
  - Meshnode 2

- **Client 1**:
  - libuavext
  - uavclient
  - OS
  - iPad

- **Client 2**:
  - libuavext
  - uavclient
  - OS
  - Notebook
Demo Video
Evaluation

> Evaluation of the performance of UAVNet
  — TCP and UDP throughput (netperf)
  — RTT measurements (ping)

> Evaluating the different parts of UAVNet
  — Optimal Signal Strength Threshold
  — Mesh Network Performance
  — End to End Throughput

> Comparison between UAVNet and ground-based approaches
Mesh Network Performance

> Measurements:
  — TCP / UDP throughputs

> Setup:
  — Multiple mesh nodes, with -60 ± 10 dBm signal strength between them
Mesh Network Performance Results

[Graph showing mesh network performance results with throughput in Mbps for 1, 2, and 3 hops.]
End to End Throughput

> Measurements:
  — TCP throughput between two notebooks, using an UAV as Airborne Relay

> Setup:
  — Notebooks 75m away from each other, UAV at different locations
End to End Throughput Results
Conclusion

> Proven feasibility of an autonomously deployable flying wireless mesh network
> Evaluation shows significant higher performance than ground-based approaches
> Starting point for future projects
Future Work

> Autonomous replacement and recharging
  — Autonomous landing and starting
  — Automatic network adaption

> Area coverage scenario
  — Self-organizing distribution
  — Collision avoidance

> Network performance improvements

> ORMAN (Opportunistic Routing for Highly Mobile Ad-hoc Networks)
Questions / Discussion
UAVNet: Prototype of a Highly Adaptive and Mobile WMN using UAVs
Communication Protocol

Start delimiter / End delimiter
Address: a=any, b=FC, c=NC, d=MK3Mag
Command: s=Send target position
Data: "modified-base64" encoded
CRC

# \r

<table>
<thead>
<tr>
<th>#</th>
<th>n</th>
<th>Notification_t</th>
<th>X</th>
<th>X</th>
<th>\r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start delimiter (1B)</td>
<td>Address (1B)</td>
<td>Command (1B)</td>
<td>Data (xB)</td>
<td>CRC (2B)</td>
<td>End delimiter (1B)</td>
</tr>
</tbody>
</table>

**Notification_t**
- Position: (GPS_Pos_t)
- Height: (uint16)
- Battery: (uint8)
- Heading: (int16)
- Speed: (uint16)
- Status: (Status_t)

**Status_t**
- Scenario: (uint8)
- State: (uint8)
- Positioning: (uint8)
- Searching: (uint8)
- Position Client1: (Submitted_Pos_t)
- Position Client2: (Submitted_Pos_t)
- Hostname: (char[])
Message Flow (Manual Searching Mode)

1. UAV / Mesh sends a HELLO message.
2. Client 1 receives the HELLO message and sends a START_CONFIG message (manual search).
3. Client 2 receives the START_CONFIG message and sends a DIRECTION message.
4. Client 1 receives the DIRECTION message and sends a DIRECTION message to UAV / Mesh.
5. UAV / Mesh receives the DIRECTION message and sends a HELLO message to Client 1.
6. Client 1 receives the HELLO message and sends an OWN_POSITION message to UAV / Mesh.
7. UAV / Mesh receives the OWN_POSITION message and starts to fly in the given direction.
8. Client 2 receives the HELLO message and sends an OWN_POSITION message to UAV / Mesh.
9. UAV / Mesh receives the OWN_POSITION message and starts to fly to the final position.
10. Client 1 receives the HELLO message.
Optimal Signal Strength Threshold

- Optimal distance between the UAVs?

- Setup:
  Multiple measurements between two mesh nodes with constantly decreased distance between them
  — TCP / UDP throughput
  — RTT
Optimal Signal Strength Threshold Results

- TCP throughput depending on signal strength

![Graph showing TCP throughput versus signal strength. The x-axis represents signal strength in dBm, ranging from -100 to -30. The y-axis represents TCP throughput in Mbps, ranging from 0 to 20. The graph displays a bar chart with bars at signal strengths of -100, -90, -80, -70, -60, -50, and -40 dBm, respectively. The bars indicate varying levels of TCP throughput at each signal strength.]
Optimal Signal Strength Threshold Results

- UDP throughput depending on signal strength

![UDP throughput diagram](image-url)
Optimal Signal Strength Threshold Results

- RTT depending on signal strength

![Graph showing RTT vs. signal strength](image-url)