The Ulm VANETStack

Frank Kargl

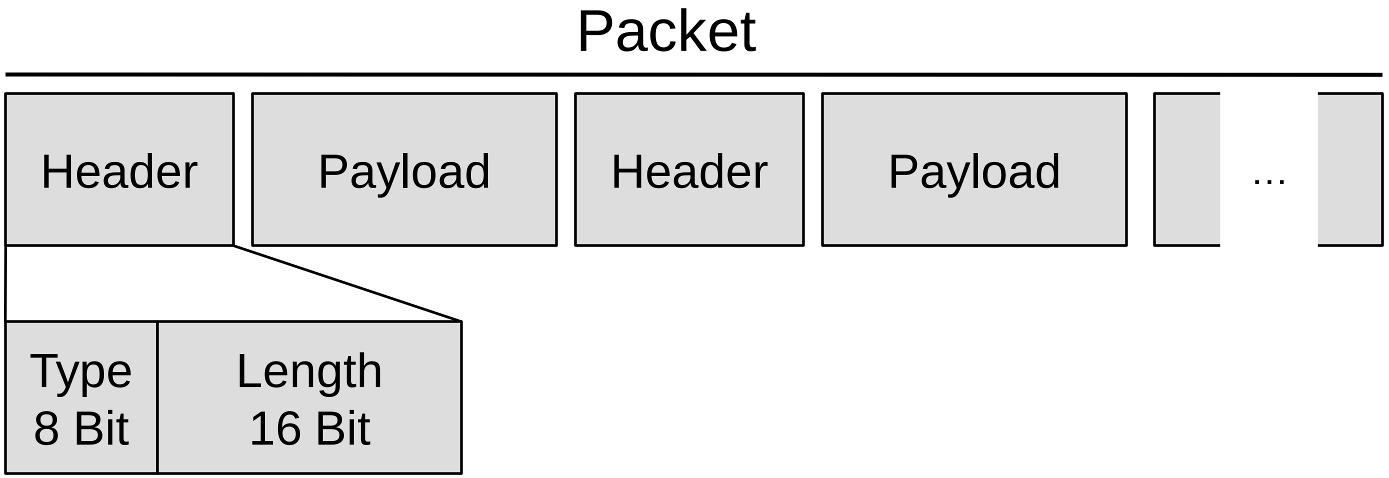
17. October 2007

Initial idea

- Two students (Michael Mosch and Fabian Wolf) doing a practical to implement a VANET communication system
- Original ideas
  - Running on a PDA, implemented using either a PDA-Java-VM or .NET compact framework and C# ➔ proved to be very complicated, as Java-VMs were very immature and .Net compact had problems with low lewel system access
  - Provide a very flexible framework so that e.g. security extensions are easy

VANETStack Packet Format

- Packet consists of chain of header-payload buckets
- Buckets are interpreted by corresponding modules



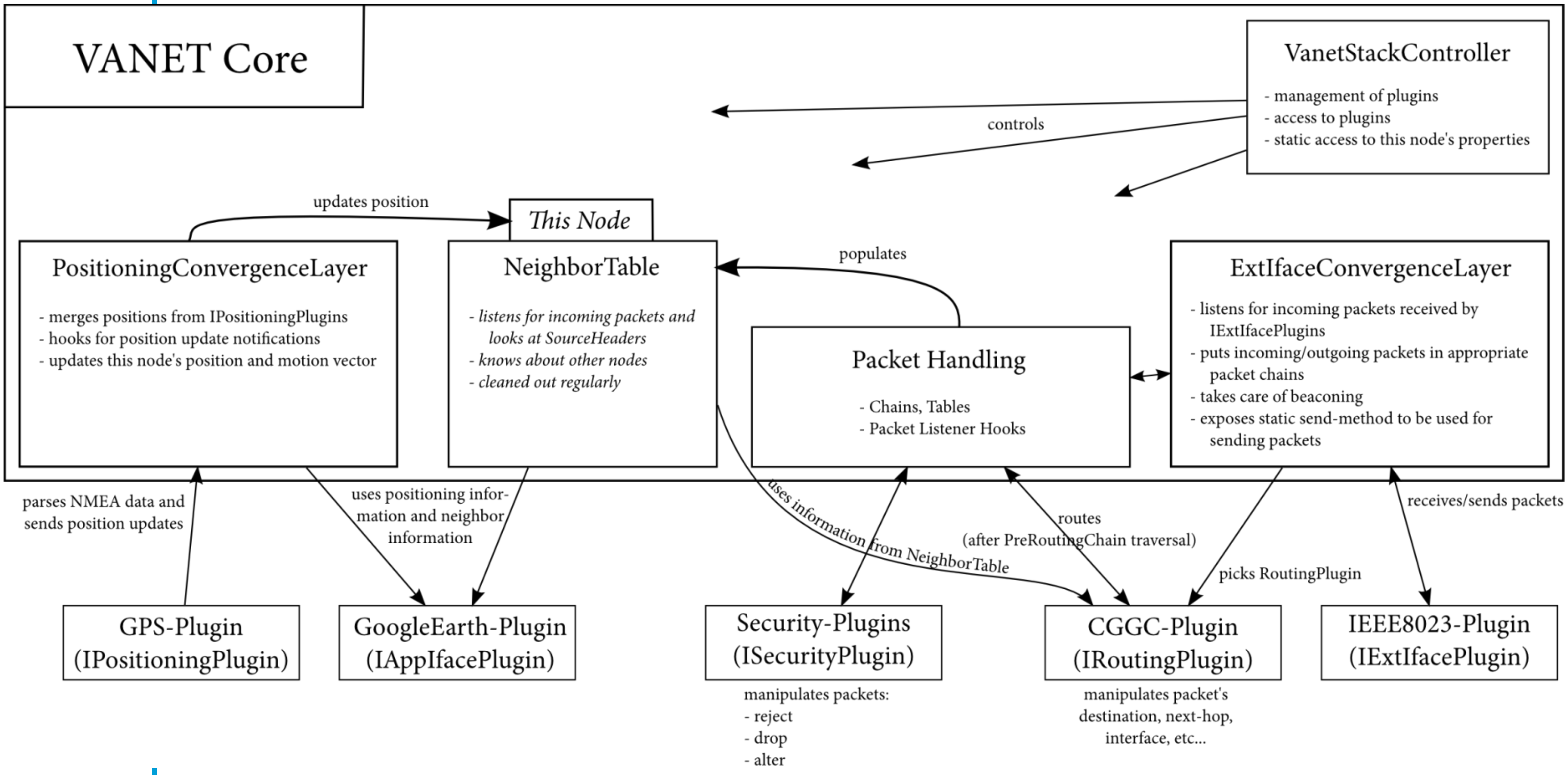
VANETStack Packet Format (2)

- Currently available types
  - Source Address (UID, packet#, timestamp, position)
  - Additional Source Information (speed, heading, …)
  - Destination Address
    (either UID or geographical area)
  - Checksum
  - Other misc payload (application dependant payload)

Architecture

- Based on modular and extensible
  Java Plugin Framework
  - Everything is a plugin
  - Plugins are accessed via well-defined interfaces
  - Existing plugins
    - Core – manage plugins, manage packet traversal, manage inter-plugin communication
    - Application – interface to stack-external components, can be e.g. an API, a socket interface, …
    - Positioning – manage position sources, currently implemented: GPS
    - Security – capture, modify, drop packets traversing through the stack (some other students start working on this)
    - Routing/packet mangling – determine next hop of packet. Currently implemented:
      beaconing, flooding, geocast, and position-based routing (CGCC)
    - External Packet Interface – abstraction layer to access different network interfaces. Currently implemented: 802.11 sending Ethernet frames using jpcap and lib/winpcap

Architecture (2)



VANET Core

VanetStackController
- management of plugins
- access to plugins
- static access to this node's properties

controls

updates position

This Node

populates

PositioningConvergenceLayer
- merges positions from IPositioningPlugins
- hooks for position update notifications
- updates this node's position and motion vector

NeighborTable
- *listens for incoming packets and looks at SourceHeaders*
- *knows about other nodes*
- *cleaned out regularly*

Packet Handling
- Chains, Tables
- Packet Listener Hooks

ExtIfaceConvergenceLayer
- listens for incoming packets received by IExtIfacePlugins
- puts incoming/outgoing packets in appropriate packet chains
- takes care of beaconing
- exposes static send-method to be used for sending packets

parses NMEA data and sends position updates

uses positioning information and neighbor information

uses information from NeighborTable

routes (after PreRoutingChain traversal)

picks RoutingPlugin

receives/sends packets

GPS-Plugin (IPositioningPlugin)

GoogleEarth-Plugin (IAppIfacePlugin)

Security-Plugins (ISecurityPlugin)

manipulates packets:
- reject
- drop
- alter

CGGC-Plugin (IRoutingPlugin)

manipulates packet's destination, next-hop, interface, etc...

IEEE8023-Plugin (IExtIfacePlugin)

Demo and Discussion