Model-driven Development of Complex Routing Protocols with SDL-MDD

Alexander Geraldy, Reinhard Gotzhein, and Christoph Heidinger
Networked Systems Research Group
University of Kaiserslautern, Germany
Context

• A generic routing protocol framework
• Design of ReBaC2/AodvLight
• Simulation results
• Conclusions
Context

- **Model-driven development (MDD)**
  - engineering approach where the formal model guides and directs all development activities
  - MDD activities range from system design over code generation and deployment to system maintenance.
  - MDD is a key approach to improving quality and productivity of system development.

- **SDL-MDD**
  - model-driven development with SDL as design language
  - addresses all phases of system development
  - supported by a semantically integrated and complete tool chain
Context

Process model

formal SDL semantics
SDL language profiles
ConTraST
SDL Environment Framework

SDL design patterns
micro protocols
micro protocol framework

ns+SDL, C-PartsSim
Context

• Complexity of routing protocols
  – different routing requirements (e.g., BE/QoS, proactive/reactive, …)
  – variety of address types (e.g., unicast, multicast, concast, anycast, broadcast, geocast, n-hop cast, …)
  – large variety of routing algorithms for each address type
  – different network types (e.g., wired/wireless, infrastructure-based/ad-hoc, stationary/mobile, WAN/LAN, high-speed/field bus, dense/sparse, …)
  – optimization potential (e.g., hierarchical routing, adaptive routing, nested routing, …)

→ Composition of self-contained, elementary routing protocols
Outline

• Context

⇒ A generic routing protocol framework

• Design of ReBaC2/AodvLight

• Simulation results

• Conclusions
A generic routing protocol framework
A generic routing protocol framework
A generic routing protocol framework

RoutingComponent

RoutingManager

RoutingAdapter

RouteDiscoverer

RoutingManager

RoutingAdapter

SendData

RecvData

RouteReq

RouteResp
Outline

• Context

• A generic routing protocol framework

➤ Design of ReBaC2/AodvLight

• Simulation results

• Conclusions
Clustering of mobile ad-hoc networks

• subdivide the network into sets of nodes called \textit{clusters}

• nodes of a cluster are \textit{cluster members}; there may be a distinguished \textit{cluster head}

• types of clusters
  – nodes with common parameters (e.g. small topological/geographical distance)
  – nodes with different capabilities (e.g. coordinator role, device role)

• hierarchical network topology $\rightarrow$ reduction of network complexity

• useful for efficient network management, hierarchical routing etc.
Repair-based Clustering (ReBaC2)

- dynamic, self-organizing process for network partitioning
- identical rules for cluster initialization and maintenance
  → graceful dynamic adaptation to topological changes
- modular clustering metrics (e.g. lower/upper bounds for cluster size, bounded cluster diameter, lower bound for link quality)
  → adaptation to specific network situations
- support for routing (logical tree structure of cluster members rooted at the cluster head, gateway nodes to neighboring clusters)
  → proactive collection of network status information during clustering
Ad-hoc On-demand Distance Vector Routing Light (AodvLight)

- reactive discovery of unidirectional routes (on demand)
- search by flooding of RREQ messages
- unidirectional response by RREPs

ReBaC2/AodvLight

- adaptive cluster establishment and maintenance
- proactive route discovery within clusters, based on tree structure
- reactive route discovery across clusters, by flooding RREQs to cluster heads
Design of ReBaC2/AodvLight

A. Geraldy, R. Gotzhein, and C. Heidinger, University of Kaiserslautern, Germany
Design of ReBaC2/AodvLight

RoutingManager

\( R_1 \)
\( R_2 \)
\( \ldots \)
\( R_n \)

RoutingAdapter

block type ReBaC2AodvLight

```
rebac2: ReBaC2
aodv: AodvLight
```

ReBaC2AodvManager

```
RecvData
RouteReq
RouteResp
```

ReBaC2AodvAdapter

```
RecvData
RouteReq
RouteResp
SendData
```

A. Geraldy, R. Gotzhein, and C. Heidinger, University of Kaiserslautern, Germany
Design of ReBaC2/AodvLight

RoutingComponent

RoutingManager

\[ R_{11}, R_{12}, \ldots, R_{1m} \]

RoutingAdapter

SendData

RecvData

block type ReBaC2

ReBaC2Manager

\[ \text{mgmReplies}, \text{mgmQueries} \]

\[ \text{RouteReq}, \text{RouteResp} \]

ReBac2Control

\[ (\text{toAlive}), \text{setCluster} \]

\[ (\text{fromAlive}), \text{aliveStat} \]

AliveSender

\[ \text{alive} \]

\[ \text{localAlive} \]

AliveReceiver

ReBaC2Adapter

\[ \text{RecvData}, \text{Register} \]

\[ \text{SendData} \]
Design of ReBaC2/AodvLight

RoutingComponent

RoutingManager

\[ R_{11} \quad R_{12} \quad \ldots \quad R_{1m} \]

RoutingAdapter

block type AodvLight

aodvManager(0,)

renaming

aodvRoutingCache

aodvRequestList

aodvRREQCache

srcSeqNo: SeqNo

destSeqno: SeqNo

aodvAdapter

SendData

RecvData

[RecvData]

[RouteReq]

[RoutResp]

[Register]
Outline

• Context

• A generic routing protocol framework

• Design of ReBaC2/AodvLight

➢ Simulation results

• Conclusions
Simulation results

- 56 nodes, randomly placed
- 400x400m, tx range 73m
  - cluster heads
  - cluster links
  - gateway links
- 8 clusters (size 3..13)
- max path length: 5 hops
  - physical links
  - overlay links
Outline

• Context
• A generic routing protocol framework
• Design of ReBaC2/AodvLight
• Simulation results

➤ Conclusions
Conclusions

• Model-driven development with SDL-MDD
  – applicable to complex routing protocols
  – tool chain fully operational and usable

• SDL sufficiently expressive, though not always elegant
  – self-contained and reusable micro protocol designs and components
  – complex routing architectures
  – “glue” for routing protocol composition
  – SDL representation of micro protocols depends on the intended composition
  – difficulties to find an adequate data representation for the generic addressing scheme (still SDL-96, due to available tool support)