

Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks

RESEARCH GROUP FOR

*Distributed
Systems*

Christian Frank

ETH Zürich, Switzerland*

Holger Karl

Universität Paderborn, Germany*

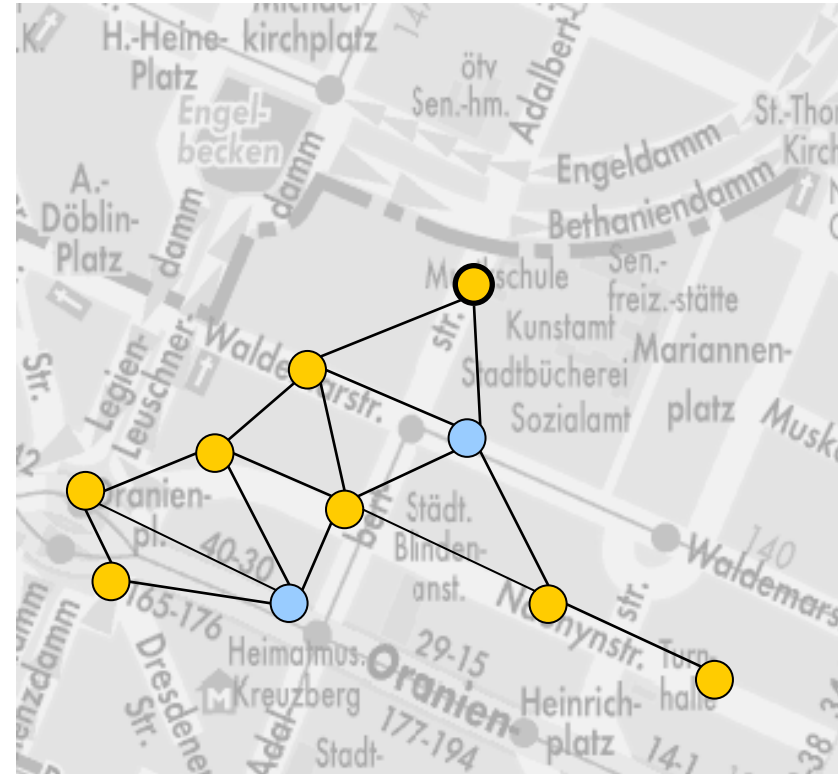
ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

*presented work created while both authors at
Technische Universität Berlin

Scenario

- Users form ad-hoc network
- Usage scenarios:
 - Find nearest store
 - Hail taxi cab
 - Connect to access point
- Stores, cabs, etc. publish services on the network
- Clients perform “Service Discovery“



© Mobilitätsverlag GmbH

Requirements

- Service-Discovery
- Efficient at times of few requests
 - Only act on-demand (reactive)
 - No requests → no network traffic
- Efficient when requests frequent
 - Prepare information in advance (proactive)
- Completely independent of infrastructure
- Post-discovery routing

Design Goals

- Re-use routing functionality:
 - On-demand (reactive) routing and service discovery similar
 - Request
 - Reply
 - Caching
 - Expanding ring search
- Extendable architecture
 - Exchangeable routing component

Related Work

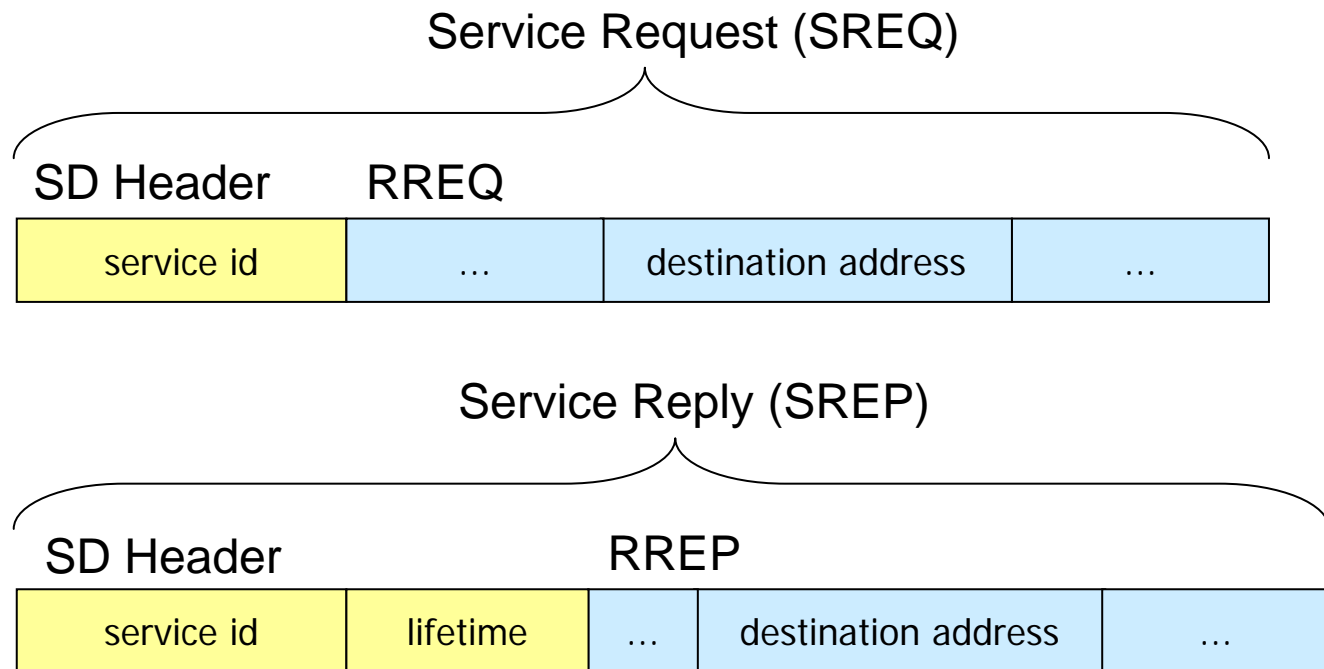
- Managed Networks
 - Service Location Protocol
 - UPnP
 - Jini Lookup Service
 - Intentional Naming Service (INS)
- Ad Hoc Networks
 - Multicast trees (e.g., L. Cheng, CSCW'02)
 - Integration with routing (Koodli, Perkins, *Internet Draft '02*)

Outline

- Introduction
- **Design / Implementation**
- Model
- Results
- Conclusion

Approach [kp02]

- Based on reactive ad-hoc network routing
- Attach service discovery headers to routing messages



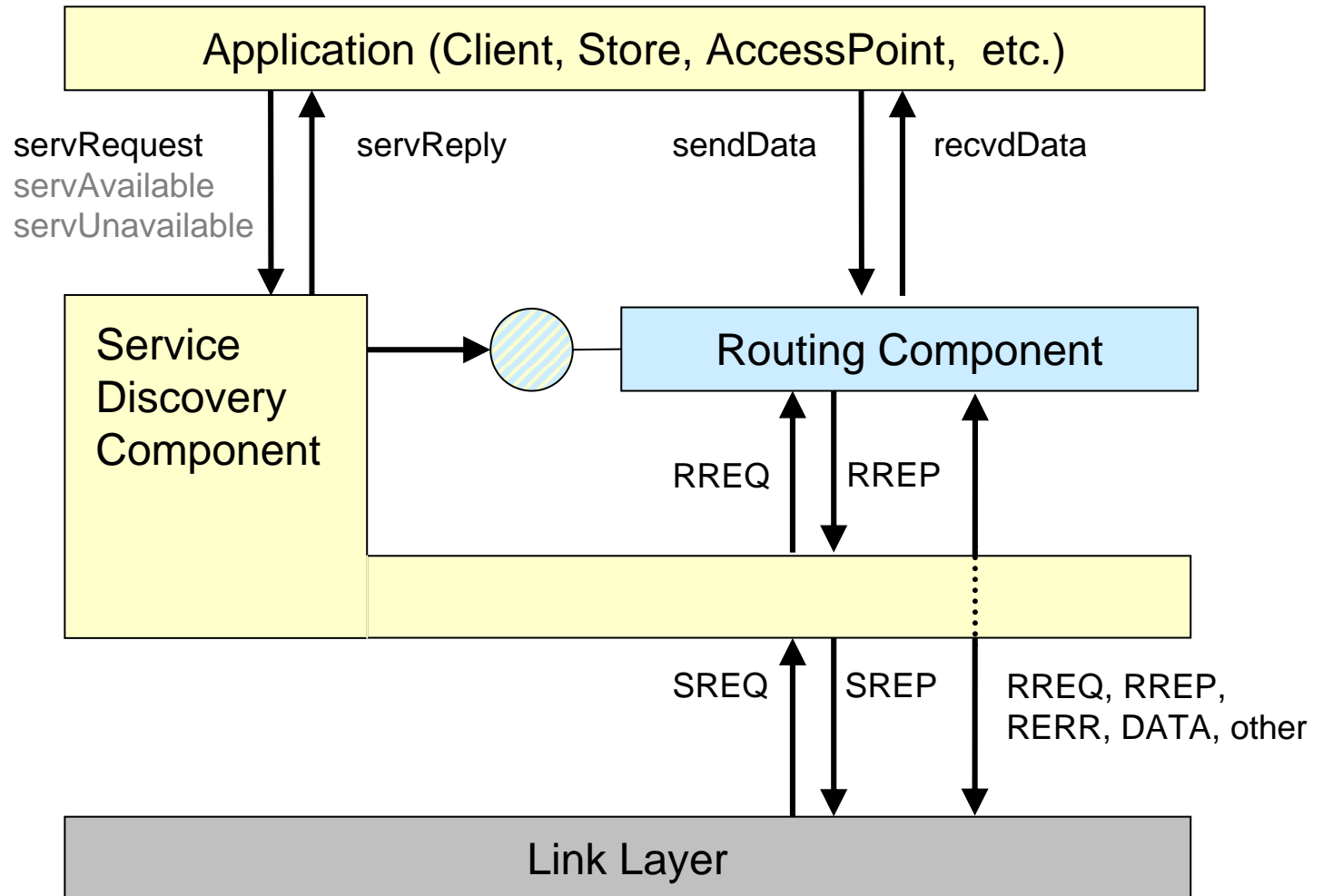
Approach [kp02]

- Each node maintains service bindings:

service-id	lifetime	provider
x	...	c
y	...	e
y	...	f

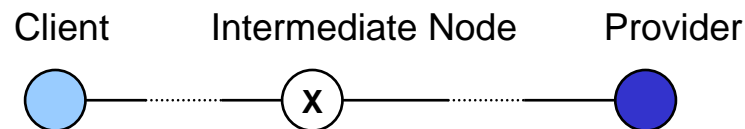
- SREQ flood
 - Destination in RREQ message is initially empty
- If node has binding for sought service
 - insert *provider as destination* and forward
- Nodes storing a route to destination will reply SREP
- Client receives SREP
 - Client knows provider and route to provider

Architecture

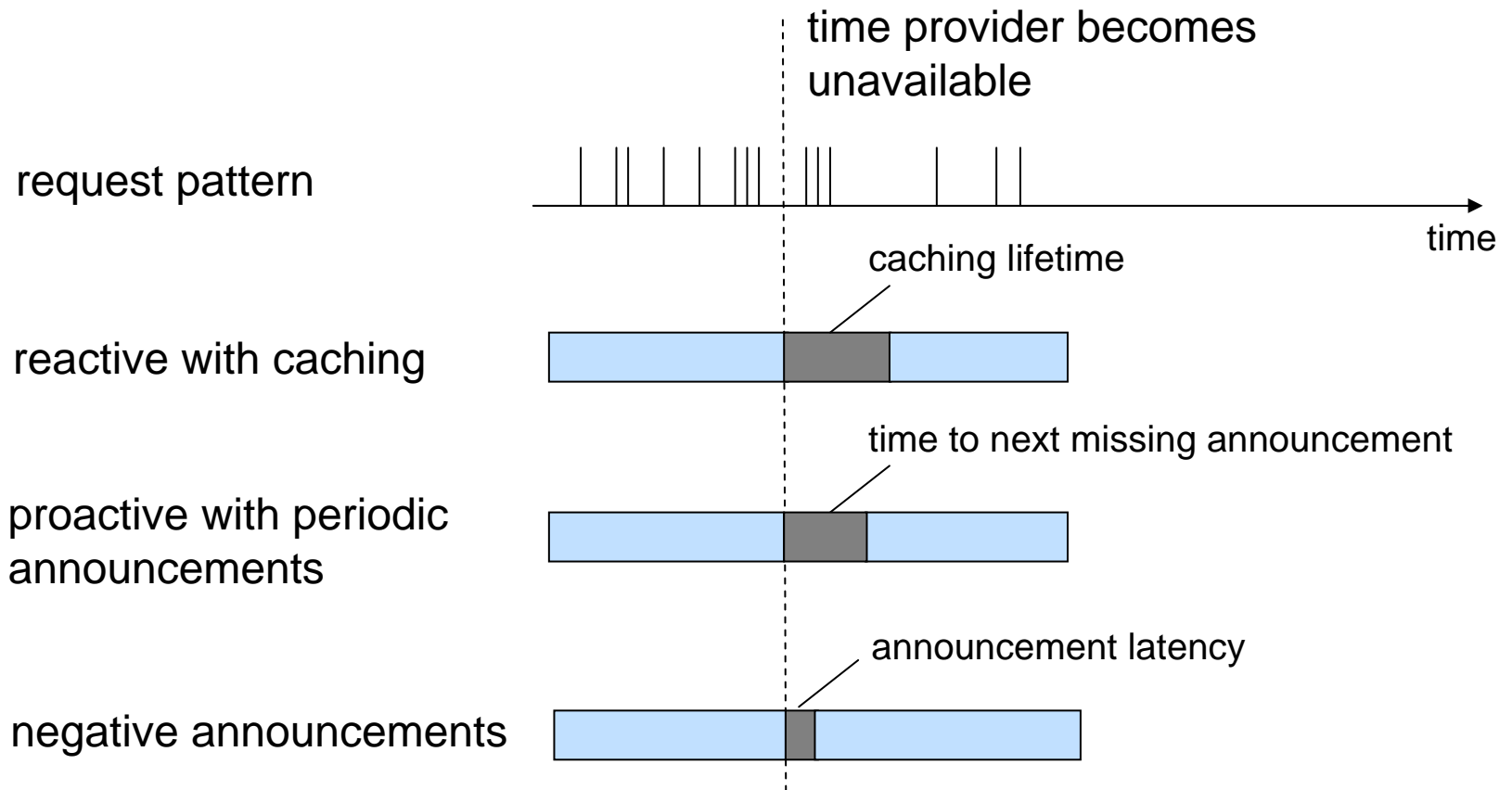


Proactive Elements?

- Be more efficient for frequent requests
- Periodic Announcements?
 - Caching enough with frequent requests
- Negative Announcements?
 - Good when providers become **actively** unavailable

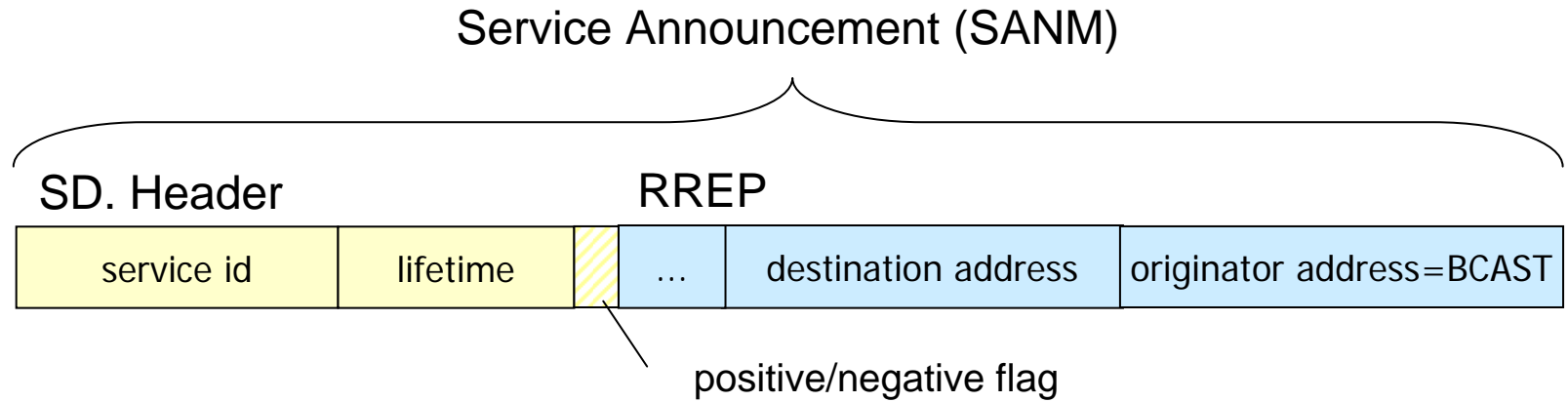


Provider Unavailable



- Benefit grows with #requests

Announcement Message



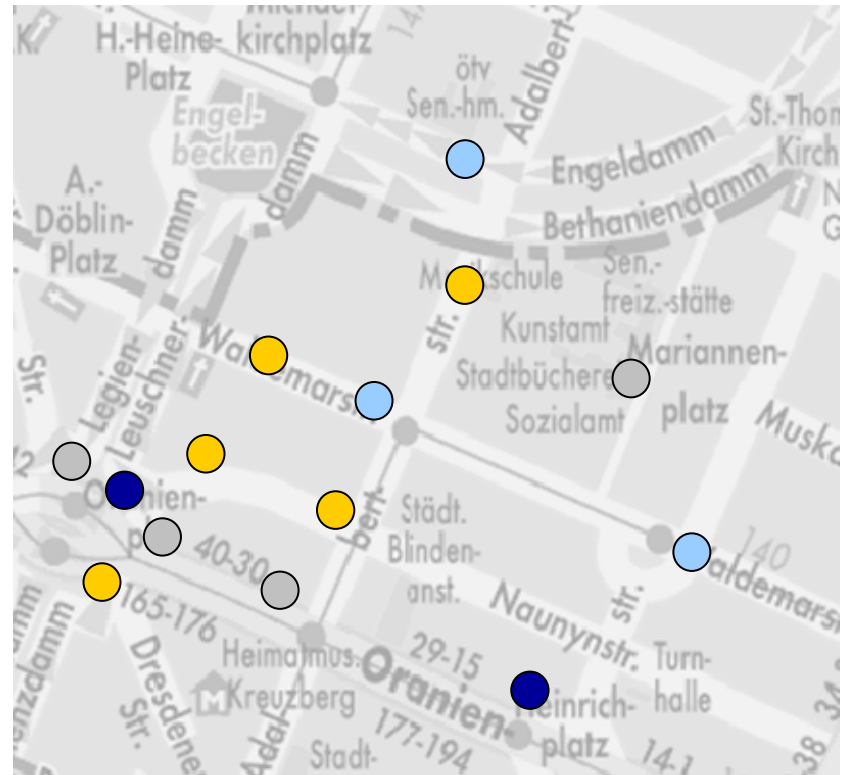
- Very similar to SREP
- Provider initiates SANM flood:
 - Service disc. header adds / removes service binding
 - RREP part creates route
- Nodes only forward if own decision is affected

Outline

- Introduction
- Design / Implementation
- **Model**
- Results
- Conclusion

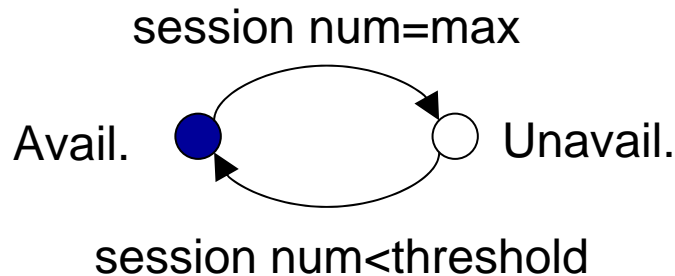
Model: Applications

- **Clients**
 - Inter-request time ~ exponential
 - Mobile
- **Stores, institutions**
 - Constant service availability
- **Cabs, bike couriers**
 - Very volatile service availability
- **Internet access points**



Access Point Providers

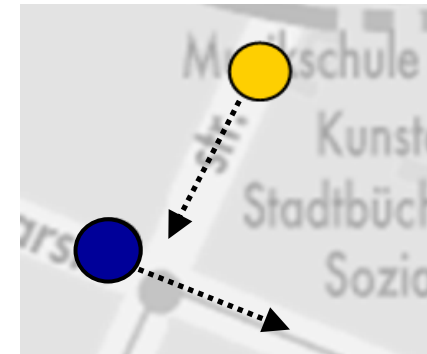
- Clients are engaged in sessions
- Provider becomes unavailable if max. capacity is reached



- Becomes available if capacity free (threshold)
- Session length $\sim \text{gamma}(\alpha, 1)$
- Traffic during session

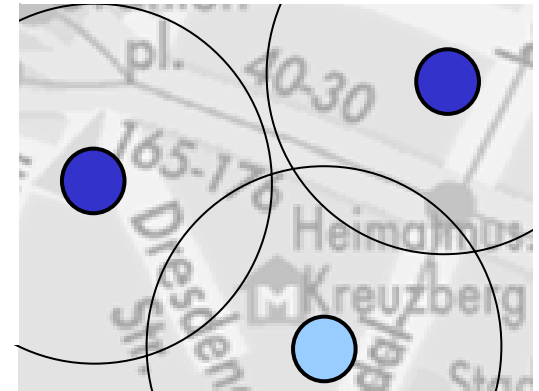
Model: Mobility

- Variant of Random Waypoint Model
- Participants travel to random destination
 - Speed \sim normal(μ , σ)
 - Different μ , σ for provider types
- Minimum speed



Model: Network

- Physical Layer
 - Simple path loss \rightarrow radius
- Link Layer
 - Broadcast and unicast message delivery
 - Latency based on message length
 - MAC modelled by random delay



Outline

- Introduction
- Design / Implementation
- Model
- **Results**
- Conclusion

Simulation: Access Point Providers

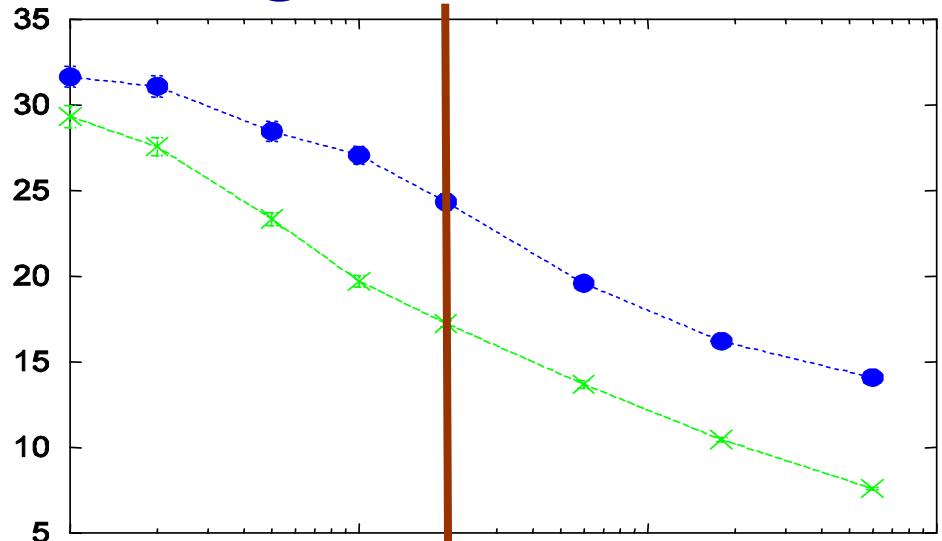
- Study protocol behaviour with AP
- 800m x 800m area (see paper f. parameters)
- AODV as routing component
- Prominent metrics:
 - Messages per request
 - Ratio of incorrect replies
- Runs:
 - Increase **cache lifetime** (under high load: 50 clients and 5 AP)
 - Increase **number of requests** (with a given lifetime)

Method

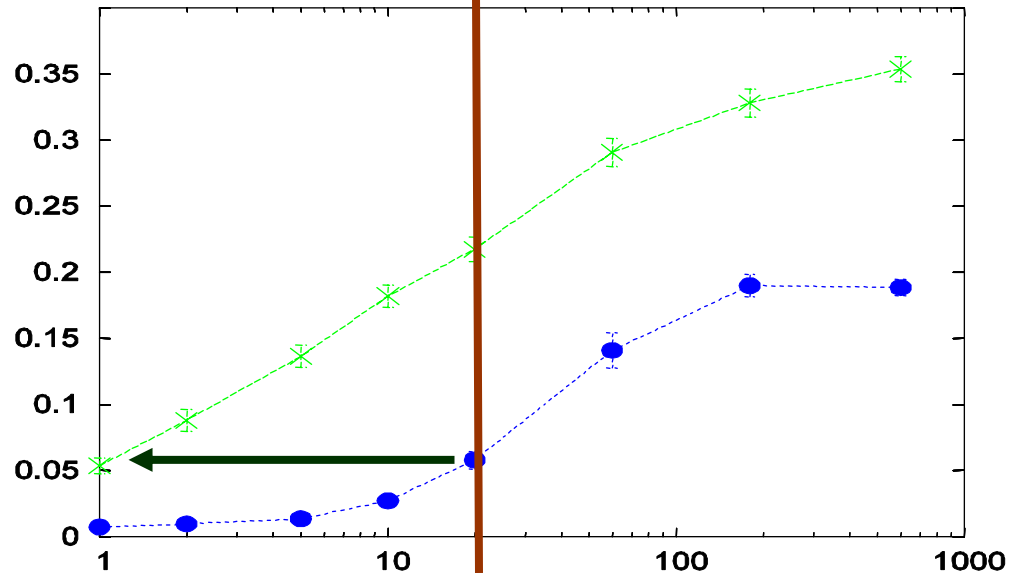
- Implementation in C++
 - Discrete Event Simulator (Omnet++)
 - TKN Mobility Framework
- Transient removal with Akaroa
- Parallel runs until reaching:
 - confidence level of 95%
 - 5% precision (or 0.05)

Varying Caching Lifetime

Messages per Request (Sent)



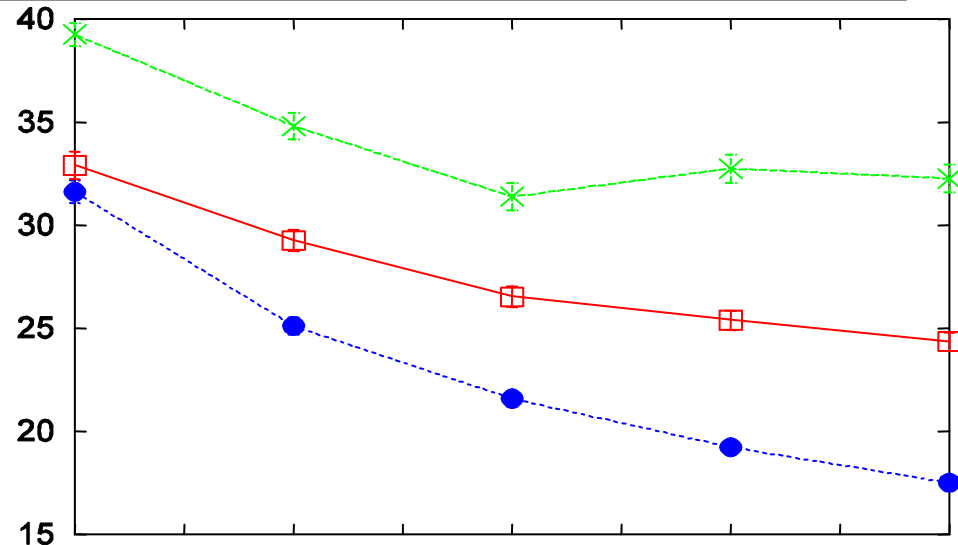
Incorrect Replies



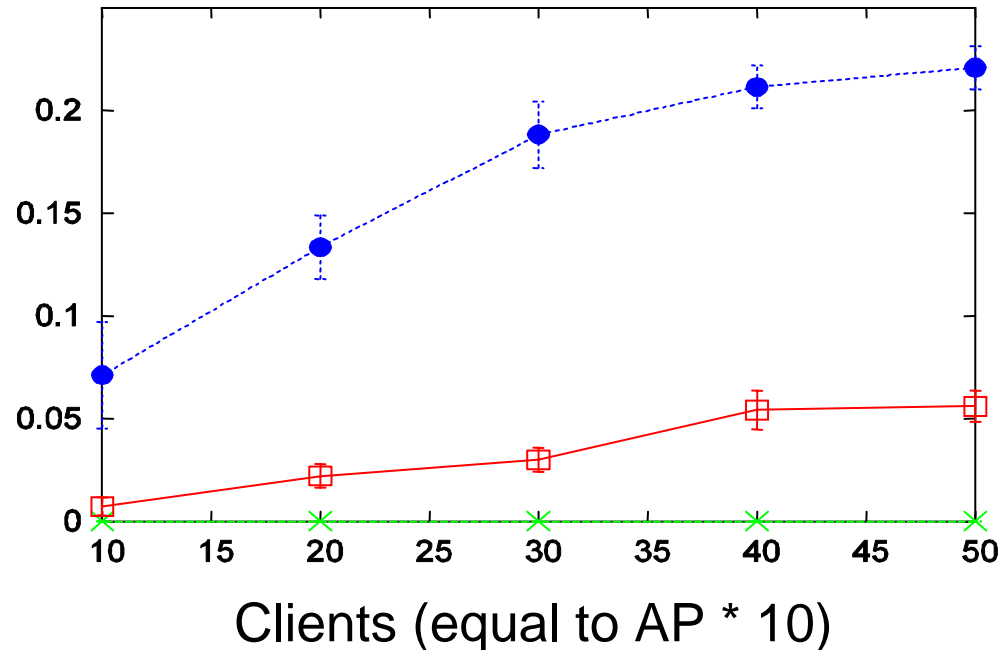
---x--- caching
---●--- negative ann.

Varying Network Load

Messages per Request (Sent)



Incorrect Replies



- no caching
- caching (20s)
- negative anomaly

Results

- Reactive without caching:
 - Most consistent
 - Least performant
- Reactive with caching:
 - Least consistent
 - Most performant
- **Negative Announcements**
 - Good balance
 - Better than customized lifetime

Conclusion

- Implemented and evaluated Internet Draft [kp02]
- Modular architecture for integration of routing and discovery
- Negative announcements perform well:
 - Make no difference for **STORE** providers
 - Provide consistency for **CAB** providers
 - Balance between performance and consistency for **ACCESS POINT** providers
- Explicitly removing cached entries is well-invested effort

Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks

RESEARCH GROUP FOR

*Distributed
Systems*

Christian Frank

ETH Zürich, Switzerland

Holger Karl

Universität Paderborn, Germany

Thank you!
Questions?

chfrank@inf.ethz.ch

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich