Supporting Transitions between Publish/Subscribe Mechanisms in Mobile Ad Hoc Networks

Master-Arbeit

Alexander Wagener
KOM-M-0481

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Supporting Transitions between Publish/Subscribe Mechanisms in Mobile Ad Hoc Networks
Master-Arbeit
KOM-M-0481

Eingereicht von Alexander Wagener
Tag der Einreichung: 29. November 2013

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz

Betreuer: Björn Richerzhagen, M. Sc.
Christian Groß, M. Sc.

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die schriftliche Fassung stimmt mit der elektronischen Fassung überein.

Darmstadt, den 29. November 2013 Alexander Wagener

Abstract

Publish/subscribe (pub/sub) systems have received an enormous amount of attention on the Internet and mobile applications. Even if they cannot be seen directly, the design principles of pub/sub systems are highly integrated and used in daily used systems and applications.

Many researches and studies in the past have grown a large number of different pub/sub mechanisms with a multitude of features for different kind of use cases and scenarios. All these systems are very specialized on their own fields and there is always a best in class approach for a different area of communication systems. The remarkable fact of these systems is, that pub/sub mechanisms provide solutions for problems encountered in the field of future mobile Internet, by providing new architectures and aspects that need our attention. While developing a new pub/sub protocol, which might summarize all features of the different existing systems, is prohibitively expensive and hard to achieve, this protocol would again be inflexible and poorly expandable, especially on future discoveries and inventions in this fast evolving research field.

A system which covers, controls, and manages the peculiarities of those individual systems in a generic way allows that each pub/sub system used remains intact in the overall environment. The overall system can still react to the different environmental and internal situations the original pub/sub mechanisms were designed for. The concept of such a system has to identify and abstract common properties of existing pub/sub systems in generic modules and build an architecture that integrates the different approaches into one usable and stable system. Which steps are taken to build such a generic pub/sub system and which tasks have to be performed are also questions that must be answered.

This master thesis describes and evaluates a transition-enabled pub/sub service, which offers the capability to switch between an arbitrary amount of different pub/sub mechanisms at run-time, to take full advantage of these individual mechanisms. The idea behind this approach, challenges, design, and implementation of a prototype are investigated and the performance of the system is evaluated. This work also covers problems and opportunities which might appear during the design and implementation of a transition-enabled pub/sub system. Limitations and capacity restraints of such a system and the prototype under various environmental conditions are also presented and discussed in this work.

Contents

List of Figures

List of Tables

1 Introduction and Motivation

Although the concept of a publish/subscribe (pub/sub) system is long known and used as a foundation for many distributed systems and applications, with the recent heavily use of social applications like Twitter[1], Facebook[2], and Google Alerts[3], pub/sub systems have received increasing attention on the Internet and social media.

Pub/sub systems are not only used from the information technology point of view, they also provide a structure for networks on the transmission layer, and thus bring many benefits for them. The characteristics of the pub/sub communication paradigm allows the construction of overlays, which differ from the well known end-to-end semantic of communication networks. These systems have proven to provide particular advantages, especially for mobile networks, where resources, such as bandwidth or capacity on the individual devices, are limited. Also worth noting, many pub/sub systems that are known and used today, follow different goals and approaches. In case of a heavily evolving environment, these systems can only partially adjust to the new circumstances. Depending on the given pub/sub system, adjustment to a new situation can be or cannot be accomplished by the pub/sub system. However, it is likely that a pub/sub system, that copes with this task particularly well, is very complex and expensive to develop and run.

This master thesis investigates the advantages and disadvantages of a transition-enabled pub/sub system for the use case of a mobile ad hoc network (MANET). The transition-enabled pub/sub overlay enables the possibility to switch between different pub/sub mechanisms to combine their advantages among different environmental conditions or approaches for which they were designed. Figure 1.1 shows how different pub/sub overlays perform under different environmental or internal conditions. The idea behind a transition-enabled system is to switch to the mechanism that performs best in the current scenario and therefore use its design to perform its best in the given situation.

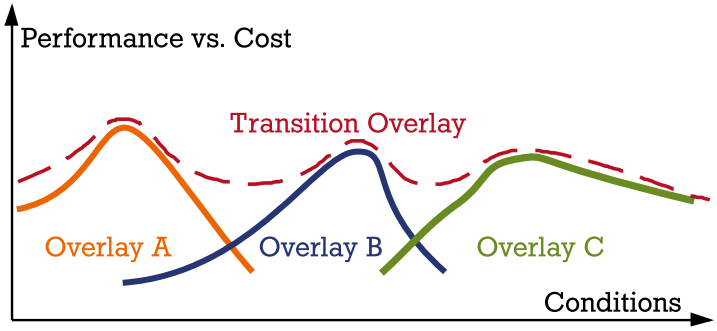


Figure 1.1.: Characteristical performance curves of pub/sub overlays under different environment conditions. The y-axis shows the generic performance versus cost of an overlay, while the x-axis shows the range of generic environmental conditions. A transition-enabled pub/sub overlay covers the whole performance spectrum of the used overlays.

A transition-enabled pub/sub system has to make decisions based on the available data to always use the best overlay possible. A design of such a system has to identify and abstract common properties of existing pub/sub systems and build an architecture that integrates the different approaches into one

[1] `http://twitter.com` [Accessed 28/11/13]
[2] `http://facebook.com` [Accessed 28/11/13]
[3] `http://google.com/alerts` [Accessed 28/11/13]

usable system. Thereby, it has to consider the specific properties of MANETs. This allows to use relative simple systems, which are highly specialized in one area, rather than cover every possible application with a very complex pub/sub system.

Future Internet is a term that describes research, technologies, and new architectures for the Internet. Especially the field of mobile networks and "Internet on the Go" is an area, which is intensively discussed and studied. Pub/sub systems do fit nicely in this research field, which is called Internet of Things (IoT) [ABB+09]. They summon opportunities to improve existing networks with extended mechanisms and services, that makes this area so exceedingly interesting. This thesis follows the basic constitutive ideas of Richerzhagen et al. [RS13]. The newly established collaborative research centre "Multi-Mechanismen-Adaption für das künftige Internet" (MAKI) at TU Darmstadt creates new conditions for future communication systems, which includes the results of this work.

1.1 Goal and Contribution

The goal of this thesis is to evaluate opportunities for transitions in pub/sub systems with the focus on overlay topologies for MANETs. A transition mechanism has to be designed and implemented, providing the possibility to switch between different pub/sub overlays on the network at run-time. The system, the impact and benefits of transitions under various scenarios will be evaluated through simulative studies. The results of these studies are discussed critically and conclusions are drawn decisively from it for further actions. The following contributions are stated in this thesis in order to reach the described goal.

- A thoroughgoing analysis of existing pub/sub solutions for MANETs, with a focus on currently used approaches and mechanisms, as well as support support for peer mobility. This also includes pub/sub solution concepts for fixed networks and middleware architecture, which are compared with each other to derive the necessary design decisions for a transition-enabled pub/sub system.
- Conception and design of a transition-enabled pub/sub architecture which supports transitions between such pub/sub overlays at run-time. The design considers in particular the distributed decision-making and self-analysis of well chosen network and overlay parameters.
- A prototypically implementation of the architecture in Java for the simulator PeerfactSim.KOM, as well as the implementation of a few standard pub/sub overlays, derived from existing pub/sub systems, to supply the transition-enabled pub/sub system with overlay references. This shows the feasibility of such a system and provides a prove of the concept.
- An effective analysis of the proposed system in simulative studies, as well as the evaluation of the system behaviour under different scenarios and workloads in the simulation, showing the impact of transitions between existing pub/sub systems.

1.2 Outline

The thesis is structured and partitioned as follows. In Chapter 2 relevant background on the pub/sub topic and on MANETs is given. Also the various pub/sub concepts and approaches are discussed. In Chapter 3 related work, such as existing pub/sub systems are presented. This includes pub/sub architectures for fixed networks and MANETs, as well as middleware. In Chapter 4 requirements are derived for a transition-enabled pub/sub system, based on the properties of the investigated systems. In Chapter 5, a design for a transition-enabled pub/sub system is presented, and afterwards in Chapter 6 the implementation of a prototype for the network simulator PeerfactSim.KOM is shown. The implementation, performance, and the fulfillment of the requirements are evaluated in Chapter 7, which includes simulative studies of the prototype. Finally, in Chapter 8 conclusions and the benefits of the proposed system are drawn and interesting questions are provided for future work.

2 Background

This chapter provides general background information about pub/sub systems and architectures, as well as an overview on adaptive complex systems. It provides an interesting insight on MANETs. Terms and notations that are required for the understanding of the different pub/sub mechanisms are prepared and explained. Mobile networks are used where communication between computing devices is not practical or usable. This freedom comes with some costs on the side of performance and complexity of those systems, that have to be paid. At this point, pub/sub systems also come into play. They offer mechanisms addressed to problems that MANETs cannot solve alone and some even use the special characteristics of a mobile environment to carry information through the network.

2.1 Mobile Ad Hoc Networks

This section provides a short overview about mobile networking. Since the developed transition-enabled pub/sub overlay is based on wireless network concepts, it is important to understand the relationships between the design decisions of the pub/sub system and technical operation of a MANET.

According to [CCL03], MANETs are complex distributed systems with network nodes[1] that are loosely coupled, in a composition to exchange information. This allows users of such a network to set it up, without the use of additional hardware or stationary components such like an access point (AP). Due to the fact that no further infrastructure is needed, the participating nodes can move freely within a range depending on the used physical connection and multi-hop routing concept. A bluetooth [ECC+99] connection allows to setup information exchange between devices in a short range of a couple of meters. But the standard, without an extension, does not support communication over multiple hops and therefore does only work from node to another node directly. The span of an array of multiple nodes is limited by this factor.

Wireless local area network (WLAN) technology, such as the standard IEEE 802.11 [IEE97] enables the possibility to build large networks with hundreds or even thousands of nodes. This allows to build a city wide Wide Area Network (WAN) with a large covering range. Other mobile communication networks like Global System for Mobile communications (GSM) [RWO95] or Long Term Evolution (LTE) [DPSB10] require the existence of expensive base stations, however these networks are commonly used to bring the mobile Internet to the user. There are pros and cons for each of these systems, depending on the use-case and application. The industry is looking for low-cost and high performance solutions that allows to connect mobile devices like smart phones and laptops.

MANETs have one thing in common, and this is also the largest difference to wired network technology. Wireless network technology shares the same transmission medium, regardless of the wireless network type or affiliation to a compound structure. So, even if two different mobile networks do not share the same interests, they still have to take care about medium access control (MAC). In addition, this medium also has only limited capacity. Different mechanisms are developed to encounter the fact of a shared medium, such like frequency division multiplexing (FDM) or time division multiplexing (TDM) [TW11]. These can be of any complexity to raise the bandwidth of a wireless network. Consequently, if an application can reduce the the impact on this, it would help to increase the performance of the overall MANET. Pub/sub overlays can support this by managing data depending on their type of information, destination and the handling of produced events.

[1] The terms peer and node are used interchangeable in this thesis.

2.2 Publish/Subscribe Concepts and Architecture

In this section, the basic idea and properties of the pub/sub communication paradigm, and its benefits for the deployment in MANETs is discussed.

As defined in [EFGK03], pub/sub is a communication paradigm, where information consumers and producers are linked together by intermediate brokers. This stays in contrast to traditional communication schemes like Remote Procedure Call (RPC) [TW11], where the client has to pull data from the server. This is particularly unfavorable if lots of data has to be delivered to many receivers and information or data packets are not generated very often. The idea behind pub/sub is, that peers with data push it to interested parties. Pub/sub forms a paradigm which delivers messages to interested subscribers. Through adaption of event-based systems, pub/sub becomes a powerful tool for aggregating, filtering, and composing of information units.

Consumers of data initially subscribe for a type of information they are interested in by using a `subscribe()` operation on the pub/sub service. Typically consumers do not know the effective source of the data they receive. Clients can terminate a subscription using an `unsubscribe()` operation. Events which are generated by a publisher can be published using a `publish()` operation on the service. The service propagates the event to all registered subscribers. Depending on the application, these messages contain mostly status information, but also can carry complex data streams. Messages can be used to adjust the routing mechanisms of the service or inform subscribers that new data is available.

Three dimensions (space, time and synchronization) are defined, in which the pub/sub service decouples publishers from subscribers and vice versa. Figure 2.1 illustrates that publisher and subscriber do not need to exactly know each other.

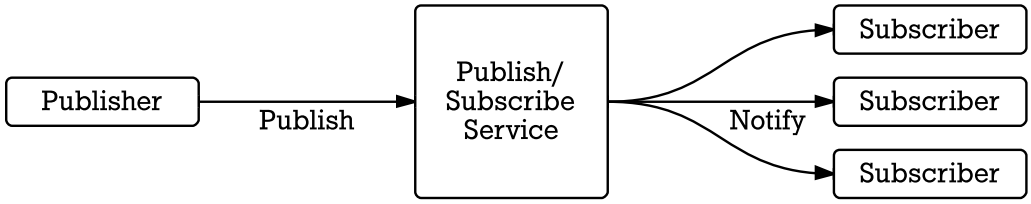


Figure 2.1.: Space decoupling provided by a publish/subscribe service (adapted from [EFGK03]).

The event service decouples them by splitting the end-to-end semantic, so that no reference has to be maintained on any side. A publisher has to publish an event only once, regardless of how many subscribers subscribed to a type of information. The pub/sub service now is responsible to notify each subscriber about that event. The second decoupling mechanism is shown in Figure 2.2.

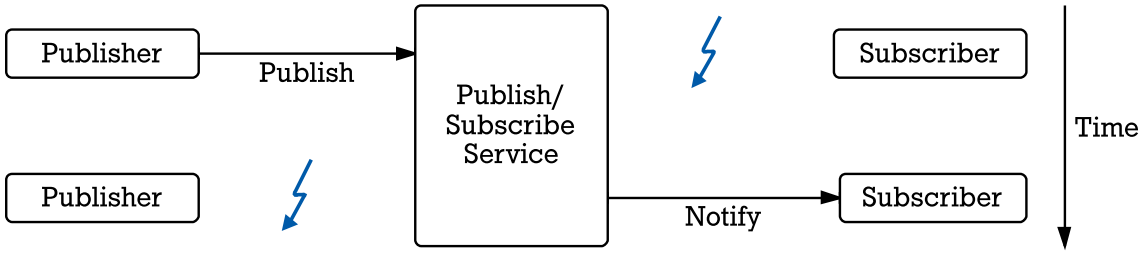


Figure 2.2.: Time decoupling provided by a publish/subscribe service (adapted from [EFGK03]).

The pub/sub service separates the interaction of publishers and subscribers in time. A publisher may publish an event at any time, even if a subscriber is not connected to the service. The subscriber might also get notified about an event, even if the original publisher of the event is disconnected. This is another step to dissolve the usual end-to-end semantic. Synchronization decoupling as illustrated in Figure 2.3

ensures that publishers are not blocked while producing events and subscriber receive events that match the subscription asynchronously as they are generated by the producers. The main control flow is not interrupted as only messages are generated and no waiting for a callback as in RPC is necessary.

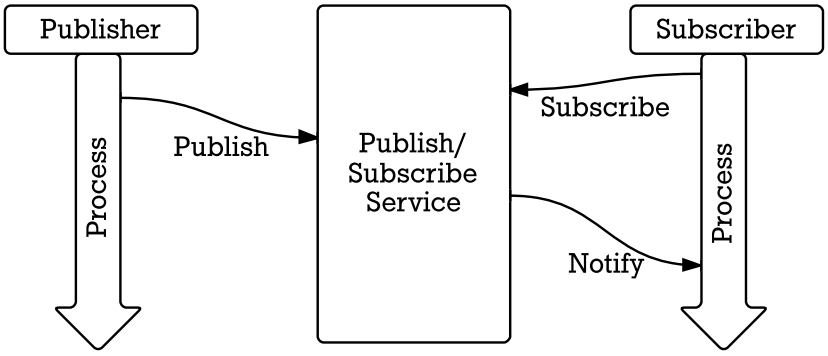


Figure 2.3.: Synchronization decoupling provided by a publish/subscribe service (adapted from [EFGK03]).

According to [HGM04], removing synchronization and dependencies between entities of the network does greatly adapt the communication infrastructure to distributed environments, like mobile networks. Due to the authors, this is caused by several reasons. The anonymity and dynamism of pub/sub allows the systems to adapt quickly to frequent connections and disconnections of mobile nodes. Also wireless devices have limited capabilities and bandwidth. Later, the multicasting nature of pub/sub systems will be discussed, but already at this point, it is unambiguously shown that this will help systems to scale to a large amount of units. As stated, the asynchronicity of pub/sub is also helpful, since mobile devices are often turned off or disconnected from the network for longer periods of time. Pub/sub systems therefore greatly adapt to the mobile environment due to its architecture and mechanisms.

Pub/sub systems can mainly be differentiated by their three essential properties, the subscription model, the used routing algorithm, and the underlying topology. First, the underlying topology of an pub/sub system is analysed. There are three known architectures of a pub/sub system.

Network Multicast This architecture uses multicast networking facilities at the data link level. A multicast message is addressed to multiple destinations. End-to-end multicasts also exist in form of Internet Protocol (IP) multicast schemes. These schemes are very scalable for large multicast groups. Multicast notification services like XCAST [BFI+07] use this concept to provide a similar scalable multicast service to transmit data packets with less effort than regular point-to-point communication.

Broker Networks Brokers are nodes located between publishers and subscribers, which act as handlers of subscriptions and publications. They are responsible for carrying information about subscriptions and publications through the network. Broker networks are based on transport level connections between nodes, meaning end-to-end communication which might include multi-hop connections on lower layers. Broker networks are hierarchical, forming a decision tree from the publisher to subscribers. An undirected acyclic graph is spanning all brokers.

Structured Overlay A structured pub/sub overlay sits between the transport layer of the network and the application. It provides additional functions which are highly specialized on the network type by abstracting from physical nodes. For MANETs they use information, which is only useful in context of mobile environment. These structured overlays can be conducted centralized and decentralized. The transition-enabled pub/sub system will be a structured pub/sub overlay.

Second, the used routing algorithm is usually executed as filter-based routing or rendezvous-based routing. Filter-based routing uses lists or filters to determine if a message is forwarded to a specific neighbour or discarded. These filters try to identify possible events that are not interesting for any

subscriber and stop the propagation of these events as soon as possible. Instead, explicit paths between publishers and subscribers are established, while maintaining the routing information at the brokers of the pub/sub overlay. Sets of filters are maintained at an entry broker to encode the information about reachable nodes through that broker. This approach is very similar to Ad hoc On-Demand Distance Vector (AODV) routing [PBRD03] which is used in MANETs to efficiently build up routing information on every node. Analog to AODV, filter-based pub/sub routing goes through three main phases. In the first phase, subscriptions (filters) are propagated to every broker leaving state information distributed along the path. In the second phase, subscriptions matching notification events follow the path set by the filters backwards from the broker to the subscriber. In the last phase, the propagation of subscriptions and notifications is reduced by avoiding paths on which a subscription was already forwarded. The efficiency of filter-based routing increases with the number of brokers in the pub/sub overlay, since notifications that do not match any subscription can be filtered out earlier on the path through the network. In Chapter 3 concrete pub/sub overlays with filter and rendezvous-based routing machanisms are presented.

As it is surveyed in [BQV05], rendezvous-based pub/sub routing convinces with controlled subscriptions distribution and matching, as well as better load balancing than filter-based routing. Each event of the possible event space is related to one or more brokers. These brokers are responsible for a partition of the event space by storing subscriptions and matching events. Rendezvous routing can be based on distributed hash tables (DHTs). These DHTs form a decentralized and distributed lookup service, containing pairs of related broker event space elements. Content based hash keys match to the related publisher and subscriber. The brokers responsible for the subscriptions and publications are also called home nodes. A very simple pub/sub system for this type of routing is the single broker overlay. These pub/sub overlays are very robust to handle inherently dynamic changes in the overlay, which almost always occur in MANETs. However, rendezvous-based routing is restricted on the respective subscription language. To represent large and complex multidimensional pub/sub systems the used subscription language must support this type of structure.

Finally, the subscription model of a pub/sub system determines how information about a subscription and publication is structured. The following subscription models are used in pub/sub systems.

Topic-Based/Channel-Based Subscription This type of subscriptions can be addressed with events, which do produce publications to a well known topic. Subscribers subscribe to topics/channels directly so that every recipient of a notification is known before the notification is generated. The topics are typically expressed as strings. Channels are also represented as strings where the matching of prefixes do resemble the affiliation of a subscription to a channel. As it is presented in Chapter 6, this subscription model is used in PeerfactSim.KOM and therefore is necessary in the implementation of the transition-enabled pub/sub overlay.

Content-Based Subscription A content-based subscription model matches subscriptions against the content of the published messages. In this case, subscribers describe their interest as an expression in a filter (e.g. regular expressions can be used). Unlike with topic-based subscriptions, recipients of notifications can not be determined before a publication is produced, however this subscription model offers more flexibility and expressiveness due to its more complex design. Instead of regular expression based subscriptions direct executable code can be transmitted. That predicate object is able to filter events at run-time, however, Java isn't a functional programming language and therefore this possibility will be not used by the transition-enabled pub/sub prototype. As presented in [FJL+01], very efficient and reliable implementations of filter algorithms exist, to build very fast pub/sub systems.

Type-Based/Subject-Based Subscription Subject-based subscriptions are a special case of content-based subscriptions. Publication messages carry subjects through the pub/sub overlay. Subscriptions are matched against the subject. Type-based subscriptions however matches event-types against filters. With these approaches, a closer integration of the language and the middleware can be achieved (e.g. Java generics can be used for subscription matching).

In summary, it is to say that at first glance, the pub/sub communication paradigm might look very simple. However, clever and powerful mechanisms behind it are useful in many situations. The most notable one is the separation between publisher and subscriber. Also various forms of different subscription models allow to customize the pub/sub system regarding the requirements of the service.

2.3 Alternative Communication Paradigms

This section briefly presents alternative communication paradigms in contrast to the previously presented pub/sub scheme. It is shown that many principles of these paradigms were later adapted in the pub/sub scheme and therefore takes on its benefits.

RPC/RMI Remote Procedure Call (RPC), by [BN84, TA90], is a method to execute functions provided on a distant device in order to reduce load on the local device or pulling data from it, and therefore provides a form of distributed interaction between two peers. RPC functions like a local method call executed on the local machine, but sending an request to the distant partner. The control flow is blocked until an answer with the requested result is received and the local process can continue its operation. This type of protocol leads to many problems when messages are lost or duplicated while traveling through the network. To reach a specific goal of error semantics, an extensive amount of effort must be taken. For example, an execute-exactly-once error semantic can hardly be reached through backups and distributed systems. Therefore, RPCs cannot be made entirely transparent on the calling side. Remote Method Invocation (RMI) is an object oriented perspective of RPCs and follows a very similar approach. Since the transition-enabled pub/sub system will be developed and implemented in Java, it is worthwhile to look specific a the Java RMI [Mic00]. Java uses its RMI middleware to access distant Java objects. The developer does not have to manually serialize and deserialize its data structure in order to exchange data. The communication is automatically established and ensured by the Java Virtual Machine (VM).

Asynchronous RPC/RMI An approach to achieve full synchronization decoupling in RPC and RMI, and consequently remove the bocking of the calling process while waiting for a reply of a procedure call. Usually it is done by splitting a procedure call into two separate calls. The first procedure call initializes the calculation of the result and store the result for the caller, ready to be pulled by a second procedure call. Since two RPCs are more vulnerable for communication problems, this approach isn't very reliable. The direct end-to-end semantic between caller and producer is still maintained with this approach. For pub/sub systems, this type of request/pull communication scheme is further advanced by not waiting for the caller to pull its result from the producer, but pushing the result from the producer to the caller as soon as it is generated. Pub/sub systems can also handle more than one subscriber for data.

Shared Spaces Distributed shared memory (DSM), as presented in [LH89, TSF90], is a form of shared information spaces. Analog to the organizational form of channel-based subscriptions, data is stored in a shared memory. By this, it is possible to enforce communication and synchronization between peers of the DSM. Typically, it is accomplished by a collection of ordered tuples. Communication is possible by placing and retrieving these tuples in the memory of a host. A common problem is the synchronization of simultaneously executed writing operations on the data. Mechanisms must ensure that a simultaneous access is prevented. In Java for example, the keyword `synchronize` is a realization of such a mechanism. Like pub/sub systems, DSMs provide a space and time decoupling by introducing a third role, the shared memory host, but they do not provide synchronization decoupling because consumers of information pull the datagrams from the host in a synchronous way. This also limits the scalability of this communication approach. However, unlike RPC it is possible that multiple peers have access to a specific information.

Message Queuing Similar to the pub/sub communication approach, message queuing [BHL95] uses a logical container on a host to store messages between peers separately. A queue is generally

treated as a global information space. These spaces are filled with messages from publishers of an information and do provide additional features like the ordering of messages that DSMs do not provide. But eager to retrieve information as soon as possible, messages are constantly pulled from the queue by the consumers. Other than in a pub/sub system, synchronization decoupling is only given on the producer side, although the end-to-end semantic is decoupled like in DSMs.

Message Passing The last for completeness presented common communication paradigm is message passing. It is the easiest way to establish communication between two entities someone can come with. Messages with relevant information are sent directly via a end-to-end path between producer and receiver. The effectiveness of this approach is largely determined by the used routing algorithm and therefore is typically not used to develop a distributed application. Message passing only provides synchronization decoupling on the producer side.

Table 2.1 conclusively presents the categorization of space, time, and synchronization decoupling of all presented communication paradigms. As seen, only pub/sub systems provide all three types on producer and receiver side. However, conceptional, pub/sub systems are constructed using mechanisms and approaches of all other communication schemes.

Communication Scheme	Space Decoupling	Time Decoupling	Synchronization Decoupling
RPC/RMI	−	−	○
Asynchronous-RPC/RMI	−	−	+
Shared Spaces	+	+	○
Message Passing	−	−	○
Message Queuing	+	+	○
Publish/Subscribe	+	+	+

Table 2.1.: Compared decoupling abilities of different communication mechanisms. (+) yes, (−) no, (○) only on producer side (adapted from [EFGK03]).

2.4 Complex Adaptive Systems

After showing the principles of pub/sub systems and similar communication paradigms, a short overview of complex adaptive systems is given. A transition-enabled pub/sub system has to adapt different pub/sub overlays into one greater and powerful service. Mechanisms and protocols used in other adaptive systems can be helpful to accomplish this requirement. Defined by [AEH05], a complex adaptive system consists of inhomogeneous, interacting adaptive agents with the capability of learning and self-organization, to form to top-level system with much more powerful services. This is particularly useful for systems on top of MANETs. Since local disturbances can always occur in a mobile environment, a complex adaptive system can react to those without the presence of a mighty super entity. Another property of a complex adaptive system is the possibility to express the system mathematically, which is particularly helpful to prove that a system does what it is designed for.

A typical form of a self-organizing distributed system are sensor networks. Small devices monitor their environment and provide the collected information for the used application. Pub/sub systems are ideal overlays for such a type of network. The publisher are lightweight sensors which do publish information once it is collected. A subscription of a subscriber might subscribe only to a specific topic, for example

temperature. Every time the temperature in once place changes or overgrows a specific threshold, a publication is generated and the subscriber is notified. Due to its self-organizing form, the pub/sub system can easily accomplish the loss of a sensor node.

An other form of adaptivity are delay tolerant networks. Delay tolerant networks are used in situations, where sender and receiver are that far apart, that the delay between the generation of a message and its recipient cannot be caught by usually used protocols, like the Transmission Control Protocol (TCP). Due to its time, space, and synchronization decoupling nature of pub/sub systems, messages can be published and the receiver can be notified without establishing a direct end-to-end connection. In the terminology of delay tolerant networks, a message is called bundle [SB07]. These bundles are stored, carried and forwarded by intermediate broker nodes and are eventually delivered to its recipient, even if there is no direct end-to-end route available form the publisher to the subscriber at any time.

It has been shown that the theme of a transition-enabled pub/sub service falls into many different areas. The pub/sub communication paradigm was introduced and the characteristics of MANETs and complex adaptive systems were shown. In the following chapter, different realizations of pub/sub services and middleware architecture building upon these basic schemes are presented.

3 Related Work

This chapter gives an extensive overview on existing pub/sub mechanisms with a focus on systems designed for the use in MANETs. Also investigated are pub/sub systems, which are based on fixed networks. Their mechanisms may still be interesting for usage in a transition-enabled pub/sub overlay. For the implementation of a transition-enabled pub/sub overlay, a look at adaptive middleware and its possibilities is also taken. In the last step, all discussed systems are compared in terms of structure, underlying topology, subscription model, and quality of service metrics. Their applicability for a MANET environment and within a concrete transition system are the focus of this to study.

3.1 Publish/Subscribe Systems on Fixed Networks

Pub/sub systems on fixed networks are remarkable on their great performance and broker structure. These systems often to not make any requirements on the used topology, since they follow other goals. Here, the organisation and forwarding of information are the centre of attention. Since the goal of the developed transition-enabled pub/sub overlay is to be lightweight and easy to manage, these type of systems give an advice to accomplish this purpose.

3.1.1 Conventional Publish/Subscribe Systems

In this section, pub/sub overlays are presented and discussed which do not rely on a concrete structure. Those systems can easily be deployed and used without much preparations on the client side. This comes in handy, if the devices on which such a system may run are small, without much storage or computational power.

Siena – Scalable Publish/Subscribe System

The first example for a pub/sub overlay mechanism is SIENA [CRW00]. It is designed for an Internet-scale event notification service, so its purpose is not mainly for small or medium sized MANETs, but also for WANs. However, SIENAs scalability for large networks helps to easily adapt the system to different kinds of underlying broadband networks.

SIENA is based on an distributed event notification service that delivers those notification event to clients via access points. A client subscribes to an object of interest, so that a logical link will be established between these two nodes. Clients of the system use access points of their local servers to publish the information about a notification that they generate and to receive the notifications. A client can also act as publisher and subscriber at the same time. Brokers are located in between the clients, which carry all messages through the network and store the information about a subscription in a predefined data model which drives the semantics of the service. A correlation between nodes, subscriptions and publications in SIENA is represented mathematically. The expression $(f \subset_S^N n)$ is defined as a notification n, matching a filter f. The following two examples illustrates the matching of publicized notifications and subscriptions. On the left side of the covering expression the subscription of an interested party is shown. On the right side an incoming notification is defined.

Subscription		Notification
string typeOf = temperature	$\subset_S^N$	*string* typeOf = temperature *int* temp = 21
string typeOf = temperature *int* temp < 28	$\not\subset_S^N$	*string* typeOf = temperature *int* temp = 32

The first notification covers the first subscription, while the second notification does not cover the second subscription, since the defined temperature did not fit the requirement. Many other pub/sub systems use this form of description to build an own notification service. SIENA uses this expression also to represent advertisements. An advertisement can be understand as a subscription, but only on publisher side. Advertisements ($a \subset_A^N n$) defining a set of notifications that are potentially generated by an object of interest, while a subscription defines a set of interesting publications for an interested subscriber. Advertisements also allow the service to decide if a subscription must be sent to an object of interest or not, so advertisements form the path for subscriptions.

To route a message about publication events through the network, a routing path must be established. One mechanism discussed by the developers of SIENA is to broadcast notifications through the whole network, forming a type of flooding based pub/sub service. Subscriptions are maintained at their access point and as soon as a notification matches a subscription, the subscriber is notified locally. As the expected number of notifications can be very high, this strategy is no longer used in SIENA. The strategy chosen by the developers is similar to a IP multicast routing protocol where notifications are sent only to event brokers that serve clients which are interested in the respective notification. It is a multi-hop strategy with additions to find the shortest path. To further optimize the notification delivery for an event, filters are used to reduce the workload on the network itself, the storage requirements for servers, and the number of subscriptions held at each server. This might be critical if the system is used on networks with low throughput and high traffic on the system. The filers are implemented using patterns so that similar events with equal meanings can be matched into one event for all subscribers of that event.

Due to the fact that SIENA focus is set on WANs and the Internet, it ignores all additional environment information which can be available on MANETs. For an transition-enabled pub/sub overlay on top of MANETs, the principle of subscription and advertisement matching can be used. It represents the publish subscribe model in its very simplest way.

Gryphon – Content-Based Publish/Subscribe System

GRYPHON [BCM+99] is an efficient multicast protocol for content-based pub/sub systems. In distinction from subject-based models, where each unit of information is affixed to a set of subscribers, content-based models handle published data in an information space. Based on the mechanisms of GRYPHON, the operation and specific properties of information spaces and functions, like filtering, are described in detail.

As briefly mentioned in Chapter 2.2, information spaces are logical associations which are accurately described and identified in GRYPHON with a set of primitives, e.g. strings, pairs, integers, etc., but more complex and richer data structures like Extensible Markup Language (XML) can be used as well. Instead of subscribing to a single subject of interest, clients can subscribe to a full range of publications by choosing filtering criteria along the information spaces. An example used in GRYPHON of a information space is the tuple [issue: *string*, price: *float*, volume: *integer*]. This describes a stock trade with stocks, the price and the volume as data structure. A subscription using the filter predicates [issue = “IBM” & price < 120 & volume > 1000] covers all trade events matching the given conditions.

GRYPHON now efficiently implements a content-based pub/sub system by handling two key problems that must be solved before such a system can be deployed. The first problem concerns the matching of events against a large number of subscriptions. A matching algorithm is the basis of the distributed multicasting protocol. To match and sort search requests and for an efficient management of information spaces a parallel search tree is used as data structure. In Figure 3.1 one possible four step matching tree is shown. The steps a1 – a4 are predicates with *int*-values in a subscription. If a subscription does not contain a predicate, the path is labeled with a (*). Filled nodes of the tree contain interested nodes while empty nodes correspond to subscriptions, where currently no subscriber is active.

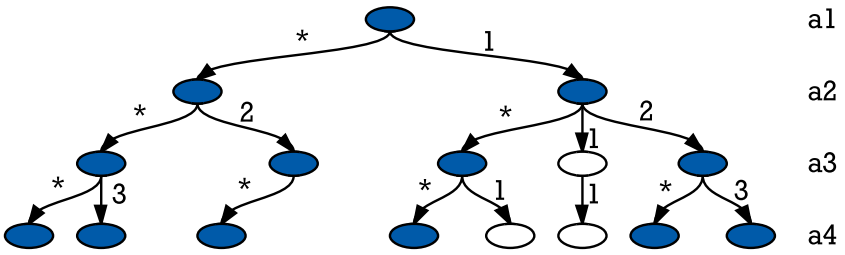


Figure 3.1.: Construction of GRYPHONS matching tree (adapted from [BCM+99]).

Each subscription corresponds to a path from the root to the leaf of the search tree. To return to the stock example given above, subscriptions concerning prices smaller than 120 would be saved on the left path of the tree and all events handling prices greater than 120 would be saved on a right path of the search tree. Each step going deeper in the search tree corresponds to a test of the value of another attribute of the information space. Matching of an event is performed by following all those paths from the root to the leaves that are satisfied by the event. As the number of subscriptions increases, using a tree structure increases the cost of matching linearly, yielding to a scalable algorithm for large number of parallel subscriptions. GRYPHON also shows some optimization opportunities for the search tree in their work by avoiding search steps removing certain attributes which are not often tested. A separate sub-tree is built for each possible value for these attributes, which does further decrease the matching time. Each node or a set of nodes of the search tree can be distributed to a broker in the underlying network. These brokers now act as event brokers which handle all subscriptions concerning the given information space.

The second problem concerns the efficient multicasting of events within a network of message brokers. This is done by an extended matching algorithm for a network of brokers, publishers, and subscribers. Each time a node receives an event, just enough matching steps are executed to determine which of the nodes neighbours should receive the message. The search area is much smaller because the number of neighbours of a node is much less than the total number of subscribers in the system. A system like GRYPHON, which takes the time and effort to build a tree structure to model the pub/sub data structure can be highly efficient, especially if the pub/sub system is geographically distributed and message brokers are connected via a relatively low speed connection for a large number of publisher, subscriber, and events.

3.1.2 Peer-to-Peer-Based Publish/Subscribe Systems

According to [Sch01] Peer-to-Peer (P2P) systems distribute tasks of an application among multiple nodes of the networks, with the goal of combining the capabilities of all participating peers into a more powerful system. This is done without the need for centralized coordination by servers. P2P systems provide an useful infrastructure for pub/sub overlays on top of the peer network. Applications like SKYPE[1] use an Internet wide P2P community, but also file-sharing and streaming applications exist as well. Several pub/sub overlays use these extended capabilities, as it is presented in this section.

[1] `http://skype.com` [Accessed 28/11/13]

Scribe – Large-Scale Publish/Subscribe Infrastructure

A similar approach to SIENA for an Internet scale pub/sub system is SCRIBE [RKCD01]. It is designed to scale to large numbers of subscribers and topics as well as support of multiple publishers per topic.

SCRIBE is build on top of a P2P overlay called PASTRY [RD01], so that it can benefit from its provided advantages like scalability, routing and fault-tolerance. PASTRY provides Application Programming Interface (API) services for routing, sending, delivering, and forwarding of messages. Any node which runs PASTRY as overlay has an unique and uniformly distributed node identifier (ID). The `route(msg, key)` operation commands the PASTRY underlay to route a given message to the node with the ID numerically closest to a given key. `send(msg, ip)` sends the given message directly to the node with the given IP address. The addressee implements a `deliver(msg, key)`-function, which is called by the PASTRY underlay if a message for the node is received. With the implementation of `forward(msg, key, nextId)`, a node is informed, just before the node forwards a message to a node with the next ID. By that, the SCRIBE-system is able to reroute or terminate a message and therefore directly influences the routing algorithm of PASTRY.

Node failures are handled by sending a message to an adjacent node which is the numerically closest node ID or its neighbour. With this mechanism, a distance metric between any pair of nodes can be derived. The authors limit the discussion on locality properties of PASTRY which could be used by SCRIBE to achieve a better performance. One property, the total number of hops that messages move can be minimized using the address space of PASTRY. In each routing step a message is routed to the nearest node with a longer prefix match. A topic, which may be created by any SCRIBE node, resembles an issue on which another node can subscribe to and build up a group of interest. Any node with the appropriate credentials for the topic can then publish events, and SCRIBE disseminates these events to all the topics subscribers. Two messages, which travel until they reach a rendezvous-point where their routes merge, form a root of a multicast tree. This information is used by SCRIBE to build a logical multicast tree on top of the P2P layer for the specific topic. Figure 3.2 shows an example of a multicast subscription tree created by SCRIBE.

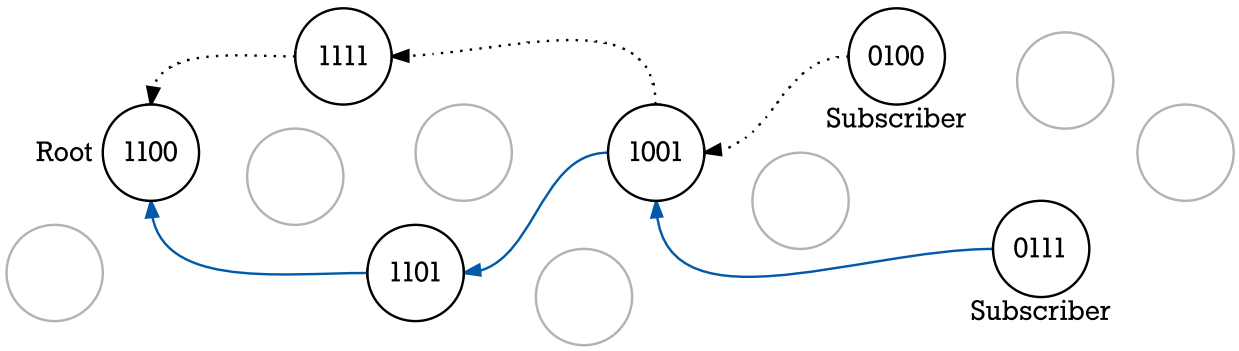


Figure 3.2.: SCRIBE mechanism for subscription and multicast tree creation (adapted from [RKCD01]).

The tree shows some nodes with a four bit ID. A topic is created with the ID `1100` and is located at the root rendezvous-node with the same ID as the topic. The colored lines indicates a three hop route from the subscriber with the ID `0111` to the root. The nodes along this path become forwarders for the subscription message. Subscriber `0111` is also added as child of the route for the topic. A second subscription message subscribing to the same topic caused by subscriber with the ID `0100` would take the doted indicated path. However, the subscription message can be terminated by the node `1001`, because this node is already forwarder for this topic. Further forwarding of that subscription message is not needed. At the final state, node `1001` manages node `0100` and `0111` as children. The tree is formed by joining the PASTRY routes from each subscriber to the rendezvous-point, again achieving better performance for the SCRIBE topic management and to support for a large set of subscribers. This example

also shows also nicely the route generation of PASTRY by matching prefixed and calculating distances based on the ID.

In conclusions, SCRIBE and PASTRY forming a decentralized, scalable, and reliable two-layer infrastructure for Internet scale pub/sub applications. The fault-tolerance of SCRIBE is based of the self-organizing properties of the P2P layer. SCRIBE efficiently supports a large number of nodes, topics, and a wide range of subscribers per topic. A two-step approach is something to keep in mind for a transition-enabled pub/sub service, where the concrete pub/sub overlay is isolated from the actual transition system. The management of subscriptions and the generation of routes based on IDs however, might not be satisfactory for a MANET environment. IDs of nodes would have to be generated based on the position of the nodes to avoid path that pierce the MANET multiple times. As it is also concluded in [TA04], "when the number of nodes in the network is much greater than the domain of attribute values, this could lead to have k useless nodes ... between two consecutive values in the range". Since the position of nodes is constantly changing in a mobile environment, fixed IDs are not reliable to use as addressing and routing scheme without increasing the overhead of the pub/sub system.

3.2 Publish/Subscribe with Support for Peer Mobility

In this section, pub/sub systems are investigated which are specialized for the deployment in a mobile environment. The presented pub/sub overlays are distinguished in structured and unstructured overlays, but firstly the characteristics of pub/sub overlays on top of MANETs are discussed.

[BCG04] explains how a mobile friendly pub/sub system can be built and evaluated, so that it can perform nicely on daily situations which can occur on a mobile environment. A proposed model consists of the communication medium, the network topology and the processes involved in the pub/sub interaction, built up to a layer structure. First, the example algorithm performs a neighbourhood discovery. This is done by periodically sending beacon signals with information about subscriptions on that node and its current velocity in the MANET. With the beacon messages, nodes can calculate a dynamic one-hop neighbourhood table. Especially neighbours with matching subscriptions to its own subscriptions are interesting to be maintained. With the movement speed information the node can adjust the frequency of sending such beacon messages. The next step of the algorithm uses the gathered neighbourhood information to disseminate events. Nodes send events to their neighbours if they have subscribed to the publication the event came from. The receiving node can now decide if an event notification is again forwarded. Events are stored on every receiving node for the case a new neighbour shows interests on it. The last step of the algorithm collects events according to their validity and the number of propagation of an event by the process. This can be quite different for every event, for example, a voice chat message is outdated after a few seconds while simple sensor information, like the temperature of a room sensor can be held for longer time. According to these findings an efficient and reliable pub/sub overlay on top of a MANET can be developed.

3.2.1 Structured Publish/Subscribe Overlays

Structured pub/sub overlays are placed between the transportation layer and the notification-service of a pub/sub network. These systems are specifically marked with a high fault-tolerance and therefore are interesting for pub/sub overlays on MANETs.

Hermes – Event-Based Middleware

HERMES [PB02] is a distributed event-based middleware architecture. The main feature of the HERMES infrastructure is the distinction between event clients and event brokers. An event client is an endsystem which publishes or subscribes to a specific event and it uses the services provided by the event brokers to communicate using notifications. So, event brokers which are located between two or more event clients

handle the entire functionality of the middleware, resulting in a very light-weight implementation for the event clients. Brokers mange routing of the actual messages, handle, and accumulate the active subscriptions, filter subscriptions, and multicasts for greater scalability. For that purpose an event dissemination tree is dynamically constructed, that manages rendezvous-nodes for advertisements and subscriptions in the network, very similar to the already mentioned SCRIBE system. The tree in the HERMES system is updated every time a new publisher joins the publish subscribe system by sending an advertisement message. After including the event publisher into the tree, publications can be sent by the new node.

In HERMES, the event brokers form the nodes of a P2P overlay routing network that is similar to PASTRY which is used by SCRIBE. For each subscription, a rendezvous-node exists in the network. To find this node, a hash function over the name of an event is used to calculate the corresponding overlay ID. That has the advantage, that no global knowledge is required in an event client or broker. The hierarchical architecture of HERMES allows to separate the network layer, the routing overlay, the filtering, and event distribution from each other.

The work of [HGM03] describes the basic construction of a pub/sub tree which is used in HERMES or other pub/sub middleware on a mobile environment. It surveys several tree construction algorithms which are running as a protocol on the nodes of a mobile pub/sub system. They attempt to produce an optimal search tree, while on the same time, minimize the cost of building and maintaining such a tree. The first algorithm they consider is a centralized algorithm running on a single node with global knowledge of the network state. The node collects all information needed, including the entire connectivity graph and user subscription information. This enables the computation of a tree which is optimal before it is distributed to all other nodes. The disadvantage of that approach is that this algorithm is very inefficient, which makes it unsuitable to a MANET. However, to improve the tree construction, the authors make some changes to the centralized approach so that it runs on each node of the network simultaneously. This removes also the need of global knowledge of every node, only information about itself and its neighbours are necessary.

In MANETs, clients often join and leave the network or disconnect because of the movement of the peers. In order to keep the connectivity alive, repair functions must continuously run on the overlying tree structure. [MCP08] introduces a self-repairing tree topology in mobile ad hoc environments. Most tree generating algorithms assume that the network topology does not change. The work presents a self-repairing algorithm working in the dynamic scenario of mobile networks by using concepts of AODV. The simplest way to build a pub/sub tree is a path finding algorithm which communicates between the event brokers. Route request and route reply messages are exchanged between adjacent nodes. The concrete information about nodes are saved on every node in a routing table. The goal is to minimize the amount of route messages caused by a broken link. To achieve that, nodes are grouped with a node as group leader and a group identifier to address the corresponding group sub-tree with a multicast address. Each node stores an ancestor list containing the identifiers of all its tree ancestors. If a node replies to a route finding request, it includes his ancestor list to the message. On the way back, the traversed nodes append their identifier to this list. The list now contains all information to build a knowledge base of node distribution on that node the request was sent to. On a link failure, the node can now easily determine which and how many nodes are affected by the break and repair the tree without sending a route request to all nodes. Another function solves long and cost ineffective tree rebuilds if a route reply can no longer reach the requestor. In AODV, if a node cannot forward a message to its destination anymore it produces a new route request by itself. To prevent an uncontrolled flooding of route requests, nodes only forward route requests as long as the request travels along the links of the sub-tree rooted at the initiator. The route request is no longer forwarded, if the message leaves the tree. This prevents loops and the chances that the disconnected node reconnects increases. The same principle can be applied to route replies. If a group leader leaves the network, or all links to a leader are broken, a new leader will be elected. This can be fast detected using timeouts. In the proposed system, a group leader is very important for maintaining the tree structure, because it also handles split tree repairs. The amount of

messages needed for merging partitions of the tree is heavily reduced, if that process is passed to the individual group leaders.

On this stage it can be seen, that tree structures in pub/sub overlays are very efficient for building and maintaining a structure for pub/sub overlays on top of MANETs. Peer mobility is well supported.

Density Driven Publish/Subscribe Service

A density driven pub/sub virtual topography is shown in [FK12]. It relies on a main parameter of a MANET, the number of clients in a given area. The algorithm is based on a density driven virtual topography which is a substrate to route the messages through the network. This is done by assigning every node in the network with a virtual height where the height represents the number of nodes in the neighbourhood. Nodes in higher dense areas will have a greater height, than nodes located in lesser dense areas. Nodes which are higher than all neighbours of that node are called hilltop. They are located in the middle of all local dense areas. Figure 3.3 illustrates a translation between the topology of the MANET (a) and the virtual density driven topography (b).

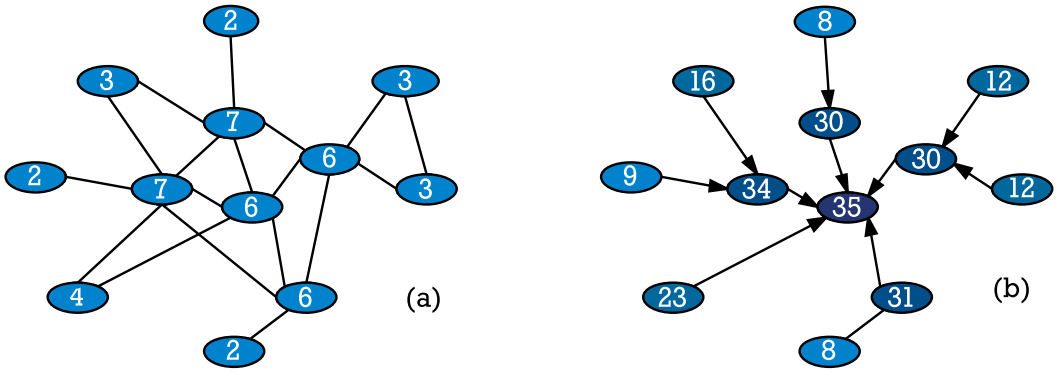


Figure 3.3.: A tree topography, constructed using the density of the network topology (adapted from [FK12]).

The numbers on each peer represents the density of a peer, respectively the amount of peers a node is responsible for. In this case its not only a one hop local density, but a multi-hop cluster density value. This value can also be used to derive Network Layer Reachability Information (NLRI). This information displays the range, starting from a node to an other node, that are reachable with an end-to-end path. A higher multi-hop density will result in better NLRI for a node, which is essential to qualify it for a broker node. The topography represents a tree, where every route by choosing the highest direct neighbour is directed to a local maximum of density. Summarizing, each node with at least one child is a broker for each node, that is connected to this node and has a lower density. The virtual topography itself is maintained by exchanging hello messages periodically. If no message is received from a specific node for a predefined period of time, it is assumed that the corresponding node is no longer in transmission-range. Every node calculates its height and personal hilltop based on these messages and the structure of the overlay is constantly updated.

With such a topology in mind, a density driven pub/sub protocol can now easily be deployed on the network. The protocols assumption is, that nodes in a dense area are mainly interested on the same topics. To illustrate this fact, the following example scenario is considered. People in a football stadium are probably more interested in an action replay of a goal than people outside of the stadium, therefore it makes much more sense to place the broker of a publishable event in the middle of it, assuming all nodes have the same capacity. Two approaches, a low cost and a low delay density driven pub/sub approach are briefly described in the proposed work.

To minimize the amount of messages sent, the low cost approach spreads publications of a node over all hilltops of the density tree. To search for matching publications of a nodes subscription set, each node periodically starts a lookup, using a walk over all hilltops of the virtual topography. On each

step, all matching publications for the node are accumulated and all information about the publication is returned to the node. The time it takes to find a matching event is limited by the time between the lookups of a subscriber. The low delay approach decreases the delay between an advertisement and a successful delivered event by saving the subscription information of a node on all hilltops of the network. Immediately after a publication event the publisher starts a lookup walk over the virtual topography. Each hilltop the publication arrives, it checks for matching subscriptions and in case of a positive match the event is forwarded directly to its corresponding subscriber. In total, the amount of messages on the second approach are increased but it provides a much faster delivery than the low cost approach. In both cases, the algorithm uses the structure of the virtual topography to efficiently navigate from hilltop to hilltop across the network. Every node which receives a message has to calculate the next hop and decides if a message is forwarded. If a node is a hilltop, the message needs to be directed to the opposite direction from where it was received in order to reach a new hilltop. In case a node is not a hilltop, it forwards the message to the next hilltop the message didn't came from using the next higher neighbour.

According to the proposed work, another advantage of that density driven pub/sub algorithm is that messages are distributed using point-to-point messages rather than broadcasts. While point-to-point messages are much more reliable than broadcasts, the structure of the virtual topography ensures that objects of interests are very efficiently disseminated in a local dense area using hilltops as central stations. Conclusively, a density driven pub/sub overlay is an excellent approach for a pub/sub system in a MANET environment.

3.2.2 Structureless Publish/Subscribe Overlays

Structureless pub/sub overlays are characterized by their simplicity, which makes them not less useful under certain circumstances. In fact, a transition-based pub/sub system will contain both, structured and unstructured pub/sub overlay approaches to form a more mighty overlay.

Flooding – Multicast Approach in Mobile Ad Hoc Networks

A different approach described in [HOTV99] to reach a large number of clients in MANETs is flooding. It uses the already existing IP multicast protocol of the network to reach each peer of the network. The main goal of this concept is to stay connected anywhere at anytime and still maintain a reliable and high performance network connectivity. The right ratio between robustness and efficiency must be found in such a system. The proposed work even goes further by recommending this type of communication in environments where clients have limited capabilities, for example small memory of computing power of mobile devices. With a simple flooding mechanism, clients only have to forward messages, regardless of their role in the application scenario. No status information must be maintained or distributed to build the system structure. The results of the studies and simulations have shown, that this kind of approach is very reliable in networks with fast moving clients, for example in car-to-car communication, where there is essentially no limit on client velocity and movement direction.

As seen in [PDJ03], this concept can even be further optimized. An optimized flooding protocol is developed to explicit address the effect of mobility. The optimized flooding algorithm extends the simple flooding protocol as follows. Every broadcast packet obtains additional fields, which contain location information. Whenever a node transmits a broadcast packet, the fields are set to the location of the node from where it received the message, as well as its own location. The simple flooding protocol only discards messages if the node has transmitted the packet earlier based on a message ID. With the location information, the node has more options to discard a message if other nodes cover a specific location. It can also delay message forwarding based on distance calculations and subsequently reduce the amount of message overhead in the broadcast system.

For a transition-enabled pub/sub system it could be interesting to study the possibility to use flooding based pub/sub overlays only for node clusters with a high internal velocity. This self-contained clus-

ter now can be managed from another pub/sub overlay as single node. Adaptive mechanisms of the transition-enabled system provide the required customization. The node cluster is addressed using a multicast address or a direct cluster leader broker node, as it is shown in HERMES.

Steam – Publish/Subscribe Middleware for Mobile Ad Hoc Networks

STEAM [MC02] has been designed for mobile environments that include a large number of entities representing real world objects using wireless technology and the ad hoc network model.

All clients of the STEAM system interact using event-based communication in order to route event subscriptions to them. As proposed, event-based communication accomplish the requirements of a mobile computing environment, compared with a traditional client/server communication model. The reason for that is, that in mobile technology a centralized middleware component does not cover all entities, due to simple range and geographical restrictions.

STEAM implements an implicit event model that allows consuming clients to subscribe to distinct event channels. In addition, it extensively uses different types of event filters to overcome the dynamic aspect of the network environment.

Another characteristic of STEAM is the introduction of proximity-based group communication. Proximity groups are logical sets of network components based on their geographical orientation. The nodes discover each other using periodically sent beacon messages. These groups allow to react to geographical and functional aspects of the application. This limits within information is propagated in the network. If real end-to-end semantic between clients in different groups is needed, STEAM falls back to a message delivery semantic to deliver events. With the addition to define delivery deadlines assigned to a specific event, a dispatcher determines the time to deliver the corresponding events to the subscribing consumers. Implicitly, afflicted with this communication mechanic, STEAM addresses many aspects of MANET environments. An expansion stage of this configuration is a density controlled MANET environment, where the group comprises a zone with more network nodes than in other areas of the network. As the pure flooding based approach, STEAM is recommended to use in a fast moving environment, like car-to-car communication.

Decentralized Publish/Subscribe Scheme

[ADGS02] presents requirements and interfaces of a basic pub/sub scheme which preserves the anonymity of the publishers. For that, an algorithm is introduced, which preserves the orientation of a logical link between two entities in the system and the reorientation of a link. Subscriptions are modeled using links in the communication graph representing the pub/sub system. But first the neighbourhood maintenance in mobile networks is defined. An entity joins the network by choosing a connection point where it is least likely that its failure disconnects the network. A neighbourhood list is maintained by processing one directional communication links to new joined entities. A set of rules guarantee that logical links always form an acyclic connection graph. By that, information is diffused in the network in an anonymous way. The link reorientation algorithm ensures that the network has at least one sink node at any time. Sink nodes are privileged brokers which send or forward any information regarding any event of the pub/sub system. All messages are routed to the next sink node. Other nodes only receive messages from these nodes.

What this pub/sub scheme does is not focusing on a performance metric of the overlay, but focusing on the requirements of safety and anonymity. It proves that those requirements also have an impact on the logical structure of an overlay, which may be in conflict with other schemes used in mobile networks. An interesting thought is that if safety and anonymity is only required in a specific condition or scenario, an transition system could switch to that scheme only during that time and later return to normal operation.

3.3 Adaptive Middleware Architectures

After the investigation of interesting pub/sub overlay for the transition-enabled pub/sub service, an overview of middleware architectures is given. Middleware with great adaptivity and utility for a transition-enabled pub/sub service is mainly focused. The shown concepts and requirements for an adaptive middleware will influence the design of the developed transition system.

Jedi – Object Oriented Publish/Subscribe Middleware

[CDF01] demonstrates an event-based distributed infrastructure implemented in Java, called JEDI. It supports the development and operation of event-based systems, using the Java object schema. This work is especially important because the developed transition-enabled pub/sub system is implemented on a Java based P2P simulator.

The architecture of JEDI is based on logical active components. They resemble an enclosed and autonomous working agent performing application-specific tasks. These components generate and receive events in the system. If required, they also communicate with each other using events. Beside the normal components which use the basic subscribe and publish operations, another type of entities are introduced, called reactive objects. These objects in the architecture are designed for components which execute a number of operations while receiving an event and again publish events in this process. An active object could be anything, from a state server over a software agent or simple tools for an affiliated service.

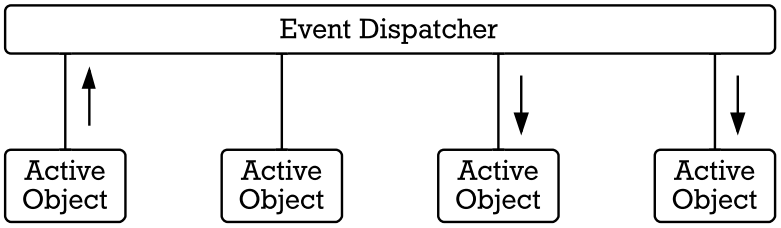


Figure 3.4.: Logical view on the JEDI architecture (adapted from [CDF01]).

As shown in Figure 3.4 an event dispatcher sits between the components and notifies each component that has explicitly declared its interest in receiving an event. The event dispatcher is the abstraction of the subscription model. JEDI provides a centralized and a distributed implementation of the event dispatcher. The centralized event dispatcher can handle a limited number of events running over a local area network while the distributed event dispatcher can also handle network-intensive applications. The distributed implementation is based on a set of dispatching servers interconnected in a tree structure. Different strategies can be used to distribute events across the hierarchy of dispatching servers.

The proposed service provides operation in WANs and MANETs, a lightweight software technology used to implement these service, and a plug-and-play approach to support dynamic reconfiguration and introduction of new service components. An object based structure like JEDI can be used to cover different approaches of pub/sub overlays by providing a common interface. Also interesting to see is that the model of pub/sub is used in a software architecture itself to build a reliable and efficient system.

Another complementary event-based architecture is YEAST [KR95]. The main advantage of YEAST with respect to JEDI is the specification of a mechanism to detect the combined occurrence of events, using a centralized server main component. With the use of event patterns, the production of published events can be reduced and therefore the efficiency of the system can be improved. For a transition-enabled pub/sub service, this concept could be adapted to provide further services for the used pub/sub overlays. Not alone the transition between different overlays or the simple routing of messages, but also straight intrusion to the used pub/sub model can be useful and should be considered.

Green – Configurable Publish Subscribe Middleware

GREEN [SB05] is a pub/sub middleware which supports a wide range of computing applications. The design of GREEN allows to exchange components of the system to actively adapt it to different network environments, for example WANs or MANETs, and different subscription models. These system components are modeled as plugins which can be exchanged and combined to build an appropriate pub/sub service.

In contrast to monolithic pub/sub middleware, GREEN provides a reconfigurable platform for applications and services on top of the pub/sub system. This is done by using the component technology of OPENCOM [CB04]. Components of the system interact and communicate with other components using well defined interfaces. When components are plugged together, they act as one system module which can be addressed in a single address-space. GREEN consists of two main component frameworks, an interaction framework is configured with interaction type plugins and an overlay framework which consists of overlay implementation plugins. The interaction framework provides the pub/sub interaction and the overlay framework provides the event broker overlay. The encapsulation of these component frameworks ensures that they can be configured, reconfigured and re-used independent of each other. An extensive example of an adaptive pub/sub overlay on a WAN environment is given and consists of the following components.

Publish/Subscribe Publish and subscribe components are representations of objects, that can be published or subscribed to. For example, it can be implemented as a topic or a channel to accomplish topic-based or channel-based subscription models.

Notifier The notifier component is an interface for a node that is called if a notification arrives, just like the PASTRY `deliver(...)` operation. It is also called the proximity-plugin, because it communicates with the application. As proposed, another version of the notifier is a script-based component, where subscribers specify composite events. These script-files can be loaded and unloaded dynamically.

Event Dispatcher Analogically to the architecture of JEDI, the event dispatcher component is responsible for the propagation of subscriptions and notifications.

SOAP Messaging The Simple Object Access Protocol (SOAP) messaging interface component defines the messages and message types of the service. The message format is based on a XML information set and is chosen because of its widespread use in open source projects.

Content Filter The content filter module provides filtering operations, such like advertisements on the publisher side. Another version of this module is a proximity based filter. The pub/sub notifier, event dispatcher, soap messaging, and the content filter form together the interaction component framework.

Overlay Component Framework As already mentioned, the overlay component framework provides the event broker layer and is therefore responsible for the transmission and routing of the SOAP messages. Possible versions of this framework are for example a content based routing overlay on top of a DHT overlay, a SCRIBE overlay or a IP multicast module.

This composition is a configuration to address the constraints of a WAN environment, with the use of a content-based interaction scheme. Additional plugins provide distinctive quality of service (QoS) and monitoring functions. The event broker overlay component consists of plugins which provide distributed event routing and filtering implementations for the selected interaction type. If the system is built for a MANET environment, a plugin for MANETs addresses the specific properties of mobile networks, such as the mobility model of the network. One implementation suggested for the plugin consists of a control component which builds and maintains the broker network topology, a forwarding component which routes events over the broker network and a state component which encapsulates key states such as the neighbour list and node state.

GREEN is a highly configurable pub/sub and event-based middleware with the ability to change components of the system at run-time. It serves as starting point for a transition-enabled pub/sub system,

where transitions are modeled and integrated by a GREEN inspired component structure. The abstraction of the different subscription and publication models are one of the most remarkable features of GREEN that are also interesting for a transition-enabled pub/sub overlay.

3.4 Comparison of Existing Publish/Subscribe Overlays

In the previous sections, existing pub/sub systems are presented. The following comparison will again give an overview of the different pub/sub designs. The contribution of these designs towards a transition-enabled pub/sub system are discussed. But firstly, the presented pub/sub systems are categorized, regarding the underlying topology, the used routing algorithm, and the subscription model. Also a deep look at the used topology of the different pub/sub overlays is given, as well as different QoS metrics are tracked. Table 3.1 presents a categorization of the analyzed pub/sub systems.

Overlay	Network Type	Pub/Sub-Model	Routing	Topology	throughput	latency	robustness	scalability
Siena [CRW00]	fixed	topic-based	IP multicast	structureless	○	○	+	+
Gryphon [BCM+99]	fixed	content-based	multicast	broker-tree	+	○	−	+
Scribe [RKCD01]	fixed	topic-based	rendezvous	structureless	−	−	−	+
Hermes [PB02]	fixed	topic-based	rendezvous	broker-network	+	+	○	+
Density [FK12]	wireless	content-based	rendezvous	density-tree	○	−	−	+
Flooding [HOTV99]	wireless	topic-based	multicast	structureless	−	○	+	−
Steam [MC02]	wireless	topic-based	proximity-based	muticast-groups	○	−	−	○
Decentral [ADGS02]	wireless	topic-based	multicast	structureless	○	○	+	−
Jedi [CDF01]	internal	channel-based	centralized	centralized	○	○	○	○
Green [SB05]	wireless	generic	generic	generic	○	○	○	○

Table 3.1.: Comparison of different pub/sub overlays and adaptive middleware. (+) optimized, (−) inefficient, (○) no prediction.

As it can be seen, flooding or multicast based approaches do show a good fault-tolerance and a low latency on small sized network or a fast moving mobile environment, however, their scalability is not given for a large number of nodes or high density areas with strong bandwidth requirements. Pub/sub systems, which take the effort to create internal structures like a density based broker tree are very efficient in terms of throughput, subscription, and publication handling. For very small networks, the complicated creating of an internal structure might not be worth the effort. For the transition-enabled pub/sub system can be noticed, that a multicast based approach is more usable if a defined threshold, based on the number of nodes or the latency requirements, is not exceeded. A switch to a more structured overlay however is mandatory on large scaled mobile networks.

One problem repeatedly occurred on structured overlays in a mobile environment. If nodes join and leave the network in a rapid way or change their position internally, the structured overlay may fall apart. These link beaks must be handled. Some systems provide active repair functions to accomplish the MANET environment, other systems just rely on fixed networks. The following scenario considers a networks, where a cluster of nodes have a high internal velocity. Other nodes outside the cluster are slow or even stationary. For the cluster with fast moving nodes a multicast based overlay would be appropriate, while on the outside a structured overlay with rendezvous-based routing would fit the requirements of scalability and efficiency.

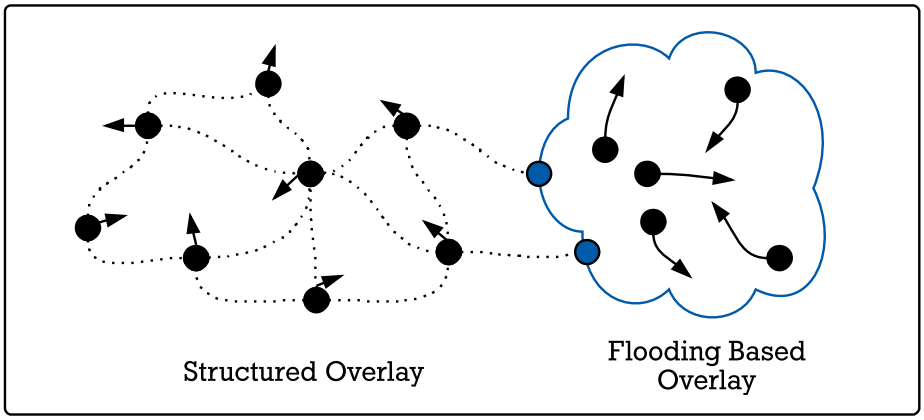


Figure 3.5.: Distinct separated pub/sub overlay cluster in a transition-based pub/sub environment.

Figure 3.5 shows a network with a node cluster of node with a high velocity and nodes outside of the cluster with slow movement-speed and stable conditions. The arrows at the nodes indicate the relative velocity and direction of the nodes. A transition-enabled pub/sub system could use different overlays based on the internal conditions of a cluster of nodes. Nodes on the rim of a node-cluster act as group-management nodes, receiving messages from other overlays and forward them internally. In the following chapter, requirements for a transition-enabled pub/sub service are presented. These requirements are based on the knowledge of the previous presented pub/sub systems and middleware.

4 Requirements for Publish/Subscribe and Transition Support

In this section, requirements for the design of a transition-enabled pub/sub overlay are derived from the related work. The developed concept and architecture of the transition system between different pub/sub overlays are based on the analysis and assumptions of existing pub/sub systems and middleware for MANETs shown in the previous chapter. The requirements need to be fulfilled by the design of the transition-enabled system presented in this thesis. The system will be tested and evaluated against these requirements with a prototypical implementation of the design. Particular attention is paid to the system behavior and adaptation options. But before specific requirements for the concrete transition service can be made, transitions on a pub/sub overlay level must be defined.

4.1 Transition Definition and General Requirements

The transition enabled pub/sub system supports transitions between different publish/subscribe overlays. For this reason it is very important to define transitions in context of a pub/sub system and in context of a network topography. Each pub/sub system discussed in the related work follows different goals and implements different mechanisms and protocol steps. Relations between these mechanisms must be identified and generalized to make the transition system adaptable in respect of the different mechanisms.

The following Figure 4.1 presents a scenario with a transition between a centralized pub/sub structure (left) and a group-based pub/sub structure (right) after the transition.

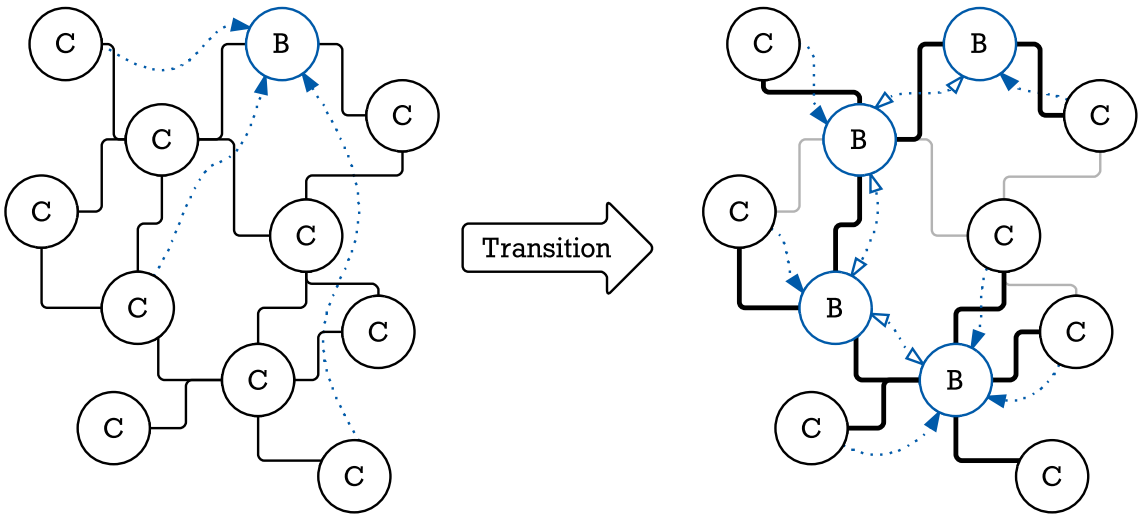


Figure 4.1.: Transition visualization on the logical overlay layer.

In both situations, the broker (B) and clients (C) without broker functionality are marked. If a node is in transmission range of another node, the connection is marked with a solid line. The membership of a client to a broker is marked by a dotted line. As shown in this example, a transition does not only affect the pub/sub overlay structure, but also the routing is changed. In the centralized example, clients have to forward messages from other clients towards the single broker. In the group-based overlay only

brokers have to maintain a routing protocol since each client is affiliated to a broker via a single hop connection.

The following general requirements are important to design a service with seamless transitions between pub/sub overlays.

Broker Migration Most pub/sub overlays use a form of broker with server capabilities to save subscriptions on specific nodes in the overlay. These nodes play an important role in the functionality and performance of the overlay. The choice of which node becomes a broker and which node only has client capabilities depends on the implementation of the switchable pub/sub overlay. While switching to another pub/sub overlay, these nodes might not be the same. The transition system must provide a mechanism that ensures a migration of these brokers during a transition.

Retainable Storage A pub/sub overlay relies on the fact that information in form of subscriptions and publications is stored in the system. During a transition, this information must be maintained. A central server approach does save this information on a node which does act as a single broker in the overlay. A distributed approach does spread this information on many brokers in the overlay. If a transition is executed between such different mechanisms, ongoing subscriptions and publications must be collected and stored on the single broker or distributed to many nodes in the overlay in order to maintain the time and synchronization decoupling even through multiple transitions.

Communications A transition has a high impact on used mechanisms and protocols in the system, since each pub/sub overlay follows a different approach. It is also hard to execute a transition between pub/sub overlays on every node in the system at the same time. Messages do travel with a finite velocity through the network and their processing does also require time. 100% synchronization between nodes cannot be achieved and the use of a clock synchronization algorithms is also not trivial. Due to this reason it is very likely that some nodes already switched to the new pub/sub overlay, while other nodes still use the old one. The transition system must provide mechanisms, that ongoing communication can be handled by every node, regardless of which overlay they are eventually using. It might also provide a mechanisms to translate messages from one pub/sub overlay into another overlay using generic abstraction approaches. For example, messages which do subscribe to a specific subject follow a generic pattern, even if they are generated by a different overlay.

Eventual Consistency In distributed computing, consistency over all elements of a compound cannot be entirely achieved, due to the restraints of synchronization. Especially in MANETs, different node clusters can be separated from each other. In terms of transitions between different pub/sub overlays, nodes have to incorporate status information in a finite amount of time if that is possible. For that, error correction mechanisms must ensure, that always the newest status is used to reach consistency with the rest of the system.

With the aid of the general requirements for transitions, concrete requirements for the transition-enabled pub/sub service can be defined. These are important for functionality and measurement of the performance of the transition system.

4.2 Functional Requirements of the Transition Service

The following functional requirements define the behaviour and features of the transition-enabled pub/sub service.

Transitions The service must support transitions between different pub/sub overlays. Transitions between these overlays are seamless form the application point of few, meaning that they do not influence or delete active publications or subscriptions previously issued by the application. The active overlay is selected depending on the state of the MANET, so that it is ensured that always the pub/sub overlay with the best assumed performance is chosen. The transition service also

ensures that transitions are executed in a controlled environment, meaning that variations or insecurities on the state of the network do not break the transition algorithm or cause rapid oscillating transitions between different pub/sub overlays.

Publish/Subscribe Interface The system behaves like a usual pub/sub overlay. It provides a generic pub/sub interface for generation of subscriptions, publications, and advertisements, and may provide additional services to manipulate the transition system directly. From the application point of view the transition service behaves transparent. No difference to a stand alone pub/sub system, beside the performance, should be observable.

Publish/Subscribe Overlays The system supports an arbitrary amount of pub/sub overlays, meaning it supports the different subscription-models, routing mechanisms and internal structures of a pub/sub overlay. It provides generic interfaces for different node-types and protocols used in these systems. Concrete mechanisms and protocols of the used pub/sub overlays are not influenced.

Monitoring The transition-enabled pub/sub service must observe the state of the network and the status of the pub/sub overlays. A monitoring component must observe and calculate the current state in order to decide if a transition to another pub/sub overlay is necessary. The service supports central, distributed, and external monitoring approaches. It also must provide the ability to define generic parameters that are monitored. These parameters may contain physical information about nodes, for example the velocity or remaining battery, but also logical system information, for example the publication rate. The decision to switch between different overlays can be made conditional, which is also defined by parameters.

Controlling The system supports central and distributed control approaches. The controlling mechanism has to manage all peers participating in the transition-enabled pub/sub overlay. It initiates the transition between the different pub/sub overlays and also supports different transition strategies.

4.3 Non-Functional and Performance Requirements of the Transition Service

The following non-functional and performance requirements serve as criteria to evaluate the system. They furthermore influence the system architecture and used protocols.

Correctness The transition-enabled pub/sub service delivers publications and subscriptions correctly regardless of the active overlay or system state. Due to the concrete routing or handling of publications, notifications may arrive at nodes multiple times or reach nodes that did not subscribe to a concrete object of interest. The transition service must be able to filter these notifications and no difference should be able to be monitored between the transition service and a stand alone pub/sub overlay. An at-most-once error semantic is provided.

Extensibility The transition-enabled pub/sub service must be arbitrarily expandable with different pub/sub overlays, storage-types, message types, protocols, monitoring, and control mechanisms in order to support an extensive variety of different pub/sub overlays.

Stability and Fault-Tolerance The transition-enabled pub/sub service maintains operability, regardless of the given network condition. Thereby it has to provide a default fallback pub/sub strategy with a robust routing mechanism. Subscriptions or publications may be lost due to node or route faults. Handling of those errors is within the responsibilities of the used pub/sub overlay and therefore is also depending on the scenario and used application.

Robustness against Churn Churn is the measured value of peers joining and leaving the overlay. While pub/sub fault-tolerance is depending on the used pub/sub overlays, the transition service itself must be robust against churn in the MANET environment. Within this environment, churn is mostly caused by movement of peers.

Scalability The transition system must provide the ability to adapt to different sized scenarios defined by the number of participating nodes, network, and traffic size. The system must also scale to

the publication and subscription rate of the applications. Scalability is also depending on available bandwidth or other network conditions. Thereby it depends on the used pub/sub overlays, weather the scalability requirement can be met.

Delivery Delay The system must response to notification producing events within a given timeframe which does not influence the operability of applications that use the transition-enabled pub/sub overlay. Delivery delay should be minimized. PeerfactSim.KOM provides workload applications with an usual timeout of 30 seconds, so notifications must arrive at the subscriber within this timeframe, otherwise the publication will be marked as lost.

Portability and Testability For further studies, the transition-enabled pub/sub service may be deployed on real mobile devices for further evaluation of transitions. Since the prototype is implemented using the PeerfactSim.KOM simulator, this requirement is already accomplished. As it is shown in Chapter 6, the simonstrator-API already provides the needed modularity to deploy an overlay implementation on a real mobile test environment. The transition service will be also functional evaluated. That means that the state of the system can be easily observed and the performance of the system can be counter-matched to the requirements of a pub/sub overlay. The usage of the controlled simulated environment in PeerfactSim.KOM does help to accomplish this requirement.

In the following chapter, a design for a transition-enabled pub/sub service is presented which is based on defined requirements described above.

5 Design of a Transition-Enabled Publish/Subscribe Service

Based on the requirements stated in the previous section and the related work, the design of a transition-enabled pub/sub service is presented and discussed. Later, this design is implemented as a prototype in the PeerfactSim.KOM simulation environment. The details of this implementation are discussed in Chapter 6. The first part of the design chapter investigates the abstraction of a generic pub/sub service. Based on the abstraction layers, the different designed layers of a transition service can be inferred. For this purpose, two questions need to be answered: How can functional blocks be built for a transition service and for pub/sub overlays. How exactly is a transition defined in this context.

5.1 Abstraction Layers

The design of a transition-enabled pub/sub system distinguishes different layers of the system design. To understand these different layers of abstractions, they are viewed from different viewpoints. Interfaces are placed on the rims of these layers to merge the different parts of the system together. Chart 5.1 illustrates these different viewpoints of the transition-enabled pub/sub overlay. Raw pub/sub messages or pub/sub overlay specific information is transmitted using the pub/sub protocol, indicated by the colored boxes and arrows. The internal flow of information is also shown. Besides the pub/sub protocol, a distinct transition service related protocol is used. Filters on the communication interface route the encapsulated messages or piggybacked information to the corresponding modules.

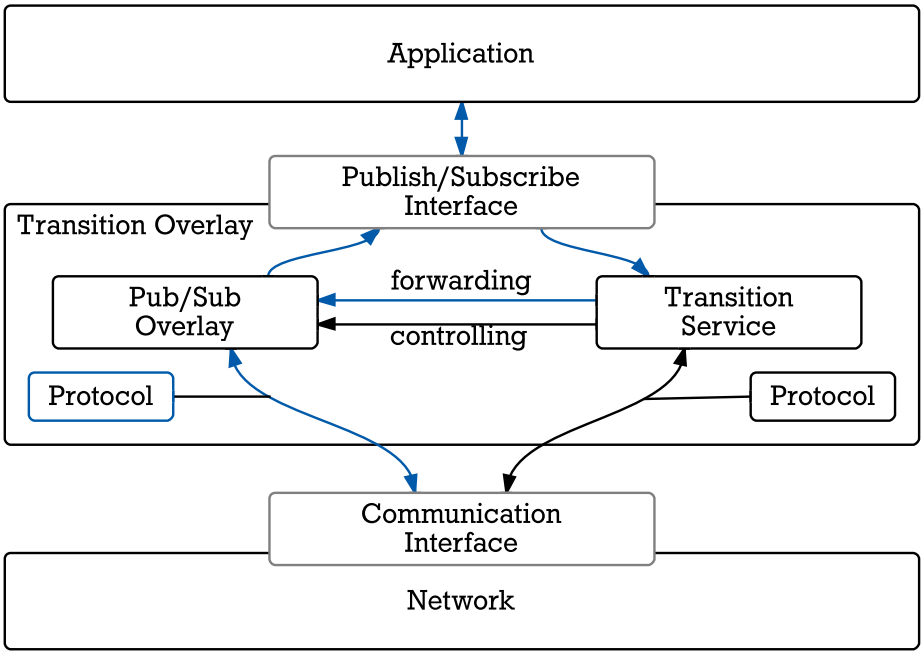


Figure 5.1.: Abstraction layers of a transition-enabled pub/sub overlay.

The first viewpoint is the user of the overall system. The user can either be a real person or an application, which uses the interface of the pub/sub overlay. He takes use of it to share information and

data between other users or applications of the system. When the user requests data of an object of interest, he mostly does not care about where information comes from or how data is routed through the network. Since the transition service takes use of different pub/sub schemes, a generic interface for all these schemes must be identified and placed into the design. The user however, does not have a clear view on which specific pub/sub scheme is currently being used by the transition service.

The second viewpoint is the view of the system administrator. The system administrator has special requirements regarding the performance of the system. He knows about the specific properties of the MANET being used and cares about how transitions do impact on the service and performance. The system administrator is the provider of the transition-enabled pub/sub service.

The last viewpoint of the transition service is the network administrator. He does not care about which types of messages are sent over the network. He has no correlation to the pub/sub system. However, the network administrator does have requirements about how messages are routed and where data is stored. This depends on the capacity of the nodes and the connection between them. He indirectly defines requirements for the transition service and impact the decision about when a specific pub/sub overlay is being used at a concrete time or system state. It must be noted, that on MANETs the network administrator does not exists, since MANETs are self organizing. In this case, the responsibility for network related issues is handled by the system administrator.

5.2 Transition Service

One focus of this thesis is to discuss the possibilities to switch between different pub/sub overlays. A component, which must be inspected, is the concrete method to trigger a transition between the pub/sub overlays. This component influences how transitions are triggered and how the participating nodes of the transition-enabled pub/sub overlay know to switch to a specific overlay. Another issue that should be discussed are tasks which must be performed before, during, and after a transition. These task have to fit the following requirements. The pub/sub service must still be able to accept new subscriptions and publications. It must also be able to notify the corresponding applications about matching notifications, therefore old subscriptions must be transferred from the old overlay to the new one. It is expected that all these task increase the workload of the system, on the nodes itself and within the network by transmitting management messages and status updates. Subsequently it is important, that transitions are triggered consistent to the expected behaviour calculated by the management components. The system should prevent situations where nodes switch the overlay uncontrolled in a non deterministic way. Following, a design of a transition service is presented with possible solutions for the identified challenges.

Figure 5.2 shows the concept of the transition service. It presents the distinct parts of the overlay and the interaction within the transition service. As already discussed, the transition service has a component for the transition engine and a component for the active pub/sub overlay. In distinction to stand alone pub/sub overlays, the storage for subscriptions and events has been moved out of the pub/sub overlay. The requirement of retainable storage, even during transitions requires to store information about subscriptions and events outside of the overlay that is being switched. The pub/sub component is an abstraction for every usable pub/sub overlay on the system. While the concrete pub/sub engine is responsible for the acceptance of subscriptions and publications, a subscription manager stores these subscriptions in the generic storage. This subscription manager also acts as filter and coordinator, if a subscription is not saved on the node itself, but on another broker node of the system. For example, for a density driven pub/sub overlay the subscription manager is responsible to send the subscription to the next hilltop, therefore it has direct access to the network. The event manager is responsible for storing events to maintain time decoupling for nodes that join the overlay lately. In case of the usage of advertisements, the event manager functions as broker coordinator to manage these advertisements. The last component of the pub/sub overlay module is the notification manager. This component is responsible for the final notification matching and acts as event dispatcher. Subscriptions are registered by

the pub/sub engine to be able to match incoming notifications. The transition engine constantly controls and monitors the system state to finally take actions if a transition from one pub/sub overlay towards another pub/sub overlay is required.

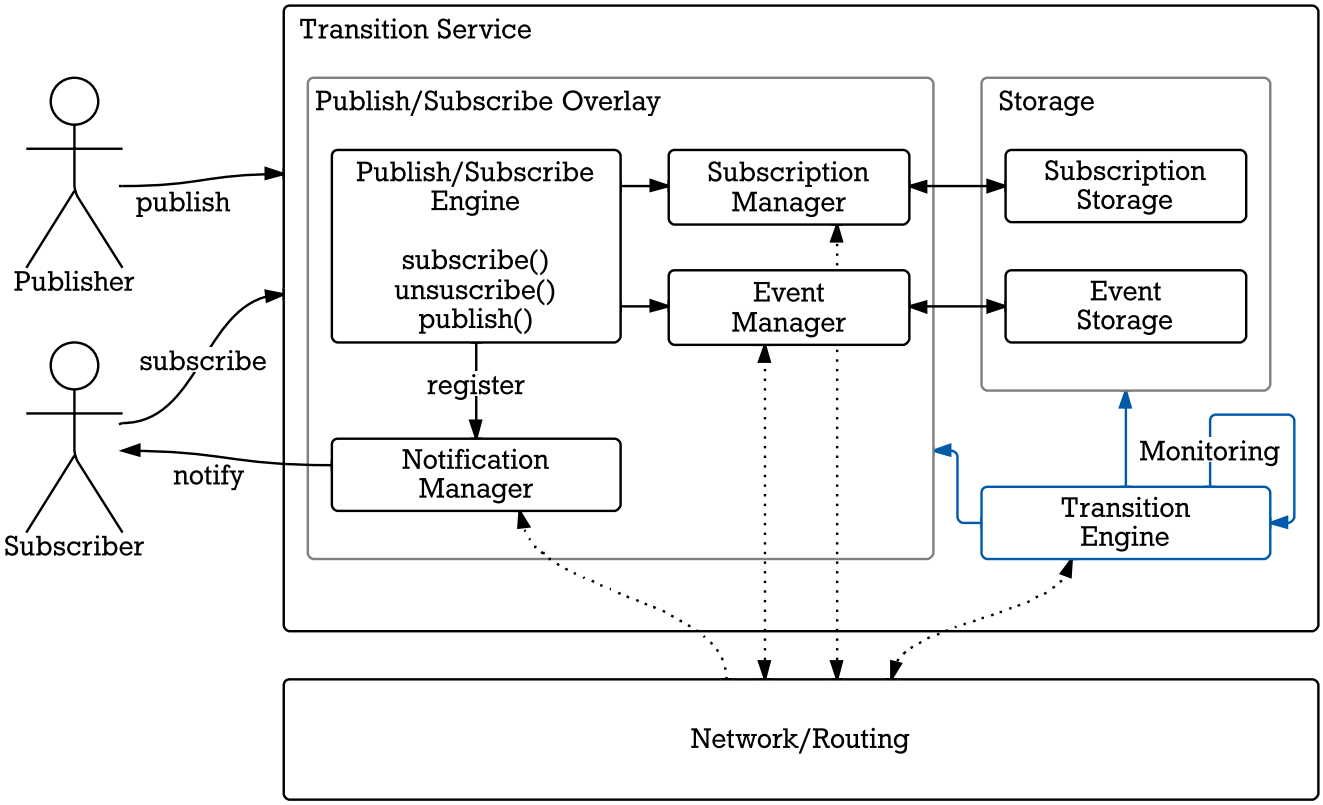


Figure 5.2.: The concept of the transition service with detailed pub/sub overlay components.

5.2.1 Transition Mechanism

This section covers the transition mechanism itself. In particular the required information which must be shared between the nodes, the way how this information is transmitted, and the steps of how management information is aggregated and used by the transition service are discussed.

In order to switch to another pub/sub overlay, the transition service first has to calculate which overlay is the best to be used on a given system state. The system state highly corresponds to the given scenario the transition service is planned to be used. Also, each later used pub/sub overlay uses different parameters to define its best to use situation. A density driven based overlay obviously would use density information, if the overlay is good to be used or not, as well as to maintain the density driven broker tree itself. For another used pub/sub overlay which uses only movement speed information, density information might be unnecessary and do not provide a good metric to calculate if the transition service should transfer to this overlay. The combined parameters of all used pub/sub overlays form the state of the system. The transition service must calculate these parameters constantly in order to calculate the best overlay to be used in the current system state. Since each pub/sub overlay comes with different qualifying metrics, the overlay must provide a way to calculate these parameters. It is also possible, that the transition service provides standard parameters like movement speed of nodes or single hop density. If a pub/sub overlay however uses for example reachability information, the overlay itself calculates these management information.

Once these parameters are calculated, the transition service collects all parameters from nodes in order to calculate the global system state. This can again be done on a single control and monitor node or distributed over multiple nodes. For that, it is possible that the monitor and control nodes of the transition service communicate with each other. Once a global system state is determined, the transition service is now be able to calculate the best pub/sub overlay to be used, using the system state. The overlays themselves have to provide conditions which directly correspond to the performance of the pub/sub overlay on the given system state. The conditions are prior defined, e.g. through benchmarking of the overlays. Based on these performance metrics, the transition service is able to choose an overlay to switch to. A transition can now be executed by notifying all nodes about a transition event.

There are different possible ways of how these management and control information is transmitted between the client nodes. One way uses distinct control messages which differ from the publish and subscribe messages the overlays use. These messages have the only purpose to maintain the transition service activities. An issue about this approach is, that these messages must be routed through the network to its destination. The design does not provide an explicit routing component for these type of messages. It relies on the routing mechanisms of the underlying transport layer. However, as messages which trigger a overlay transition should reach all nodes anyway, the easiest approach would be to simply flood these messages through the network.

Another approach takes use of the nature of a pub/sub service. All participating nodes subscribe to a specific transition related object of interest, for example the topic [*String*=transition, *overlay*=name, *long*=time]. All necessary information exchange is covered by publishing and subscribing nodes. If the management information are transmitted over subscriptions and publications, it is required that the transition service is always able to fall back to a robust default pub/sub strategy. The service can also not be run separately without an active overlay in order to perform only management and testing operations on the system. In such a case, the pub/sub service would be temporarily deactivated but nodes would still be able to join and leave the network in a deterministic way if needed. Also if the active overlay follows an eventually consistency strategy, the distribution of management information and transition messages are not guaranteed to reach the the designated nodes in time. Overlays which follow such a strategy must be marked and the service would not use it as distribution overlay for the management information. Nevertheless, the greatest advantage to use the active overlay as distribution overlay is that management information is always distributed on the most efficient way, assuming that the active overlay is the most efficient overlay in the current system state.

5.2.2 Transition Strategy

The transition-service is responsible for the calculation of the best overlay to be used. Once an overlay is chosen as best overlay in a given state, the transition service performs a transition from the active overlay to the new one. However, a transition can follow different transition strategies to reduce the impact on the performance of the overall pub/sub service. A simple solution to execute a transition is that the trigger immediately reacts to a change of the determined network and overlay state. This means that as soon as a better overlay for a given system state is calculated, the transition service executes its transition routine, which switches the overlay to the new overlay. Although this solution of a transition strategy can be implemented easily, it has no use in a real system design. In fact it has probably a negative effect on the whole system, if the state of the service fluctuates on a threshold right between two good solutions of two competitive overlays. The rapid oscillating transitions between these overlays would take up all resources of the pub/sub service until the system falls in a stable state where only one overlay is constantly determined as the best overlay to use.

Due to this reason, it is required to provide an intelligent module which capsules the transition strategy. That module must be deeply integrated in the system so that it can refuse a transition, even if a better overlay is calculated on a given system state. Once again, two possible design solutions of such a module are provided and further discussed. The first module designs a hysteresis component to prevent

transitions between overlays if the time to the last transition did not exceed a given delay. The delay time must be at least as long as it takes to execute a transition and to give the system the time to reach a stable state after a transition. It must also grant sufficient time, so that the clients can take advantage of the currently active overlay to execute their publish and subscribe operations if they are interrupted by the transition operation. Another way to implement the hysteresis of transitions is the use of the calculated performance metrics of the used pub/sub overlays. Since the transition service constantly monitors the system state, a transition can be triggered only if the overlay of choice does exceed the current pub/sub overlay by a defined threshold. This hysteresis approach assumes that the performance metric of different pub/sub overlays only change in a predictable way, i.e. no sudden peaks occur.

The time it takes to reach a stable state of the network varies depending on the given scenario. For example, a higher workload increases the time it takes for the client to manage old and ongoing subscriptions and publications. Other external circumstances like the velocity of the nodes can increase the fluctuation rate between the calculated values of overlays, once again increasing the risk to fall into a blocked system state where transitions are executed in a rapid way. An adaptive transition strategy can remedy this by responding to these external influences and adaptively change the delay in within a transition is not allowed to be triggered, depending on performance metrics of the overlays and the time. This can be done directly by inspecting the distance between the calculated performance metrics of the active overlay and the best to use overlay. Since the calculated performance metrics of the overlays are also responding directly to the defined parameters of an overlay, the transition delay is consequently adjusted to a given system state. However, the adaptive transition strategy can also monitor the rate of transition requests and increase the delay between transitions every time a transition is executed, indirectly reacting to rapid state changes of the whole system. Since the time of a planned transition plays a major role within the whole transition process, it is requested that the transition service design not only provides the possibility to inform a participating node directly but also inform the nodes about planed overlay changes. All presented transition strategies are implemented, evaluated, and the results are presented in Chapter 7.

Up to this point, Figure 5.3 illustrates the steps which must be taken by the transition decision process. Each module is exchangeable to provide an adaptive system design for various scenarios.

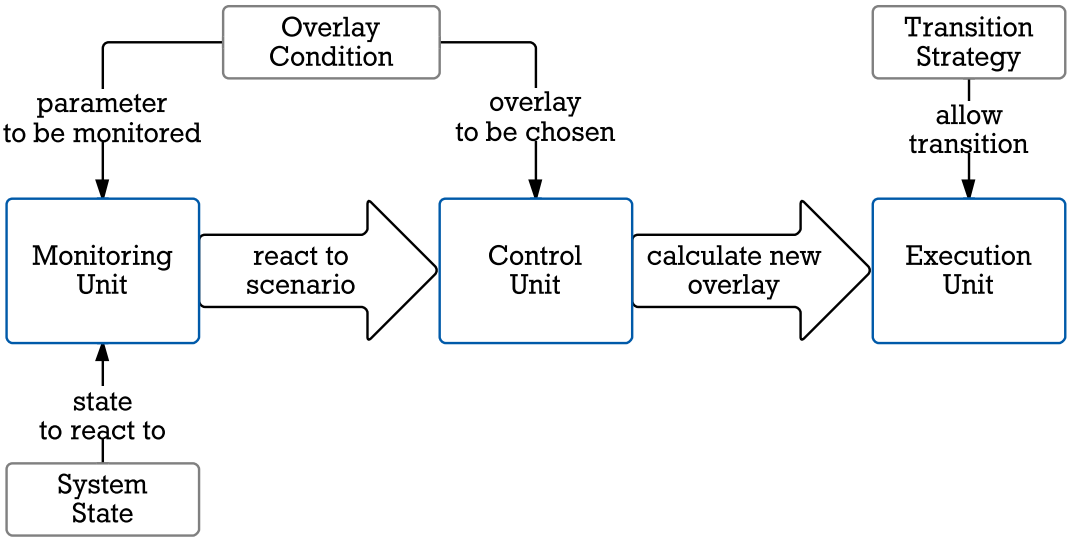


Figure 5.3.: State diagram of the transition decision process.

The monitoring unit is responsible to constantly monitor the system, network, and overlay state. Depending on the overlay conditions, the parameters, which also form the performance metric of an overlay, are chosen to be monitored. The control unit calculates the concrete system state and chose an overlay

depending on the overlay condition. Finally the transition strategy decides if a transition is allowed to be triggered by the execution unit. The implementation of this design is presented in Chapter 6.

5.3 Concept and System Architecture

One important part of the design is the modeling and abstraction of the pub/sub domain. Figure 5.4 presents an overview of the different acting parts of the system, as well as control flows.

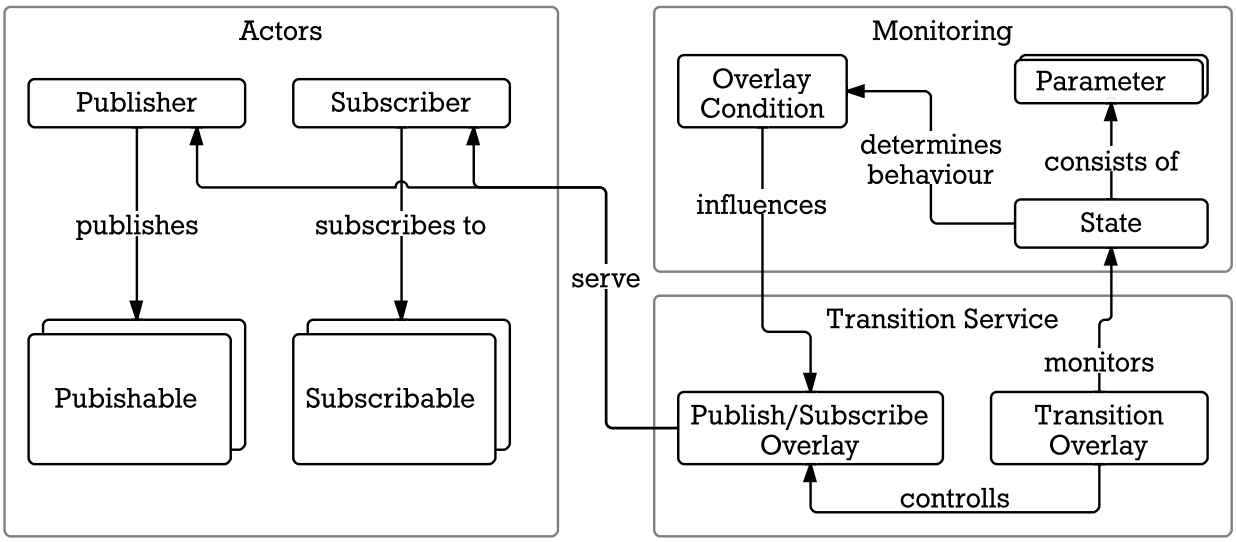


Figure 5.4.: Domain architecture of the transition-enabled pub/sub service.

On the left side, the actors of the system are shown as publishers and subscribers. It is possible that a publisher and a subscriber are the same. Usually these entities are users or applications on a node of the pub/sub overlay. To cover the different approaches of publish/subscribe systems, like subscription model, publisher and subscriber disseminate abstract publishable and subscribable objects. These can be further implemented as topic, channel or subject depending on the used pub/sub overlay.

The most important part of the domain in the pub/sub service, which is the actual transition enabled pub/sub service, consisting of a transition engine and many pub/sub overlays managed by the transition overlay. The pub/sub overlays serve the publisher and subscriber by providing a generic pub/sub interface. On top of the pub/sub service the concept of a monitoring domain is placed. Monitoring of the state of the P2P system is important to determine if a transition should be executed. The state of the system consists of abstract parameters, which again can be implemented as relevant attributes of a P2P system, and simultaneously serve as performance metric of a pub/sub overlay. Since the state of the system is not enough to decide if a transition is appropriate, conditions for different pub/sub overlays are included. The conditions determine the behaviour of the transition service and are based on the monitored states and parameters of the states.

5.4 Publish/Subscribe Overlay Layers

This section covers the properties and interfaces of a generic pub/sub overlay. Unlike a stand alone version of a pub/sub overlay, the overlays used in the transition service have to provide additional information and components to be used properly. As already mentioned, they have to provide metrics to define the performance of a pub/sub overlay on a given system state. The calculated parameters are matched to conditions for each overlay by the transition service in order to determine the best overlay to be used. During a transition, ongoing subscriptions and publications must be dropped by the

current active overlay and transferred to the new one, so that the space and time decoupling can still be maintained by the superior pub/sub service. This process has to be transparent for the application.

A generic publication and subscription storage is provided by the transition service so that ongoing subscriptions and publications can be stored in an overlay-independent way. This archives the needed abstraction layer between the different overlays and their ways to handle subscriptions and publications. Each overlay can also store additional information for each subscription or publication in it to perform its functionality. For example, a broker has to store a corresponding client address which will be notified later in addition to a subscription. However, another pub/sub overlay is still able to read the information needed to generate its own subscription and publication tables.

Different pub/sub overlays may save information on different nodes (for example central broker vs. distributed brokers). As a result of a transition, such publication and subscription information might need to be transferred from the storage of the old node to the storage of the corresponding new node. The transition algorithm can handle this activity by splitting and collecting publication and subscription information and transferring it to the nodes. The new pub/sub overlay performs a warm start where publications and subscriptions are already stored in the system. However, in this case, the transition algorithm must be aware of the structure of the pub/sub overlay so that the required storage information can be transferred. To handle this problem and to keep the transition algorithm simple, the new pub/sub overlay can also perform a cold start without any publication or subscription information saved. Old subscriptions and publications are injected locally once again on their original nodes. By that, the pub/sub overlay itself does transfer the information to the corresponding nodes responding to its internal structure. In this case, the publication and subscription rate is increased artificially until a stable state of the system is reached. The transition algorithm does not perform metric calculations for the time of the transition to prevent incorrect conclusions. If hysteresis to the transition decision is applied, monitoring directly after a transition cannot influence the transition decision anyway and therefore can be paused.

5.5 Component Structure

In this section, specific design decisions of the concrete service are presented. Figure 5.5 presents the component structure of the transition-enabled pub/sub service. This diagram also acts as guide for the following chapter, where the implementation of the transition-enabled pub/sub service is discussed.

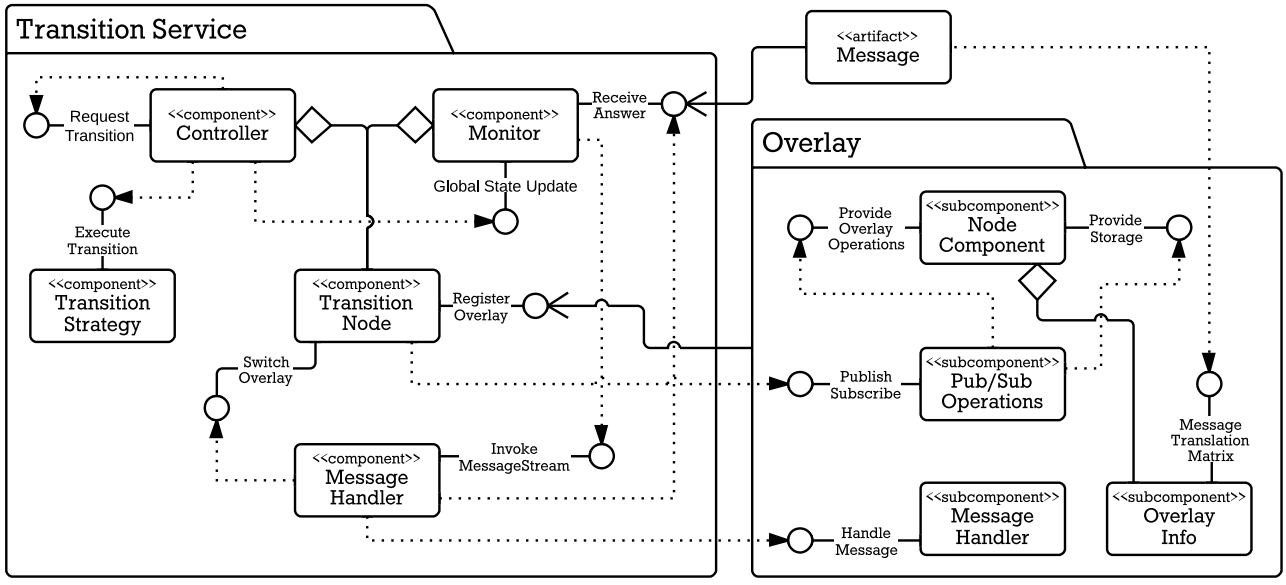


Figure 5.5.: Component diagram of the transition-enabled pub/sub service.

As it can be seen, the system is divided into the transition service package and an overlay package. The transition node component contains the controller and monitor component which do execute management functions by itself, like updating the global state or executing transitions. The overlay info component in the overlay package does provide all necessary information for the transition service, like favourite state conditions or a message translation matrix for the translation of messages between different pub/sub overlays. Another important component is the pub/sub operations component. This component is the pub/sub interface for an overlay, providing the pub/sub engine. The key interface however is the switch overlay interface provided by the transition node component. This interface is responsible for switching the current overlay on a node. Typically, transitions are triggered by messages, which by itself are handled by message handlers. Further details of the service are discussed in the next chapter.

6 Implementation

This chapter presents the implementation of a transition-enabled pub/sub system called TRANSOS. TRANSOS is a prototypical implementation of the design presented in Chapter 5. The main system contains the transition system, controlling, and monitoring mechanisms. In addition, the system is equipped with three competitive pub/sub overlays to provide a variety of different pub/sub mechanisms and to examine the switching protocol. However, TRANSOS can be extended with an arbitrary amount of optional overlays to enhance its functionality and performance as whole pub/sub service. TRANSOS is implemented using the Java based PeerfactSim.KOM simulator, which is shortly presented in the following section.

6.1 Overview of the P2P Overlay Simulator PeerfactSim.KOM

This section briefly describes the structure and the operation of the PeerfactSim.KOM [SGR+11] simulator. The transition system does directly use the simulators API. PeerfactSim.KOM is a discrete-event P2P simulator, written in Java. It is still further developed to reflect a wide range of scenarios in the area of P2P. The simulator has a modular architecture. These modules are stackable into different layers of abstraction. Each layer represents a part of the ISO/OSI-model of communication systems [Tan81] with its different protocols and mechanisms. The simulation framework comes with a variety of modules containing protocols for overlays, applications, and routing algorithms. PeerfactSim.KOM also comes with a large number of logging, statistics, and visualization capabilities.

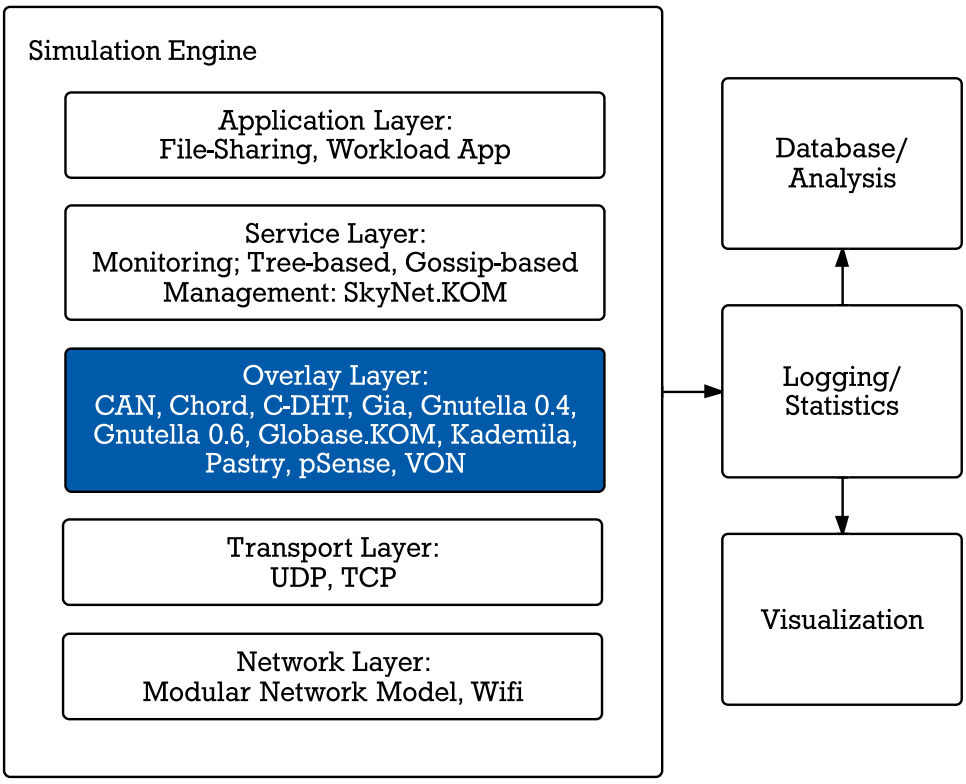


Figure 6.1.: The modular architecture of PeerfactSim.KOM (adapted from [SGR+11]).

Figure 6.1 illustrates the modular layer architecture of PeerfactSim.KOM. The transition enabled pub/sub prototype will be located in the marked overlay layer. Before accurate linking of the individual components is presented, the individual layers are presented shortly.

Network Layer and Movement Model

The network layer focuses on the simulation of data transmission between arbitrary peers of the considered network. This is done by using mathematic formulas and functions to model a connection between the nodes. This can be for example a fix connection or a mobile interconnection, with its special characteristics like a covering range and transmission collisions. Also other influencing factors like latency, jitter or peer positioning can be simulated. For the simulation of the transition overlay a wifi 802.11 layer is used. This module simulates a mobile connection with a peer range of about 220 m. The range of each node is also varied by a small amount randomly. Beside the range of the peers, it also models the upload and download capabilities for each node. One part of the network layer is the movement model used to simulate the chosen scenario. There are a couple of models available, but the evaluation of the transition-enabled pub/sub service focuses mainly on a random waypoint movement model [SMO06]. The random waypoint movement model basically moves the nodes towards a random destination on the map with a predefined speed. If the destination is reached, a new destination will be chosen by each node separately. The time between each step and the velocity of the peers can be chosen freely. Also a gauss markov movement model [CBD02], and a social movement model [HN11] is available. The gauss markov movement model simulates more realistic movement paths for mobile devices, without sharp edges or sudden motion changes. The social movement model emulates group behavior with the formation of local groups around a point of interest, which does fit into the scenario chosen for the transition system. PeerfactSim.KOM allows a flexible configuration of the network layer with different sub-net strategies for a wide extension of the simulation environment.

Transport Layer and Simonstrator-API

The transport layer module of PeerfactSim.KOM provides an end-to-end communication service for all layers above. Similar to the network layer, it comes with a couple of protocols ready to use, which do allow addressing of single peers. It models an entity of a connection between different peers. Each transmission in PeerfactSim.KOM uses messages as abstraction of transmitted packets or chunks. A message is an object in Java, with a defined size to calculate the transmission time, that is shifted around between the peers. If the system is deployed on real separate mobile devices, these messages are serialized using Java serialisation capabilities. On the receiving mobile device the massages are deserialized back into Java objects. The transition service uses UDP-messages for the transmission of messages. Lost messages or flow control is handled manually by the transition-enabled pub/sub service or the pub/sub overlay. This layer can be directly called by using a `sendMessage()` or `sendBroadcast()`-function. `sendMessage()` sends a message directly to an addressed peer, while `sendBroadcast()` does not have a direct recipient. A `sendAndWait()`-function provides a simulation of timeouts if for example the addressed peer is not in range of the sender or a collision occurs. The receiving peer of a message with is addressed by the `sendAndWait()`-function has to manually acknowledge this message with a generated acknowledge-message. This massage must not be counterchecked against the messages sent by `sendAndWait()`, since this is already done by the simonstrator environment. This simplifies protocols on the next layer, since they do not have to maintain an own timeout module. However, if a message is targeted at multiple peers, it is better to use the `sendBroadcast()`-API. The reception of messages is modeled in message handlers. These handlers provide functions that are executed on the reception of a message. Messages can simply be forwarded by broadcasts to build a flooding protocol or they can start a complex calculation for data requests. If a message is not forwarded by a peer, the message simply is discarded. This, of course, only matters if a message is sent as broadcast. Unicasts, that are addressed to a specific peer, are filtered out on the network layer. The main difference between broadcasts and unicasts is, that broadcasts sent by a peer do not need to be prepared in terms of agreements between the peers.

Before a unicast is sent, request-to-send (rts), clear-to-send messages (cts), as they are defined in the IEEE 802.11 standard, control the transmission. So, broadcasts are more likely to produce collisions than unicasts, but they do need less overhead. Depending on the overlay and the requested service, these different transmission-types must be balanced against each other. In both cases, if a message collides or does not reach the receiver, it must be resent manually by the overlay.

Overlay Layer and Publish/Subscribe-Interface

The most important layer of PeerfactSim.KOM is the overlay layer, since it contains the implementations of the different P2P overlay models. This is also the layer where the transition system is placed in. However, the transition system splits this layer into the essential transition service and the pub/sub overlays. PeerfactSim.KOM already comes with a huge variety of overlays, like the in Chapter 3 mentioned PASTRY overlay. Since PeerfactSim.KOM focuses on the simulation of peers, this layer can be used by implementing an `OverlayNode`-interface. An overlay node is a peer of the network with all its functionality. PeerfactSim.KOM also comes with a couple of already premade extensions of the overlay node, like the `JoinLeaveOverlayNode`-interface. Join/leave-nodes provide additional functions for joining or leaving the network when they go online or go offline. The transition system uses the `PublishSubscribeComponents`-interface for implementing its pub/sub capabilities. This sort of nodes must provide `publish`, `subscribe`, `unsubscribe`, `advertise`, and `unadvertise` methods, which are used by the application. This does model all necessary pub/sub operations. With a skeletal implementation of an abstract overlay layer, PeerfactSim.KOM provides an easy access to integrate a new overly into the layered architecture of PeerfactSim.KOM.

Service Layer and External Monitoring

The service layer of PeerfactSim.KOM offers additional services for an application or the whole system. This layer allows to put additional components between the overlay layer and the application layer. Due to its global knowledge, it is able to monitor the activity of the overlay layer. Using this layer allows easy access to attributes of the P2P system. The evaluation of the transition system does not take use of the service layer of PeerfactSim.KOM, since a complex monitoring layer is not needed to be used. However, a service layer monitoring component could be used as external monitoring component for the transition system to transit to its different pub/sub overlays. For the prototype implementation, the transition system uses an own implementation of a monitoring component.

Application Layer and Workload

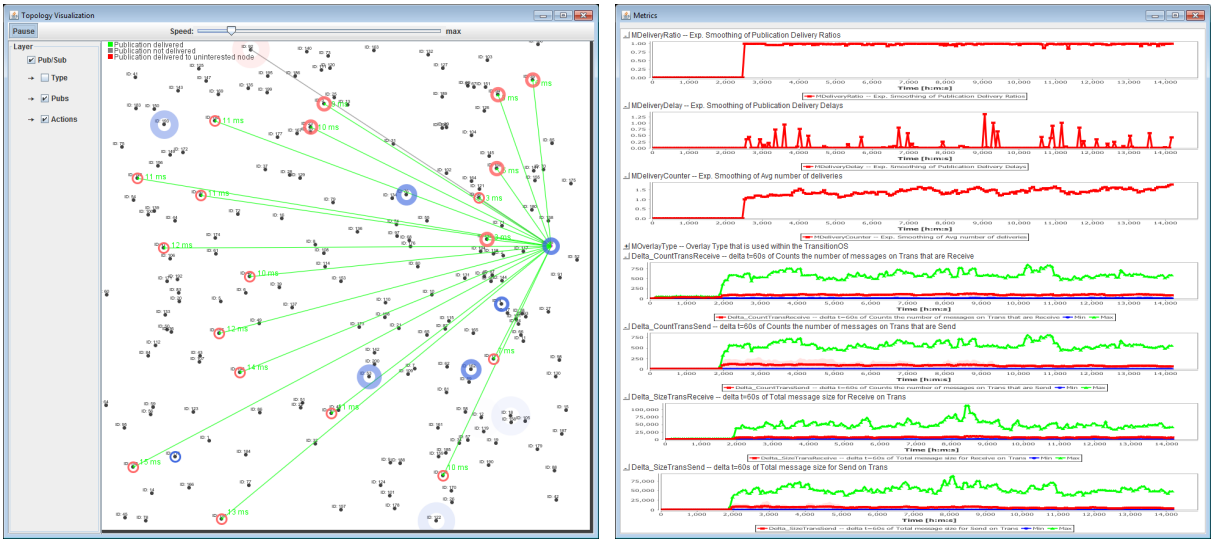
The application layer is the user part of a complete simulation environment. Depending on the type of scenario, the application layer utilizes the layers below by triggering special events or requesting data from it. The transition system and the pub/sub overlays are using a workload application which constantly generates publications and subscriptions on different nodes. By this, it allows to test the whole system and evaluate it later. The workload application also evaluates the requirement of correctness of the transition system by testing if a publication arrives the correct nodes which are interested in a topic.

Visualization and Data Logging

Besides the different layers, which to model the whole P2P system, PeerfactSim.KOM comes with a large number of testing, evaluating, and visualization capabilities. A logging component collects data during a simulation. It is able to access important P2P attributes like packet loss and overhead and makes it available. Based on the collected data, statistics can be generated and evaluated against each other between different simulation runs. The collected data can be stored in a local or external database for later use or displayed on-the-fly during the simulation.

In addition to the logging and statistic components, a visualization component allows to show a live illustration of the P2P system. The default visualization component of PeerfactSim.KOM provides a overhead view of the network with dots representing the different nodes. The dimensions of the view are

equivalent to the simulated size of the scenario. The visualization component can be extended with different functions, depending on the simulated system. For example, each node can be colored depending on the active pub/sub overlay. Lines between peers represent messages or received notifications from another peer. While the visualization of a system is not so important for its evaluation, it provides an easy inspection method for the implementation of the transition system and its pub/sub overlays. By this, it also fulfills the requirement of testability of the transition system. Figure 6.2 shows snapshots of the PeerfactSim.KOM live visualization and statistics generation.



(a) Live visualization of the overlay **(b)** Live plotting of performance metrics

Figure 6.2.: Live visualization of a scenario (a) and plotting of performance metrics (b) in PeerfactSim.KOM.

The visualization window shows a simulated area of 800 m × 800 m with 300 peers. Each node has colored rings around it representing the publishers and subscribers. Lines between the peers representing successful delivered publications from the publisher to the interested node. The statistics window shows a live generated and during the simulation continuous updating diagrams, representing the mean delivery ratio of publications, mean delivery delays between a publication and the reception on a subscriber, the total number of received messages on the transport layer, as well as the total number of sent messages on the transport layer. The absolute difference between the last two values represents the total number of lost messages.

6.2 TransOS – A Transition-Enabled Publish/Subscribe System

TransOS is a implementation of a transition-enabled pub/sub system using the PeerfactSim.KOM simulation environment. This part of the thesis presents how this system is implemented based on the previously discussed design. Factory generated `TransOSPubSubNodes` contain and provide all functionality for each node of the simulated scenario. Each node contains a bunch of sub-components for different pub/sub overlays, as well as monitoring and controlling functionality.

As mentioned in Chapter 5, the transition system uses abstractions for items that can be published or subscribed to, named publishables and subscribables. Normally, these are the abstractions of items from different pub/sub approaches like channel or subject based approaches. For simplification purposes and better integration of other pub/sub projects, the pub/sub environment of PeerfactSim.KOM does only provide topic-based and attribute-based pub/sub components. Nodes can subscribe to different topics and are notified on a topic using notifications. These topics consists a character string modeled as Uniform Resource Identifier (URI). This does enable the possibility of creating channels by using prefixes as categorization. A string of `/broker/publication` does create a topic space representing a channel,

which in turn represents the correlated subject. With the use of attributes, this concept can be enhanced with further semantics of a topic, building a content-based pub/sub system. TransOS uses these topics from the PeerfactSim.KOM environment and therefore is able to represent every possible pub/sub overlay that might be used within the transition service.

6.2.1 Transition Realisation

In order to switch from an overlay to another one, the overlay itself is contained in a subcomponent module. These packages represent one specific pub/sub overlay. Each pub/sub overlay module must implement a few interfaces provided by the transition system. The `PubSubOverlayInfo`-interface provides all necessary information of the pub/sub overlay, like the environmental conditions the overlay should be used in, the used message types, and the parameters that must be monitored by the transition system. This information is important for the transition service to manage and control the overlays. The second interface that must be implemented by all used pub/sub overlays is the `PubSubOverlayNodeComponent`-interface. This interface contains the actual implementation of the pub/sub overlay with its pub/sub operations, message handlers, and storage. While the overlay information is unique for each pub/sub overlay, the node component can be used multiple times, depending on the nodetypes of the actual pub/sub overlay. For example, an overlay can consist of broker node components and client node components. At this point, the pub/sub overlays must provide an access to the local pub/sub storage of each node. This is used to transfer the information about subscriptions and events from one overlay to another overlay. Subscriptions, publications, and advertisements, as well as their undoing are directly forwarded from the transition service to the pub/sub operations of the node component. Pub/sub modules can be added and removed at run-time from the transition service. If frameworks like OSGi [MVA10] are used which do allow the dynamic management of Java modules into the Java/VM environment, the code of each pub/sub overlay can be loaded and changed during the run-time of the transition-enabled pub/sub system.

6.2.2 Transition Protocol

In this part, the transition protocol is discussed, as well as the concrete mechanics and components used in the TransOS prototype. The transition protocol itself, as it is implemented in the prototype, passes through several phases, including network and overlay analytics, calculation phases, and transition execution phases. Sequence diagram 6.3 presents the execution of one management cycle.

By triggering the global state update routine at the control component of a node, the control component requests the overlay and network state from a monitoring component. TransOS only uses control and monitoring components which are on the same node and only one per transition overlay, but due to its modular design it can be easily extended to multiple and different nodes. The monitoring component now sends a `StatusRequest`-Message as broadcast to all other nodes, requesting all parameters defined by all used pub/sub overlays. The state of each node is replied via a `StatusUpdate`-Message as unicast to the requesting monitoring node. After a short qualifying waiting period the state of the node-cluster containing all nodes that answered the state request and returned to the control component. The control component now determines if a transition to a different pub/sub overlay is required. If a transition of the pub/sub overlay is required, this decision is broadcasted using a `SwitchOverlay`-Message to all nodes, forcing them to switch the active overlay. This sequence is repeated infinitely after a waiting depending on the used transition strategy.

Three transition strategies are provided for testing and evaluation of the system. A simple transition strategy, a hysteresis transition strategy, and an adaptive transition strategy. The simple transition strategy does trigger the transition between different pub/sub overlays immediately as a better overlay is calculated from the control component of TransOS. The monitoring interval of this strategy is set to 33

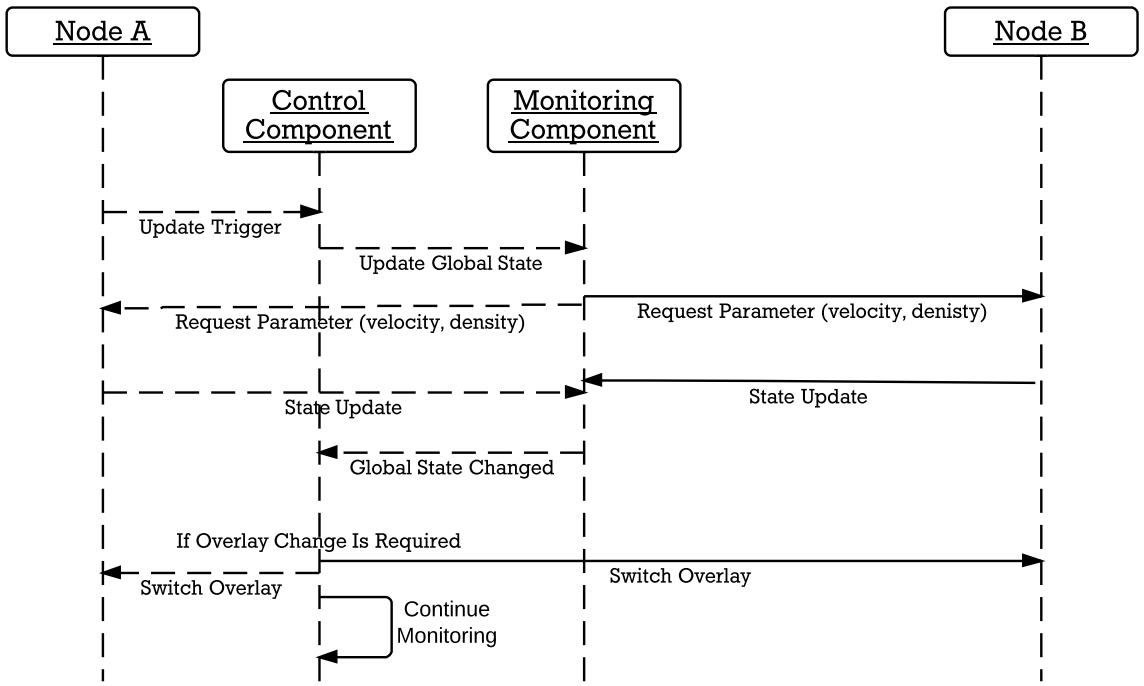


Figure 6.3.: Sequence of a transition in TRANSOS, with a control and monitoring component on one node and a separate peer.

seconds to achieve a fast response time on fast evolving network structures. The hysteresis transition strategy stops repeatedly switching of overlays in a rapid way by blocking all monitoring and transition operations for five minutes after a transition on the system occurred. This reduces overhead and high utilization of the network on specific situations. While the hysteresis transition strategy provides a time based mechanism to improve the transition system, the adaptive transition strategy also includes a delta value based on the calculated results between the active overlay and ready to use overlays. Monitoring operations can be paused up to ten minutes if the network reached a stable state and no change on the system can be measured. The intervals are adaptive chosen of and the time between monitoring operations is reduced if the transition system detects that the active pub/sub overlay becomes less performant than the cycle before. However, if a external monitoring component is used which has no impact on the system itself the provided monitoring intervals from the transition strategies can be ignored. In this case they only determine if a transition can be triggered or not.

6.3 TransOS Publish/Subscribe Overlays

As already mentioned, TRANSOS comes with three different pub/sub overlays which do provide a basis for testing. The following section briefly presents these pub/sub overlays. The different overlays were chosen to show a wide spread of different possible overlay mechanisms. It is also expected, that these mechanisms show a different performance under certain conditions, while they are choosen to cover a huge spectrum of different conditions.

Pub/sub overlays in TRANSOS sending messages to propagate information about subscriptions and publications between the nodes in the overlay. Excluding raw management and controlling messages, messages which carry a publication or subscription of an overlay use generic interfaces of that type providing general information about the notification or subscription. This generalisation allows to translate these messages from one overlay into another overlay without discarding and losing the whole information. This is needed if some nodes did not proper switch to the main pub/sub overlay used in the

transition service, which might occur when they went out of range for a short moment of the switching message was lost on its way to the node. An pub/sub overlay of TransOS might provide a translation matrix to integrate pub/sub message into its overlay. This interface provides a private network-to-network interface (PNNI).

6.3.1 Single Broker

The first pub/sub overlay discussed is the single broker overlay. In this case, one node participating in the overlay is elected as broker. The election is placed at the initialization of the overlay and might consider different circumstances to make its decision. For example, the election chooses the node with the most storage or power reserves, like a stationary central server. It is recommended that a broker is found with the most uptime, since this overlay relies on the provided services by the broker. Also mechanisms has to be found to detect if the broker went offline so that a new broker can be elected. These mechanisms are also placed in the overlays own coordination functions. If the single broker is lost due to several reasons, also all previous made subscriptions and publication saved on the broker are lost. After the election of a new server, every client must subscribe again the topics they are interested in, which might result in an increased traffic for the time of the broker initialization.

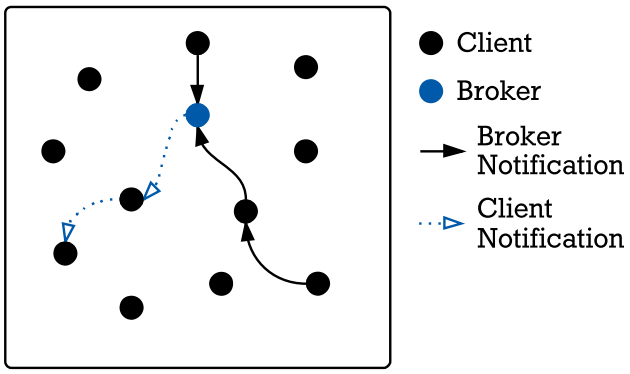


Figure 6.4.: Concept and overlay view of a single broker overlay.

As shown in Figure 6.4 one node is chosen as broker. Every publication and subscription is sent to that node. If another node is not in direct communication range of the central broker, the messages are routed using multi-hops over other client nodes. Publications are matched against subscriptions at the central broker and the corresponding client nodes are notified. For this reason, the single broker maintains a directory with subscription/client pairs, where each subscription is correlated to a client. This client will be notified by the broker if a matching publication arrives the broker. Exceptional to this overlay is the use of a global knowledge routing algorithm. This routing algorithm access the topology information provided by the simulator. Unicast messages are automatically forwarded along an optimal path determined by a dijkstra algorithm. This is done to simulate a most optimal routing algorithm to reach the single broker in the overlay.

Since the focus of this thesis is placed on the evaluation of the transition service, some mechanisms of the single broker overlay are simplified. The node with the lowest overlay ID is chosen as broker. This does also reduce the need of complex protocols for server election for this overlay. It is expected that this overlay shows its best performance in an environment with slow movement and a high spot of node density where in the middle of this spot the broker is placed in. To reduce the traffic of the network, the broker node with the lowest number of hops to all other client nodes should be chosen as single broker. Each hop of a message does block the possible transmission-space of the network. By reducing the number of hops to the broker, a decreasing possibility of collisions can be achieved.

6.3.2 Distributed Broker

The second pub/sub overlay follows an entirely different approach. In the distributed broker overlay, each client node is its own broker, so every node is broker. This approach comes with some advantages and some disadvantages. With the lack of central coordination, the node-to-node protocol can be held very simple. No special arrangements between peers must be established. If a peer is interested in a topic, it subscribes to that topic locally. Publications which might be published on a different node are now sent to all other clients participation in the overlay. A very simple flooding protocol can be used to send the messages through the network. At each node hop, the publication is resent to the from the source node further distant nodes. Another way to implement a distributed broker overlay is to send a subscription to all nodes, rather than sending the notification. In this case, each node knows about every subscription/client pair. Publications can now again be merged locally and the interested peer can be notified directly without flooding the rest of the network. In this case a better routing algorithm can be chosen which improves the performance of the overlay, especially if the publication rate is very high. Subscriptions however are still sent to every node using a flooding based algorithm.

The distributed broker overlay is implemented using the first subscription model, meaning that only notifications are sent over the overlay. This is done because the goal is to evaluate transitions between different pub/sub overlays. The performance of subscription flooding approach is highly depending on the used routing algorithm for end to end node notification. Also no special actions must be taken if a node joins the overlay lately. Due to its simple structure, the distributed broker overlay provides a very failsave overlay for the transition service which is important to qualify the system for the requirement of operability and stability. This overlay is used as default overlay to provide a stable fallback strategy.

Figure 6.5 shows a scenario with an active distributed broker overlay. In this case, publications are sent to every node of the network. Subscriptions are merged locally on the nodes. The figure illustrates a view hops of a possible path taken by a notification. Two additions to a simple flooding based routing protocol are made to improve the system. First, peers wait a random amount of time before they forward a message with a mulitcast. This is done to prevent collisions if all neighbours of a node forward a message at the same time. Second, random early discard (RED) is used to determine if a node forwards a message or not. If a message has been already forwarded in the neighbourhood, it is dropped with the probability depending on the number of occurrences. This reduced the bandwidth used by the overlay.

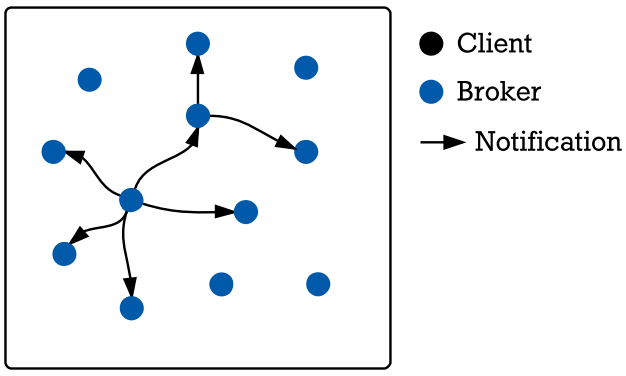


Figure 6.5.: Concept and overlay view of a distributed broker overlay.

Since this overlay orients on the SIENA pub/sub system (see Section 3.1.1), it is particularly well suited for an environment where connections between nodes do not consist very long. This is especially the case where nodes travel with a high movement speed through the network. For that reason, conditions for this overlay are a high movement speed of the nodes, a high changerate of the local density at the nodes, and a low publication rate.

6.3.3 Tree Broker Structure

The tree broker pub/sub overlay models an overlay between the single broker and distributed broker extremes. This overlay also represents an overlay with a more complex structure and protocols. The tree broker overlay contains normal client nodes, where a few of them are elected as broker node. A client node who is not a broker is correlated to a specific broker node. This broker node does store the subscriptions of this node and the subscriptions of other client nodes which are assigned to this broker. Broker nodes are chosen so that each client has a one-hop connection to its broker. Publications are pushed as follows. If a client node has to publish a publication, it sends a notification to its broker node. This notification now is sent to all neighbour broker nodes of the original broker. These brokers now once again forward the notification to all neighbour broker nodes whereas all brokers that are already notified are excluded. If a publication is generated at a node which is a broker, it is sent directly to all other brokers. Publications are merged against subscriptions at the broker nodes and the client nodes are notified by the broker nodes.

The broker tree forms a broker substrate, where the brokers are always directly connected with each other. In order to elect specific nodes as broker nodes, peers that are already broker periodically broadcast a hello-beacon-message to all neighbour nodes. The information about all broker neighbours is maintained on every node. Clients chose one of these broker neighbours as personal broker and subscribe on this broker with their interests. Brokers use this information to forward subscriptions to these brokers. If a hello-message is no longer received for a specific time, the broker contact is removed. This is also the case if a the reception of an unicast message to a broker is not acknowledged. Similarly, brokers do delete subscriptions of clients if they do not confirm a notification. The broker does assume in this case, that the client is no longer active or switched its state. Table 6.1 gives a short overview over the used parameters of the tree broker overlay, which worked very well in the execution.

Parameter	Value
Hello Cycle Time	10 s
Neighbour Timeout	20 s
Neighbour Validation Cycle Time	10 s
Retransmission Timeout	2 s
Transmission Retries[1]	∞
Client Backoff	0 s – 2 s
Broker Backoff	2 s – 10 s
Trail Size	∞

[1] With new broker or fallback as being own broker.

Table 6.1.: Standard parameter values for the tree broker overlay.

The broker substrate has to maintain a certain density of brokers in order to cover the whole network and to achieve the one-hop client/broker connection. For this, peers can switch their state and become a broker or a client. To decide weather this is necessary, all nodes use the broker neighbour information to calculate strong connections between the brokers. If a partition is detected, a client becomes broker of the overlay. This switch is secured with a random backoff procedure to prevent that too many clients become broker at the same time. Clients that became broker immediately produce a broker hello-message to trigger the backoff algorithm at the other clients. Also, brokers become clients if they detect that the neighbour brokers are already strong connected. Brokers that become a client also send a goodbye broadcast to all neighbours. These nodes can use this information to remove the broker from the broker

table and if it is necessary subscribe at another broker. In this case, the time it takes for a timeout is saved. Using this strategy, the brokers are always set in an area with higher density. This is mostly the case in the middle of the covered area of an ad hoc network. At the edge of the network, the probability that a node is client is higher than in the middle of the network.

Using such a broker tree ensures that subscriptions are handled between broker and clients via a one hop connection and publications are sent through the broker tree which reduce the possibility that a notification is lost on its way to an interested client. This forms an easy to maintain protocol, where no routing knowledge has to be produced and therefor no Routing Information Protocol (RIP) is needed.

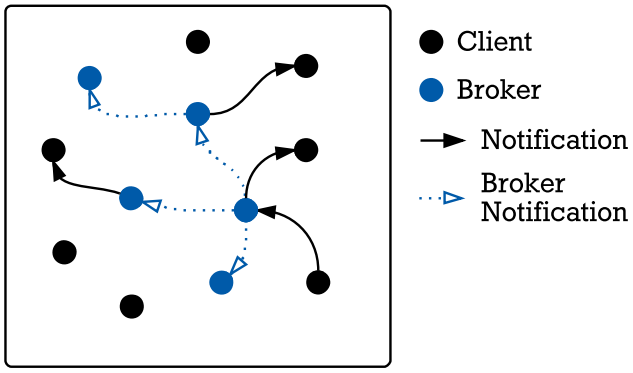


Figure 6.6.: Concept and overlay view of a tree based broker overlay.

Figure 6.6 shows a broker tree consisting of three broker nodes with a bunch of client nodes. The arrows illustrate the path of a publication starting from a client node, through the broker substrate, notifying an interested client node of this publication. If a client node leaves the range of its broker node, a new broker node must be found. This implies that the client node, which is leaving the influence of its broker again subscribes to the topics it is interested in. Nodes have to constantly maintain these operation with an increasing factor depending of the movement speed of the nodes. The conditions for this overlay are set to a slow movement speed and a high publication rate.

6.3.4 Parameters and Calculation of Overlay Conditions

Parameters describe different properties of the examined system and together they form a state, resembling these properties. They are abstractions of values that are measurable in an ad hoc network overlay. Parameters in the transition service are generally used for two different things. First, the transition service determines based on the parameters, which pub/sub overlay is chosen. The second use is by the pub/sub overlays to calculate its internal structure. For example, a density based pub/sub overlay as described in Chapter 3 does use density information to calculate the corresponding hilltop equivalents. The pub/sub overlays also use the parameters to define the conditions on which they are best to used by the transition service. The following parameters are used in the TransOS prototype and are calculated as follows.

Movement Speed Parameter The movement speed parameter describes the velocity of a peer in the network. The velocity is the real movement speed of the device in the scenario. It is measured using the provided location of a node from the simonstrator-API. The distance between two location after a 10 second period is used to calculate the velocity during this interval and therefore is only an approximation of the real movement speed. But since the state of a node is also checked periodically, the calculation of the velocity is sufficient. Besides the simulated environment, for a real device a GPS-module can provide location and velocity information in order to provide movement speed information for the overlay.

Density Parameter The density parameter describes the local density on a node and is calculated passively by monitoring the incoming messages on a node. It is a complete passive calculation of the nodes density. If a message is received on a node the node counts the sender as neighbour node and therefore increases the local density. A timeout specifies the time in which the node again has to receive a message from the neighbour, otherwise the neighbour node is deleted and the density is decreased. Using this passive density parameter, not only the density on specific node can be calculated, but also the average density of a whole area with multiple nodes.

Density Changerate Parameter The changerate of the density is derived from the passive density parameter presented above. The changerate simply describes the variation of the density on a node between a time interval. The density changerate is also a measurement of the uniformity of the ad hoc network. If the node enters an area of lesser density, the density changerate goes negative, since the local density is decreasing. On the other hand, the density changerate is positive if the peer enters a higher density area. So, a constant density changerate other than zero indicates a non-uniformity network while the node travels through the network with a high velocity.

Number of Nodes Parameter The number of nodes participating in the overlay is important to calculate the expected connectivity of the overlay. This parameter is only evaluated on monitoring nodes and depends on the incoming number of answers of status requests. Nodes do not have to register to the transition service. They can start its activity without any preparation.

Number of Subscriptions Parameter The number of subscriptions on a node is an estimation of the workload of one node. It is calculated locally by simply counting the number of subscriptions on the node. It indicates that a higher number of subscriptions does require a higher local bandwidth to deliver notifications to this node. This information can for example be used to determine if a node should be a client or a broker of a pub/sub overlay. Nodes with a very high number of subscriptions are brokers and therefore do not need to subscribe at an other broker, reducing the overhead.

Publication Rate Parameter The publication rate is a similar metric as the number of subscriptions to indicate the workload of the overlay. Other than a simple integer, the publication rate is a value per time interval and is also measured passively by counting the number of publications on a node during a specific time interval.

Time Parameter The time parameter is used to define calendar entries of a time based transition between different pub/sub overlay. With that parameter, the system administrator is able to define a time in which the transition service switched to a specific pub/sub overlay, ignoring all other parameters. For example, during maintenance at night, the system could switch to a more stable overlay or re-initiate another overlay for better control of the system.

Random Parameter The random parameter simply produces random values on a node. This parameter is particularly useful to test the monitoring component of the transition-service if no scenario for the network is defined. The random values can be used to trigger decisions of the pub/sub overlay of decisions of the transition service without tuning of other performance metrics of the network.

7 Evaluation

This chapter covers the evaluation of the transition-enabled pub/sub service TransOS. The evaluation is based on a real world scenario. This scenario takes estimations on parameters of the system based on probability of human behaviour. Known models are used to model this behaviour and keep it in a controlled environment. The evaluation itself is done by means of simulations with the previously presented PeerfactSim.KOM simulator. The deep study of the design and the implemented prototype help to prove the concept of transition-enabled pub/sub services and to understand the impact of transitions and the performance of different pub/sub mechanisms. It also testifies that the design of the system satisfies the defined requirements of a transition-enabled pub/sub system. Simulation is chosen as evaluation technique, since simulations are highly controllable and the measurements are repeatable.

7.1 Evaluation Technique and Goals of the Evaluation

The evaluation itself consists of two separate steps. In the first step, the central broker, distributed broker, and tree-broker overlays, as described in Chapter 6, are evaluated within different scenarios and environmental conditions. The results of these simulations are further used to tune the transition system and set the conditions for the specific pub/sub overlays. As already mentioned, the presented pub/sub overlays are chosen to cover a wide range of different approaches in terms of distribution and protocol complexity. It is expected, that each pub/sub overlay shows a distinct performance on specific scenarios. Thereby, only the performance of the pub/sub systems without interference of the transition system or monitoring operations is determined. Transitions are not triggered within this scenario.

In the second step, the actual transition service is evaluated. The previously evaluated pub/sub mechanisms are used as switchable overlays. The evaluation will show the influence of transitions on the systems performance. Multiple scenarios are specifically chosen to show the functionality of the system and its behaviour during a transition. The goal of the evaluation of TransOS is to show that the concept of a transition-enabled pub/sub service is realisable. It must be shown that transitions at the run-time of the service can exchange the used pub/sub overlay, without loosing data or having a negative impact on the system. Also, transitions must be executed intelligently to react to different states of the network and to show a positive effect on the performance metrics.

The fundamental scenario which all variations are based on is a university campus management system scenario. For a proper definition of the scenario the workload of this application must be set. The management system provides materials to study, like scripts and e-books. People can subscribe to a specific topic they are interested in and once new material is published, the clients are notified. This scenario is chosen, because it covers a huge variety of different real life situations, which come close to an everyday based usage of a MANET. The campus management system scenario basically covers the following situations. During lectures, many mobile devices are grouped together. Teachers constantly produce new material in form of drawings or even live streams of a lecture. A live stream is treated in this scenario as a set of publications, containing small parts of the video and audio as payload. Between lectures, people are moving around which results in a higher velocity of the nodes. They also more likely access the Internet during this time to keep their social Internet life up to date. The Internet is accessed via a single access point of the MANET, so it is expected that the workload on this node increases for this time until the next lecture begins and the motion comes to a rest. During the weekend, no lectures are held and the workload on the system is very small, as well as the number of participating peers is also rather low. The evaluation of TransOS keeps this scenario in mind to show possible benefits of a transition-enabled pub/sub service for such a kind of scenario.

7.2 Parameters to Study

Before the system can be evaluated, the parameters to study are chosen for defining the evaluated scenario. Parameters can be divided into system parameters and workload parameters. System parameters include both, hardware and software parameters. Workload parameters are characteristics of user requests. The parameters are varied in the study, covering a large range of environmental conditions.

Number of Peers The number of peers describes the number of participating peers in the scenario. The definition of the number of peers is important, because it defines the complexity of the topography a pub/sub overlay has to maintain. The number of peers is distinguished between online and offline peers at each time of the simulation. The average number of online peers defines the density of the scenario. The simulation setup ensures that peers come online only once. If a peer is taken down, it will not participate in the scenario anymore. The maximum number of peers in a scenario is defined as $|P|_{\max}$, while the number of online peers is defined as $|P_\star|$.

Scenario Size With the scenario size (S) the area on which the peers can be placed is defined. In combination with the number of online peers at a time, the average density over all nodes can be calculated as $\rho^* = |P|S^{-1}$. In the simulation, only rectangular area is considered and simulated. Also for simplification, no objects besides the peers are placed in the simulated area, which might have an influence on the range of the peers or block wireless communication completely. The peers can also move on the area freely. Edges of the simulated area act as walls. If a node reaches the rim, it bounces of and follows a new way.

Velocity of Nodes The variation of the movement speed of the peers is expected to show a high impact on the performance of individual pub/sub mechanisms. With an increasing velocity of the nodes, the probability of a single-hop link break is increased. This is caused by the limited range of the nodes defined by the network link layer. With increasing velocity, pub/sub overlays which rely on fixed or structured topology have to execute repairing operations with a higher rate. Link breaks are managed by updating routing tables, if the routing algorithm uses these. The overall performance is expected to decrease.

Movement Model The movement model describes the behaviour of the peers in the scenario in terms of velocity and direction. For the evaluation, three different movement models are available, a random movement model, a social movement model, and a gaussian movement model. These models are already presented in Section 6.1. The social movement model has many parameters that model social interactions between users. However, as the evaluation should be detached from the used movement model, and for simplification reasons, the random waypoint model is used to evaluate TransOS. However, the service is also tested with the other movement models to check if the overall system behaviour remains the same.

Churn In terms of pub/sub, churn describes the model of "connectedness" between adjacent nodes. As already mentioned, the time of how long nodes can communicate with each other depends on the velocity of the nodes and the movement model. It also depends on the rate of how the nodes join and leave the simulation. Churn is a combination of these three dimensions.

Simulation Length The length of the simulation defines the amount of data a single simulation run will return. It also defines the magnitude of the simulated scenario. One simulation run is always divided into four parts. In the first phase, the initialization phase, peers join the simulated scenario. The second phase starts after 30 minutes of the simulation. In this phase, nodes start to publish on topics. After 60 minutes of the start of the simulation, performance metrics are measured. 20 minutes before the simulation is over, data measurement is discontinued.

Link Layer Model As described previously, the ISO/OSI link layer of the PeerfactSim.KOM simulator is exchangeable. The wifi 802.11 underlay is mainly chosen, because is is used in the evaluation of the density based pub/sub overlay (Section 3.2.1). This underlay does emulate the real world conditions for a wireless connection. However, some abstractions have been made since the reality can never be fully captured. A second very simple wifi underlay model is available. This unity base-

line model is freely configurable in terms of range of the nodes, as well as upload and download bandwidth.

Workload The workload of the application is determined by the publication rate and the number of subscribers. In terms of pub/sub, workload is mainly dependent on the size of publication. A publication will result in the transmission of a notification. This notification carries the data from the publisher to all interested subscribers. Also depending on the used pub/sub overlay, workload is not distributed evenly across all peers. Increasing the number of publishers (|Pub|) or the delay between publications (Δ_{Pub}) does also increase the workload of the pub/sub service. Increasing the number of subscribers (|Sub|) does increase the number of independent peers that must be notified of a publication. The effects of changing this value depends on the mechanisms used in the pub/sub overlay.

7.3 Performance Metrics and Simulation Setup

Performance metrics are used to measure the performance of the system. These metrics are derived from the requirements of the transition-enabled pub/sub overlay and are used for comparison in order to ultimately be able to make a statement whether the requirements have been met.

Publication Delivery Rate The delivery rate corresponds to the rate of successfully delivered notifications versus the overall expected number of delivered notifications. During the simulation, the application is aware of any peers that subscribed to a specific topic. To reach a delivery rate of 100%, all subscribers of a topic must be informed within a given time interval, which is defined in the requirements (Chapter 4) of the system. Should not all subscriber be reached, the publication delivery rate is correspondingly lower.

Publication Delivery Delay The delivery delay is defined as the time between the creation of a publication and the notification of a subscriber. This time is important to measure for a precise evaluation of the pub/sub overlay performance under time critical applications. The delay of a notification depends on the state of the pub/sub overlay. If the overlay was just initialized, the delivery delay is expected to be higher as the overlay has to setup its internal structure. For the evaluation of the single pub/sub overlay without transitions, measurements are only taken with omitting the startup and ending of the scenario. However, the evaluation of the transition service does capture these phases due to the nature of transition-enabled overlays. The requirement of the transition service is do reduce the delivery delay during transitions as much as possible to allow seamless transitions.

Message Count The traffic on the MANET is measured to evaluate the pub/sub service against available bandwidth. The traffic corresponds to the number of sent messages through the network. It is divided into incoming ($|C_{\leftarrow}|$) and outgoing ($|C_{\rightarrow}|$) traffic. While the average traffic over the whole overlay does give a good metric for the overall efficiency of the overlay, the traffic can be unevenly distributed over the peers. In this case, it is useful to look at the traffic per node ($|C(p)|$).

The simulation setup strongly orients on the capabilities of the PeerfactSim.KOM simulation framework, which is introduced in Section 6.1. PeerfactSim.KOM calculates random values with an internal seed to achieve a deterministic behaviour of the simulation, which is repeatable. This principle is also used in the network simulator (NS-2)[1], which is used in many other studies of MANET overlays. To gain statistical significance, each evaluation run of TransOS and the individual overlays is repeated five times with different seeds for the random generator. For the results, a 95% confidence interval over these results is shown. The details of the parameter setup used in the evaluation is summarized in Table 7.1. A distinction is made between variable and constant parameters. For these parameters, different workload models are developed in the following section.

[1] http://www.isi.edu/nsnam/ns/ [Accessed 28/11/13]

Name	Symbol	Unit
	variable	
Simulation Size[1]	$S = w \times h$	m^2
Total number of distinct peers	$\|P\|$	none
Churn Rate	$\Lambda(t)$	s^{-1}
Velocity	$v \in [v_{min}, v_{max}]$	ms^{-1}
Movement Model	$M \in \{\text{Random}, \text{Social}, \text{Gauss}\}$	none
Transition Strategy	$\tau \in \{\text{Simple}, \text{Hyst}, \text{Adapt}\}$	none
Delay between publications	Δ_{Pub}	s
Number of Subscriber	$\|\text{Sub}\|$	none
Number of Pub/Sub[2]	$\text{Pub} \cap \text{Sub}$	none
	constant	
Simulation Length	$T_s = 120\,\text{min}$	min
Notification Size	$\eta = 45\,\text{Byte}$	Byte
Number of Publisher	$\|\text{Pub}\| = 10$	none
Transmission range	$r = 220\,\text{m}$	m
Sampling Interval	$\Delta_I = 1\,\text{min}$	min
Movement Interval	$\Delta_M = 5\,\text{s}$	s
Link Layer Model	$\lambda = 802.11$	none
Access Link Bandwidth (outgoing)[3]	$\hat{R}_\uparrow = 8\,\text{Mbit/s}$	MBit/s
Access Link Bandwidth (ingoing)[3]	$\hat{R}_\downarrow = 8\,\text{Mbit/s}$	MBit/s
Number of Topics	$\|\Theta\| = 1$	none

[1] Width (w) × height (h).
[2] Number of nodes that are both, publisher and subscriber.
[3] Only on unity baseline link layer.

Table 7.1.: Overview over variable and constant parameters used in the simulation setup.

7.4 Model Development and Workload Selection

In order to perform an accurate analysis, the development of the workload model orients on real life scenarios and other works which have evaluated pub/sub systems. [KCC05] has surveyed MANET simulation studies from proceedings between 2000 and 2005. The workload selection for TransOS is based on the cumulative results of this survey. The goal is to produce results, that are not specific to the scenario used in the experiment, while the conditions used to evaluate the system do consider the aspects of MANETs. In the survey, 61 simulation studies are listed. In the presented simulations, the number of peers are varied between 10 and 30000 peers. The simulated area varies between 25 m × 25 m and 5000 m × 5000 m. In terms of pub/sub overlays, it is important to achieve an always connected overlay. Because of that, the neighbour count, respectively the neighbourhood density of a node is a important parameter. The neighbourhood density of a node based on the transmission range and the simulation area can be calculated as $\rho = \pi r^2 |P| S^{-1}$. This does not account for the edge of the simulation area.

The transmission range of the studies does vary between 3 m and 1061 m. For the evaluation of TransOS, a transmission range of 220 m is assumed. This is the average transmission range of the wifi

802.11 link layer used in PeerfactSim.KOM. The transmission range of the unity baseline link layer is set accordingly. For the calculation of the node density of each study, the original listed transmission range is used for the calculation. The arithmetic mean neighbour count over all 61 studies is $\tilde{d} \approx 13.59$. Now, every combination of the simulation size and the number of nodes can be chosen, without losing connectedness of the MANET.

Workload A – Scenario

The development of the workload assumes the following prerequisites. The scenario size must be big enough to reach a certain hop count for massages traveling through the network. This is important to evaluate the routing and the behaviour of the pub/sub overlays under the estimation of forwarded publications and subscriptions. The minimum hop count to reach one corner of the simulation area to the opposite corner depends on the footprint of the nodes and the network diameter. As already mentioned, a circular and even node footprint with a radius of $r = 220\,\text{m}$ is assumed. The minimum number of hops that a message can take along the maximum path from the source to destination can be calculated as $\text{hop}_{\text{min}} = (w^2 + h^2)^{0.5} r^{-1}$. If a minimum hop count of 2 is requested, the minimum scenario size is $S \approx 311.13\,\text{m} \times 311.13\,\text{m}$ with the given transmission range. However, under normal workload a minimum hop count of 2 does result in many one hop connections in the middle of the simulated area, which should be avoided.

The simulation studies presented in [KCC05] use an average simulation size of $1003.6\,\text{m} \times 900.3\,\text{m}$. For the evaluation of TransOS and the pub/sub overlays, a square area of $2000\,\text{m} \times 2000\,\text{m}$ is used. This results in a minimum hop count of 12.85 hops. Realizing this, the minimum time to live (TTL) of a message must be set to at least 13. Some simulation runs are also done with a quadrupled area of $4000\,\text{m} \times 4000\,\text{m}$. Due to this reason, the TTL for a message is set to 50 for all simulations. $|P| = 400$ is used as default number of nodes. Together with the default simulation area a neighbour count of $\rho \approx 15.21$, or with the larger area a neighbour count of $\rho \approx 3.80$ is calculated.

A rather high density for the default evaluation is chosen. This is done due to the compensation of decreasing connectivity probability with the increasing number of hops [AJP08]. It also comes very close to the calculated average density of the investigated simulation studies. A low neighbour count is used to show the behaviour of the service under a less denser scenario.

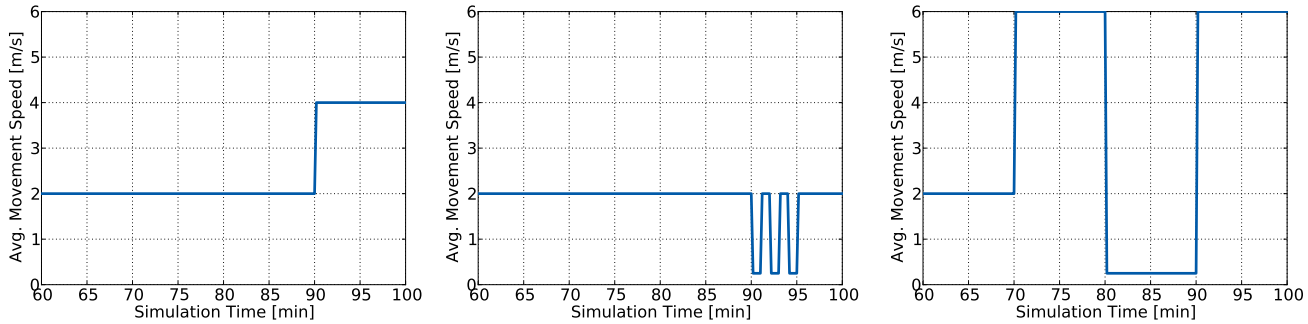
The transition-enabled pub/sub service supports the join and leave of nodes during run-time. However, adding nodes or removing nodes during the simulation is not trivial. Due to restrictions of the simulation environment, this cannot be done instantaneously. Nodes have to join one by each other and the time it takes until they participate is not very well predictable. Also, changing the quantity of the nodes has always an influence on other parameters of the simulation like the ratio of publisher and normal nodes. So, by joining nodes during the simulation, the number of publisher and subscriber must also be increased accordingly. This again will result in a higher publication ratio since more publisher nodes are available. Also, removing nodes may produce unpredictable results. For example, removing the broker in the single broker overlay does change the results of the evaluation significantly more than removing nodes which are neither broker, publisher or subscriber. Due to these reasons, the evaluation of the system under variation of the number of nodes during run-time is excluded. Nevertheless it may be interesting for the future work on the development of TransOS.

Another important factor is the initialization of the scenario. As already mentioned in the scenario length parameter definition, the simulation starts with empty caches, queues, and tables. The system fills the caches, queues, and tables until a steady-state of activity is reached. As described in [Scr82], evaluation results generated prior reaching steady-state of the system cannot be used for analysis. For TransOS a default simulation length of 120 min is used, with the measurement of the performance between the 60^{th} minute and the 100^{th} minute of the evaluation.

Workload B – Mobility Model

As second important evaluation scenario, the mobility model of the nodes is analysed. Mostly the random waypoint movement model is used for the evaluation. For this model, we define the default velocity as $v = 2\,\mathrm{ms}^{-1}$ with a variation of $\pm 0.5\,\mathrm{ms}^{-1}$. This rather slow movement speed is a result of the campus management system scenario, in which the nodes travel with the speed of a normal walking person. A car-to-car communication scenario would assume a much higher velocity of the nodes. However, this of course only accounts for the evaluation of the distinct pub/sub overlays.

For the evaluation of TransOS the average movement speed of the nodes is used to trigger a transition between the pub/sub overlays. For that, the movement speed is varied over the whole available value space. In Figure 7.1, three variations of the movement speed scenario are presented which are used to show the effects of different transition strategies in TransOS. For example, a rapid oscillation of the movement speed is used to investigate the effect of an unstable transition parameter. It should be noted, that using the random waypoint model, nodes first finish their movement towards a waypoint before the speed of the node is changed.



(a) Velocity Variation (Small) **(b)** Rapid Velocity Variation **(c)** Velocity Variation (Huge)

Figure 7.1.: Workload definition for the mobility model with three different scenarios.

Workload C – Throughput

The last important definition of workload is the amount of handled events on the pub/sub service. This is mainly determined by the number of publishers and subscribers, as well as the delay between notifications and the notification size. The delay between publications counts per publisher. To calculate the average publication rate on the pub/sub service, the publication rate must be multiplied by the number of publishers. The number of publishers is held constant at $|\mathrm{Pub}| = 10$ over all variations of the scenario and is also constant for different amounts of nodes. The evaluation of TransOS however does include variations of the delay between publications. Figure 7.2 shows a workload scenario with a variable delay between publications on which the transition of a pub/sub overlay is triggered to.

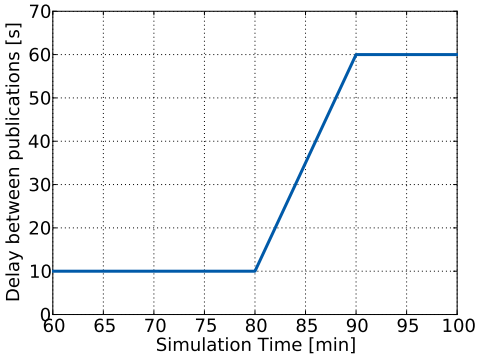


Figure 7.2.: Workload definition by variation of the delay between publications.

In Table 7.2, the variable evaluation parameters for evaluation of the pub/sub overlays are listed. Only one parameter is varied at a time while remaining parameters are fixed to their default values. Due to space constraints, not all evaluation results are presented in this chapter. Additional results are presented in Appendix A. The following plotts of traffic shows the overall traffic on the link layer which includes also overlay unspecific traffic like handshakes of the 802.11 link layer.

Parameter	Values
S	$\underline{2000\,\text{m} \times 2000\,\text{m}}$, $4000\,\text{m} \times 4000\,\text{m}$
$\|P\|$	25, 50, 100, 200, $\underline{400}$, 800, 1600
v	$0\,\text{ms}^{-1}$, $1\,\text{ms}^{-1}$, $\underline{2\,\text{ms}^{-1}}$, $3\,\text{ms}^{-1}$, $4\,\text{ms}^{-1}$
M	$\underline{\text{Random}}$, Social, Gauss
Δ_{Pub}	20 s, $\underline{10\,\text{s}}$, 5 s, 2.5 s, 1.25 s
|Sub|	10, $\underline{20}$, 40, 80

Table 7.2.: Parameter sets for the pub/sub overlay evaluation. Only one parameter is varied at a time. Default values are underlined.

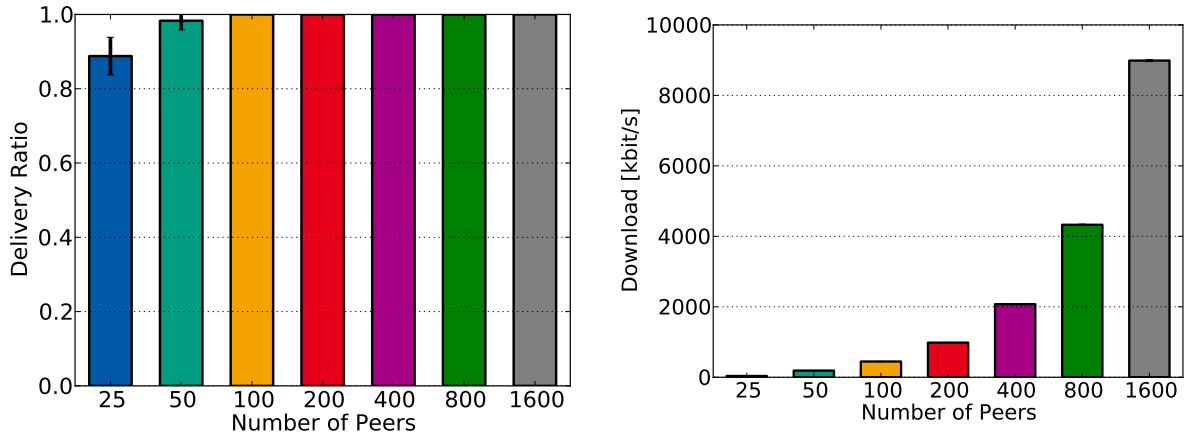
7.5 Overlay Evaluation

Before the actual transition service can be evaluated, the performance of the pub/sub overlays which are going to be used within the system must be assessed. These measured performance metrics are then used to define the conditions for each pub/sub overlay. The conditions define a range of states on which an overlay is preferred and used by the transition overlay. They thereby define when a transition should occur.

7.5.1 Distributed Broker Overlay

For the distributed broker pub/sub overlay, it is expected that the performance is steady for environmental changes of the participating peers, like the variation of the velocity of the peers. The reason for that is the rather simple mechanism the overlay. No distinction between clients and brokers is made and consequently changes in the topological disposal of the peers should not effect the performance of the overlay. However, due to its simplicity, it is expected that with increasing workload, the distributed broker overlay experiences a rather harsh increase of traffic. This is caused by the flooding based distribution method and routing protocol of the overlay. A notification is sent via multicast to every peer of the network. While this makes the overlay very robust, it has a negative impact on the scalability of this overlay performs well below average. For the beginning, a look at the behaviour of the distributed broker overlay under the variations of the number of peers is taken (workload A).

The evaluation shows the typical behaviour of a flooding based pub/sub protocol. Figure 7.3a shows a delivery ratio of 100% with a peer count of $|P| > 50$, meaning that every subscriber of a topic got notified if a publication on this topic occurred. Only at the lower end of the number of peers, the delivery ratio starts to drop due to the formation of disconnected clusters. This is easily understandable since the average neighbour count of 25 peers in an area of $4\,\text{km}^2$ is only 0.95. If a look is taken at the distribution of the delivery ratio over the number of peers (see Figure A.1a), the clustering can start to be seen with 50 peers that approximately 9 peers did not receive 10% of the overall publications, while at 25 peers approximately 12 peers did not receive all publications.

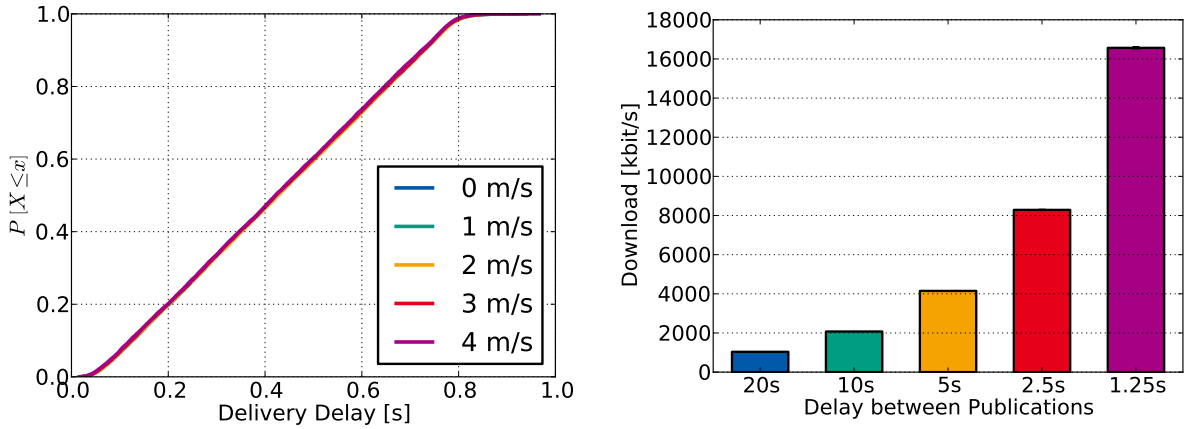


(a) Delivery Ratio vs. #Peers

(b) Summarized Traffic vs. #Peers

Figure 7.3.: Delivery ratio (a) and summarized traffic size (b) under the variation of the number of peers shown for the distributed broker overlay.

The drawback of the good delivery ratio is the increase in traffic. It is expected that a flooding based overlay produces an exponential increase of traffic with an increasing number of peers. As it can been seen in Figure 7.3b, the increase of traffic is linearly. This is a result of the random early discard of broadcasts, which is an implemented mechanism on the distributed broker overlay. Also, the traffic is evenly distributed across all peers. In the appendix the distribution of the traffic is shown (Figure A.1b). It shows a sigmoidal graph for all scenario sizes which in is in terms of pub/sub a even distribution. A very small effect is included in here due to peers in the middle of the simulation area forward messages more likely than peers at the rim of the simulation, so uploading traffic is smaller here.



(a) Delivery Delay Distribution vs. Velocity

(b) Summarized Traffic vs. Publication Delay

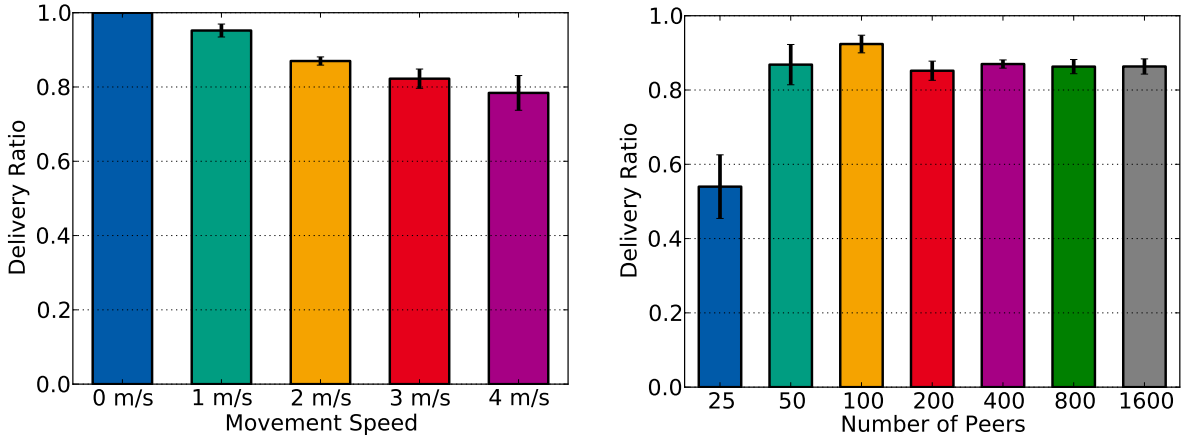
Figure 7.4.: Delivery delay distribution with various peer velocities (a) and summarized traffic with variation of the delay between publications (b) shown for the distributed broker overlay.

As expected and seen in Figure 7.4a, the variation of the average velocity of the nodes does not have any impact on the functionality of the overlay. The delivery delay stays in a range of 1 s for a few nodes and 0.1 s independent of the velocity of the peers. If the workload on the overlay is increased in form of a higher publication rate, the amount of traffic on the network link layer increases correspondingly. In Figure 7.4b a linear corresponding increase of traffic on the network link layer can be observed with a higher publication rate. Figure A.1 in the appendix shows that an increase of the number of subscriber does not have any effect on the traffic on the network as expected. This is caused due to the reason that

subscriber of a topic are not notified independently. A publication on a topic does cause that every node is informed and peers perform a local merge of their subscriptions against the notification which has no effect on the traffic at all. Also seen in the appendix, the bar graph of the traffic shows the summarized traffic over all peers, so not every peer receives data with 16 MBit/s.

7.5.2 Single Broker Overlay

The single broker pub/sub overlay embodies the stylistic principles of the pub/sub communication paradigm. Publications are merged against subscriptions on broker nodes. Nodes subscribe to a topic on broker nodes and the clients must be notified on a corresponding publication. The routing of this overlay is artificially optimized by using the global knowledge routing capabilities of the simonstrator-API. The first interesting observation for the single broker overlay can be seen in Figure 7.5a. The single broker overlay does only achieve a 100% delivery ratio of publications during a steady-state scenario. With increasing velocity of the nodes, the delivery ratio drops with roughly 5% per $1\,\mathrm{ms}^{-1}$. This is caused by the functional principle of the global knowledge routing algorithm. The routing information between nodes is cached by the simulator. Once a message is sent, the receiving node or intermediate nodes may have already changed their position in the routing topology. If a node cannot longer be reached over the changed routing path, the message cannot be delivered. Since the single broker overlay has no mechanism for compensation of message loss, the notification on that specific publication is also lost. A routing path error will lead to a new calculation of the routing information, but for demonstration purposes of this effect the notification of the subscriber is not resent. Since the default velocity of the nodes is set to $2\,\mathrm{ms}^{-1}$, this effect can also be observed on the variation of the number of peers. Also the traffic around the single broker is very high, which does result in further message drops.



(a) Delivery Ratio vs. Velocity

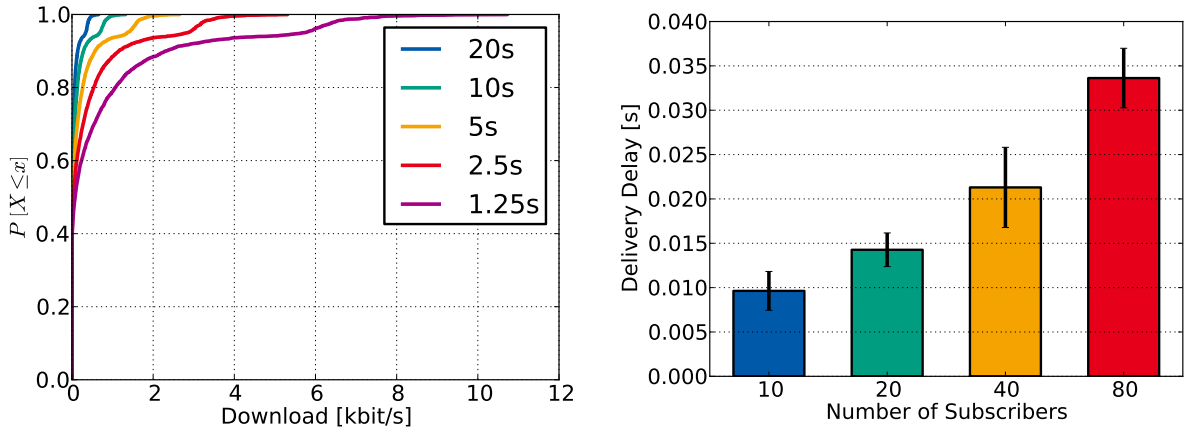
(b) Deliver Ratio vs. #Peers

Figure 7.5.: Delivery ratio against velocity of the peers (a) and delivery ratio against the number of peers (b) shown for the single broker overlay.

Figure 7.5b shows that the number of nodes, respectively the node density has no influence on the delivery ratio under a certain movement speed. This shows that the calculation of the dijkstra route does not include a recursive fallback to route the message over another path. Normally, it would be expected that the delivery ratio increases with density, since more possible routing paths are available. In the appendix (Figure A.2) the dependency of this effect and the size of the world is shown. With a larger world size this effect is emphasized.

Also moved to the appendix is the summarized traffic of the nodes. The only thing that can be observed here is that the amount of traffic does not change with the overall number of nodes, except at the very lower end where clustering occurs. By the virtue of the confidence intervals no trend can be detected.

The most dramatic difference between the single broker overlay and the distributed broker overlay can be seen in the distribution of the traffic over the nodes. Since a single node is responsible for all broker activities, the traffic on this node is directly depending on the publication rate and the number of subscribers that must be informed. If only one publication is assumed, the outgoing traffic on the single broker node is determined by the number of subscribers and the notification size. The ratio is exactly $1 : [|\text{Sub}| \times \eta]$. Figure 7.6a shows the unevenly distributed amount of traffic over all nodes. An increasing summarized amount of traffic can be observed for a relatively small number of nodes. The traffic for a node is depending on the propinquity of a node to the single broker node, as these nodes have to forward more traffic for the overlay. A very small amount of peers (the single broker) does handle a very large amount of traffic, which can be seen on the top end of the shown distribution function. While the delivery ratio is just like the distributed broker overlay independent from the number of subscribers, the average delivery delay increases with the number of subscribers on the single broker overlay. In Figure 7.6b, this effect can be nicely observed. The reason for that is that the single broker has to notify each subscriber on a publication one after the other. All these messages cannot be sent simultaneously, which results in an increasing average delivery delay.



(a) Traffic Distribution vs. Publication Delay

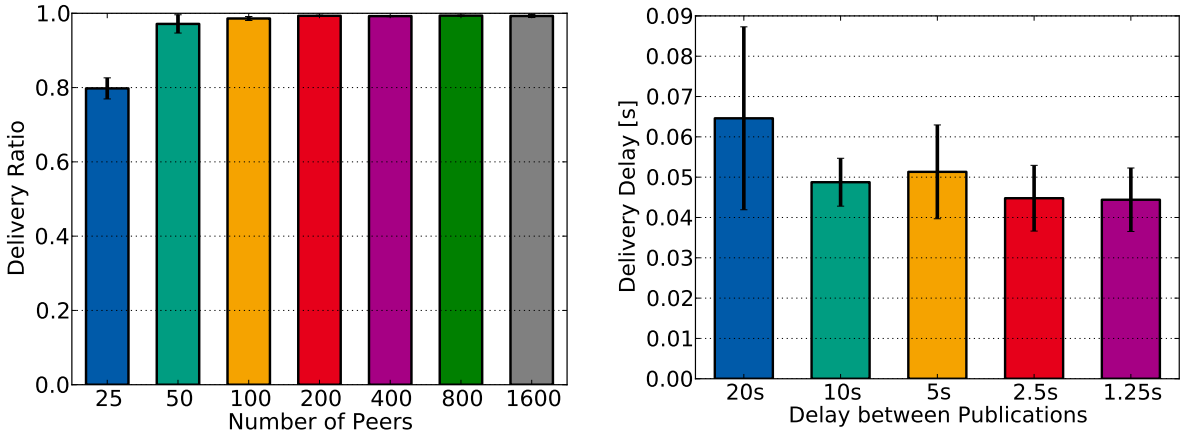
(b) Average Delivery Delay vs. #Subscriber

Figure 7.6.: Traffic distribution depending on publication delay (a) and delivery delay against the number of subscriber (b) shown for the single broker overlay.

7.5.3 Tree Broker Overlay

The most interesting overlay to evaluate is the tree broker overlay. This pub/sub overlay has a complex internal structure with active recovering capabilities to maintain a broker substrate carrying notifications through the network. The evaluation of the broker tree shows that the tree performs very well under different conditions. The delivery ratio is constant at around 98%. Clustering appears similar to the distributed broker overlay at the lower end of the number of nodes (see Figure 7.8b). One very interesting effect can be seen in Figure 7.7b. The average delivery delay decreases with more workload, respectively with decreasing delay between publications. The threshold value appears to be between a publication delay of 20 s and 10 s but due to the relatively large confidence intervals, no guaranteed proposition can be made. One explanation for this might be the following. With the decreasing delay between publications, broker neighbour information is updated more often. If a broker cannot be reached by its client, a new broker is chosen within a timeout delay of 2 s. Under normal circumstances, a loss of a connection to a broker is detected by a neighbour timeout (see Table 6.1). The normal hello-beacon cycle time is set to 10 s. Thus, if the delay between publications is faster than the 10 s hello cycle and the 20 s broker neighbour timeout, the tree overlay can be repaired faster than with the default fallback strategy. Also,

the absolute values in the bar chart are very small (0.065 s versus 0.045 s) so at least it can be declared that the decreasing delay between publications does not have a negative effect on the overlay.

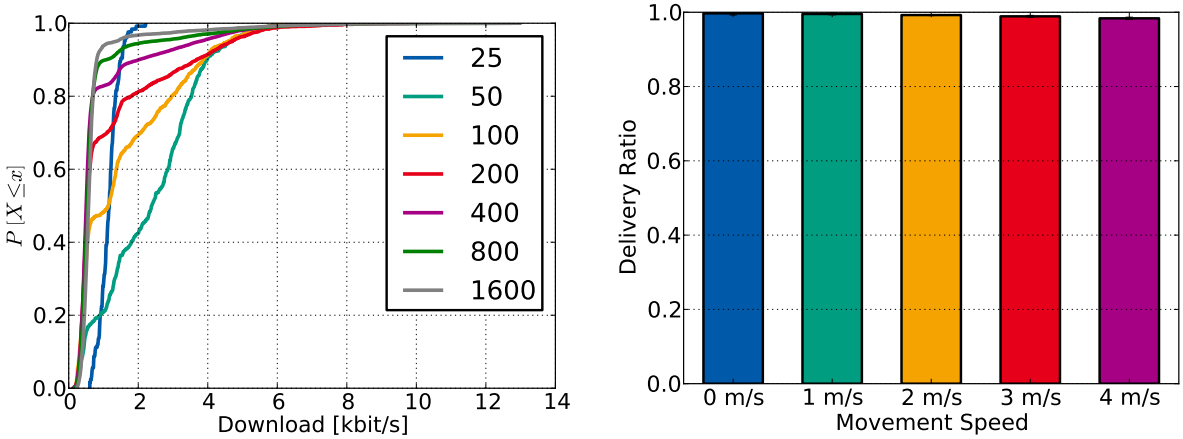


(a) Delivery Ratio vs. #Peers

(b) Delivery Delay vs. Publication Delay

Figure 7.7.: Delivery ratio depending on the number of peers (a) and average delivery delay depending on different publication delays (b) shown for the tree broker overlay.

The traffic distribution of the tree broker overlay is similar to the single broker overlay, since brokers have a higher occurrence of traffic than client nodes. A small difference which can be seen in the distribution can be observed in Figure 7.8a. Notifications are forwarded over the broker tree substrate resulting in a higher traffic for these broker nodes. Taking readings from the distribution function shows that the number of brokers is inversely proportional to the number of nodes. For 800 nodes, the amount of brokers is only at around 10%. For 50 nodes however, the amount of brokers is around 80%. This is caused by the reason that independently of the number of nodes, the whole area has to be covered by the broker substrate. If less nodes are available to fill the area, more of the nodes have to become a broker. Even more evidence of this effect can be seen if only the outgoing traffic is plotted (Figure A.3a). The first small step in the distribution function are the brokers which do forward the notifications, the second larger step are all publishers constantly sending publications.



(a) Ingoing Traffic Distribution vs. #Peers

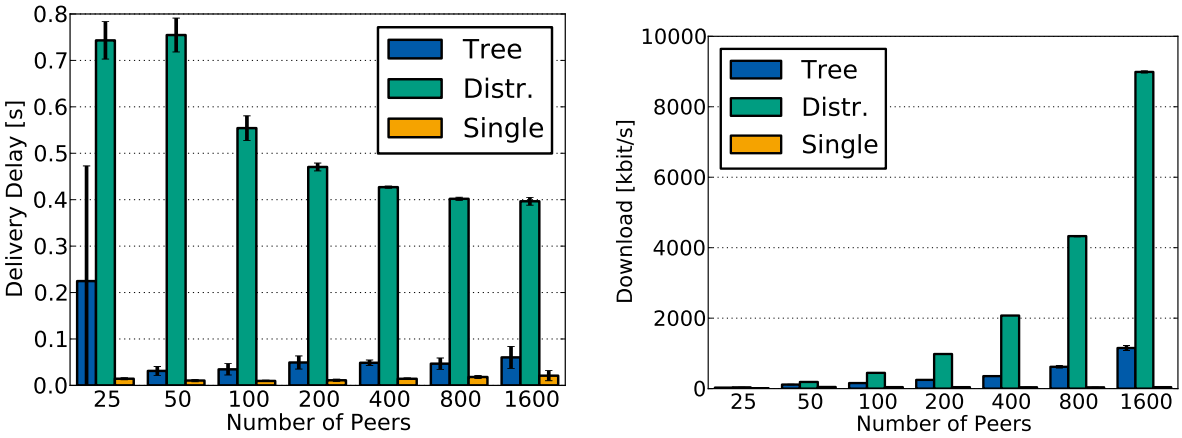
(b) Delivery Ratio vs. Velocity

Figure 7.8.: Distribution of ingoing traffic depending on the number of peers (a) and delivery ratio depending on the movement speed of the peers (b) shown for the tree broker overlay.

If a look at the overlay is taken under the variation of the velocity of the nodes it can be seen that the delivery ratio is not affected by the velocity of the node, so even with a high churnrate the tree overlay can keep its activity alive. Figure 7.8b shows that the delivery ratio just barely starts to drop at $4\,\mathrm{ms}^{-1}$. But since no high velocity scenario is assumed for the evaluation of the pub/sub system, this isn't a problem for the outcome of the evaluation. At last a look can be taken on the delivery ratio with different world sizes, see Figure A.3b. Both runs are made with default amount of nodes $|P| = 400$. It can be seen, that even with a large world which needs many hops to pass through, the delivery ratio of publications is not effected. This shows that the broker tree substrate is connected most of the time.

7.5.4 Direct Comparison of the Different Overlays

If the different pub/sub overlays are compared directly, the following interesting observations can be made. At first, a comparison between the overlays in relation to the number of nodes is presented.



(a) Average Delivery Delay vs. #Peers

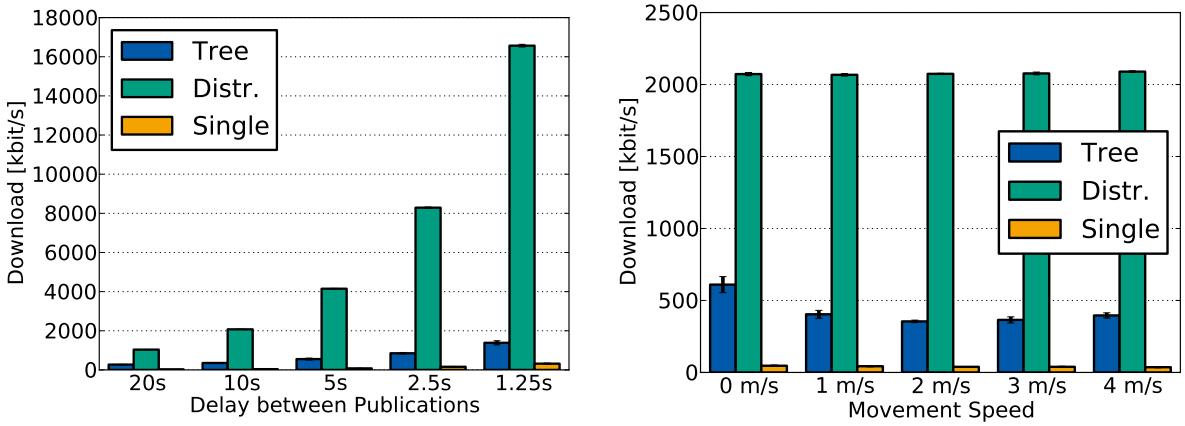
(b) Summarized Traffic vs. #Peers

Figure 7.9.: Comparative delivery delay (a) and summarized traffic (b) under the variation of the number of peers.

The delivery delay, shown in Figure 7.9a, shows a huge gap between the distributed broker overlay and the other overlays. While the distributed broker overlay levels out at a delay of 0.4 seconds, the tree broker overlay still performs four times better. The distributed broker overlay does have a higher delivery delay than the other overlays because it uses multicasts for communication. Broadcasts are randomly delayed before they are forwarded by a node to avoid collisions with other messages. With an increasing number of peers, the delivery delay decreases for the distributed broker overlay, because the flooding based notification distribution has a potential higher probability to reach a peer at the edge of the transmission range of a node and therefore use less hops to path through the network. If the pure summarized traffic of the overlays is compared, see Figure 7.9b, the distributed broker overlay produces much more traffic than the tree or the single broker overlay. The single broker overlay produces the least amount of traffic since it resembles a synthetic ideal pub/sub overlay with an optimal routing protocol. The tree broker overlay is much better in the consumption of bandwidth than the distributed broker overlay. However, as already discussed, the distributed broker overlay has the best delivery ratio of all overlays independent of the number of nodes. A comparison figure is presented in the appendix (Figure A.4). Good delivery ratio comes at high costs in terms of traffic. All three overlays show no relation in the performance with the change of the number of subscribers. Comparative plots are moved in the appendix. Only on the summarized uploading traffic, a slight increase can be observed for the single broker overlay. However, on uploading traffic the confidence intervals are very high since the uploading traffic strongly depends on the arrangement of the nodes, the actual routing, and subscriber distribution. On 10 subscribers, the uploading traffic of the single broker overlay is identical to the

uploading traffic of the distributed overlay. Such points, where one overlay starts to perform better than another overlay can be used to trigger the transition between these overlays in the transition-enabled pub/sub service.

If the overlays are compared under different publication rates, see Figure 7.10a, the distributed broker overlay again shows an exponential increase in traffic which is much higher than the traffic on the tree broker overlay. For the variation of the velocity of the nodes however, no significant change of the traffic can be observed on any overlay.



(a) Summarized Traffic vs. Publication Ratio

(b) Summarized Traffic vs. Velocity

Figure 7.10.: Comparative traffic under variation of publication ratio (a) and velocity of the peers (b).

In the appendix, further comparisons of the different overlays in relation to the world size and the movement model are presented. With the change of the movement model, no critical problems or changes in the performance can be observed. The use of the social movement model for example, which produces groups of nodes, does not have an influence on the tree broker overlay in terms of the delivery ratio. It would be expected the the groups of nodes, respectively areas with a high spots of density have an influence on the formation of the broker substrate. Only a change of the delivery delay on the tree broker overlay depending on the movement model can be observed. But since the reasons for that is correlated to many influencing factors, a valuing statement on this cannot be made.

The comparison of the different pub/sub overlays has shown that basically one overlay always performs best under the variation of one parameter. Regardless of the parameter, no distinct switching point could be observed on which one overlay performs better than another overlay. The good delivery ratio of the distributed broker overlay comes with a huge drawback on traffic. The problem that arises from the evaluation results of the different overlays is, that no condition for an overlay can be set to one specific level of a parameter. The performance of an overlay must always be valued in consideration of all performance metrics. Due to this reason, the evaluation of TransOS uses synthetical conditions for the different overlays which trigger on a specific level of a parameter to execute the transition, even if that means that the service switches to an overlay which performs slightly worse than the original overlay. The effects of different transition strategies and the effects of different monitoring components are evaluated.

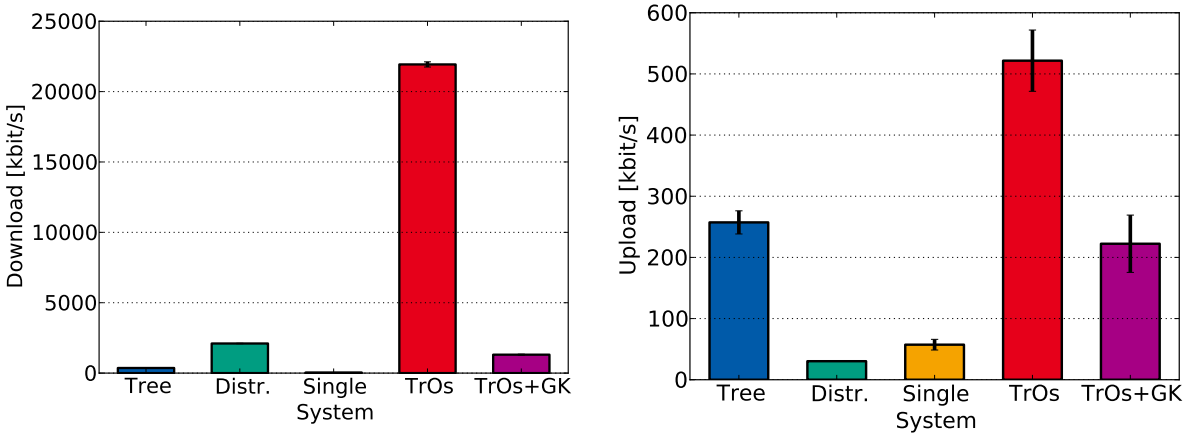
7.6 Transition Evaluation

The second part of the evaluation investigates the performance of the transition-enabled pub/sub overlay itself. The previously presented evaluation results helped to define transitions based on the constructed performance metrics. Two exceptions are made in the evaluation of TransOS. First, and as already mentioned, the evaluation of the distinct pub/sub overlays has shown, that the performance of an overlay

can only sufficient be measured under the inclusion of all performance metrics. Due to this reason, synthetical transition conditions are defined to trigger a transition on a specific value of an overlay performance metric. Second, the single broker overlay is excluded from the transition service due to its global knowledge routing protocol. The unicast routing protocol cannot be changed at run-time on the simulator. Since only the single broker overlay uses global knowledge routing, it must be excluded from the system. As a consequence, transitions can only occur between the distributed broker and tree-based overlay

7.6.1 Proof of Concept Evaluation

To prove the concept of the system, a special condition for triggering the transition is used. Transitions are executed based on a calendar entry similar to the time parameter mentioned in Section 6.3.4. The timings for the switches are defined as follows. Timing 1 starts with the tree overlay and switches at minute 70 to the distributed broker overlay. At minute 90 the overlay is switched back to the transition overlay. Timing 2 also starts with the tree overlay. From minute 70 until minute 80 the distributed broker overlay is actively chosen. From there on the overlay is switched every minute until the end of the simulation.



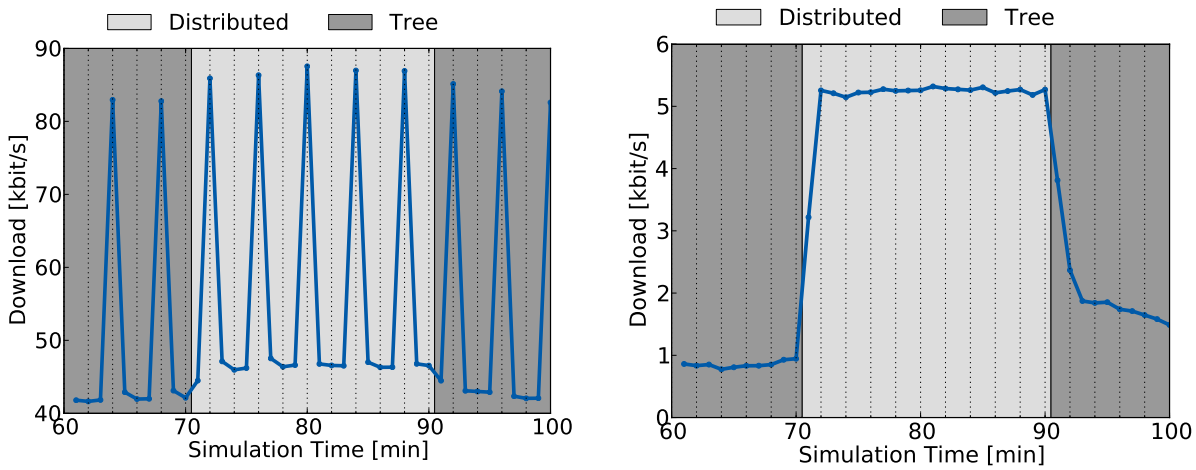
(a) Incoming Summarized Traffic

(b) Outgoing Summarized Traffic

Figure 7.11.: Summarized traffic for the transition-enabled pub/sub service in comparison with the overlays.

Figure 7.11 shows the summarized traffic, download and upload. What immediately can be seen is that the service itself uses much more traffic than the pub/sub overlays by itself. This additional traffic is caused by the primitive monitoring approach. If global knowledge monitoring is used, which does not produce any additional messages, the traffic used is in between the tree based overlay and the distributed broker overlay. This is obvious, since transitions are executed between these overlays. The distribution of the traffic is moved in the appendix. It can be seen that at least the relations are maintained, so the simple monitoring does no exceed the traffic of one node predominantly.

In Figure 7.12a, the transitions over time, as well as the traffic over time is shown. The spikes in traffic are caused by the periodical monitoring, but it can also already be seen that the average traffic increases when the system switches to the distributed broker overlay. Figure 7.12b shows the same simulation run with global knowledge monitoring. The overall traffic here is not dominated by the monitoring, after switching from the tree to the distributed overlay, traffic increases by a factor of five. A small delay can be observed between the above mentioned timings and the transition. This is caused by the monitoring interval since the time parameter does not trigger the transition directly. Instead, the timing parameter is observed by the monitoring mechanism and if a change is detected, the transition is executed.



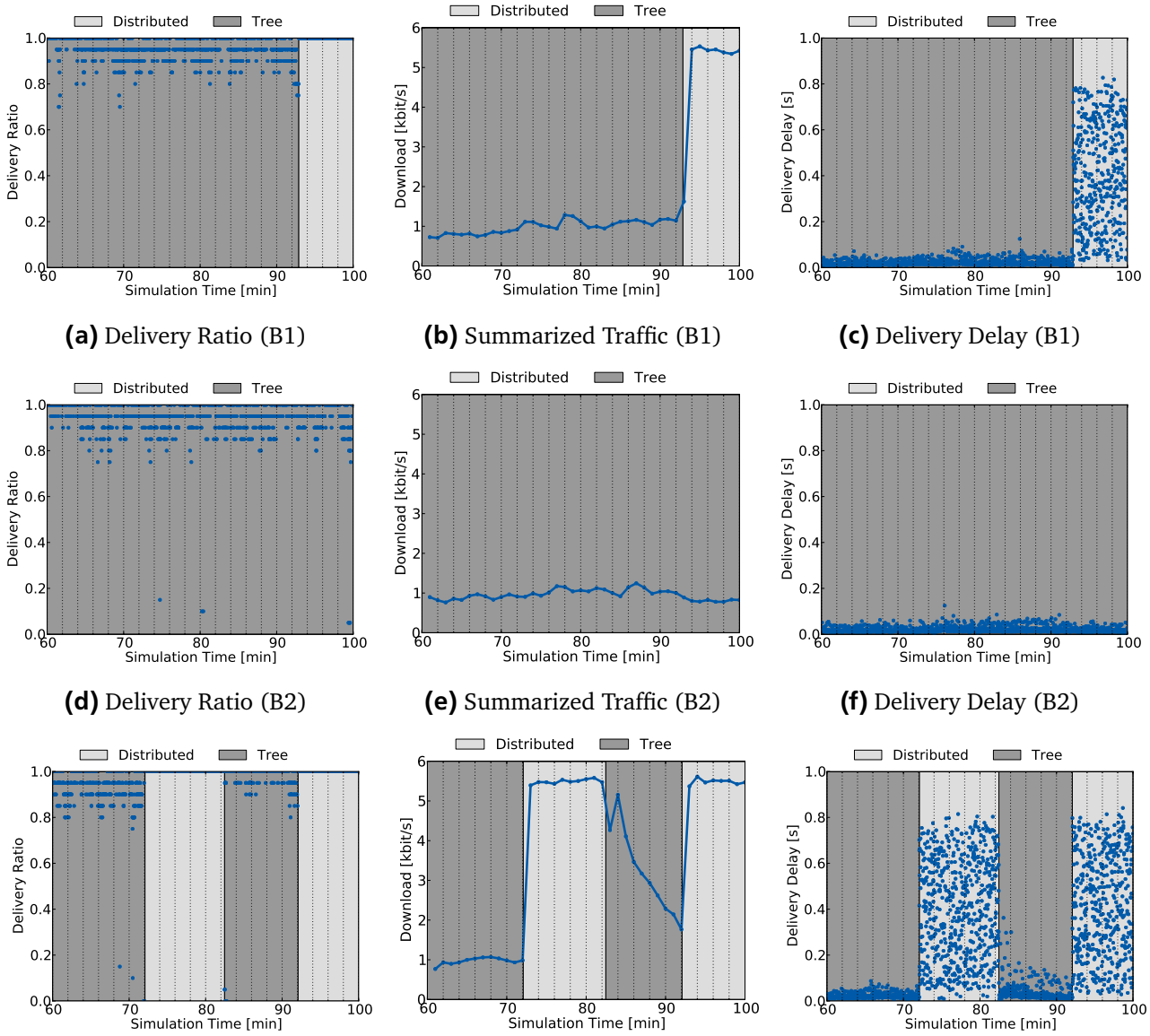
(a) Simple Monitoring

(b) Global Knowledge Monitoring

Figure 7.12.: Transition-enabled system over time with simple monitoring and global knowledge monitoring in a time parameter scenario.

The scenarios for triggering based on the velocity and delay between publications were already presented (workload B/C). Figure 7.13 shows the performance of TRANSOS when triggering based on the measured node velocity. Different performance metrics are presented under the defined movement speed models. The effects of the individual pub/sub mechanisms can nicely be seen. The second speed model definition (workload B2), the rapid oscillation of the movement speed shows that the system did not change the overlay, even if the simple transition strategy is used. This is caused by the internal calculation of the movement speed parameter. Since the velocity of a node cannot be received from the simonstrator-API directly, the nodes extrapolate it from their position change in the scenario. This includes smoothing of the values, therefore the rapid change of velocity cannot be detected by the transition overlay. The discretisation of the delivery ratio, as observed in Figure 7.13a is caused by the limited amount of subscribers. So if there are 20 subscribers in total and one subscriber is not notified, the delivery ratio drops by 0.05.

As presented, during workload C, the publication delay is changed continuously between minute 80 and minute 90 of the simulation from a delay of 10 s to a delay of 60 s. The publication delay condition is set to an average measured value of 15 s, so it expected that a transition is executed around minute 80 and minute 85 of the simulation. As can be observed in Figure 7.14, the transition is triggered at minute 81.5. At this time, the publication ratio is still at around 4 publications per minute and publisher, resulting in more traffic at the distributed broker overlay. However, as the publication rate further decreases, the traffic also decreases. Finally it is in a range like the tree overlay on a high publication rate. For the cost of increased traffic when switched to the distributed overlay and a little higher delivery delay, 100% delivery rate is achieved.

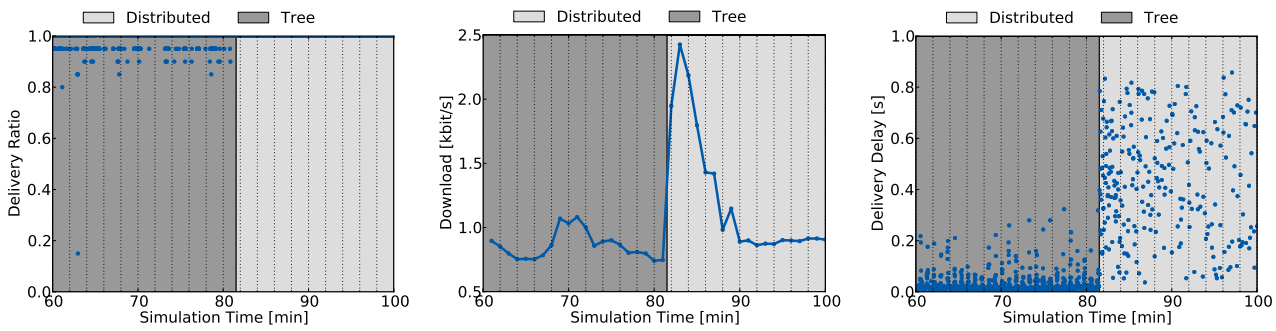


(a) Delivery Ratio (B1) **(b)** Summarized Traffic (B1) **(c)** Delivery Delay (B1)

(d) Delivery Ratio (B2) **(e)** Summarized Traffic (B2) **(f)** Delivery Delay (B2)

(g) Delivery Ratio (B3) **(h)** Summarized Traffic (B3) **(i)** Delivery Delay (B3)

Figure 7.13.: Transition-enabled service triggering on velocity of the peers. (a – c) shows the delivery ratio, the summarized traffic, and the delivery delay under workload B1. (d – f) presents these performance metrics under workload B2, and (g – i) shows it for workload B3.

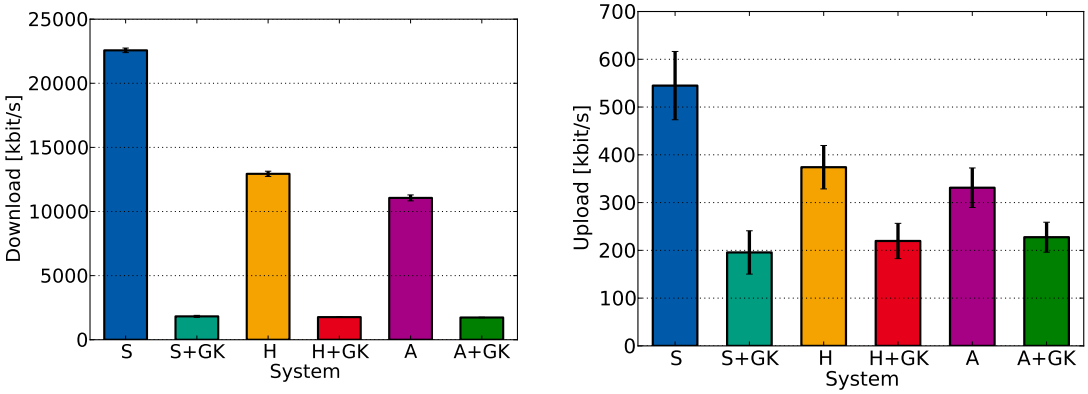


(a) Delivery Ratio (C) **(b)** Summarized Traffic (C) **(c)** Delivery Delay (C)

Figure 7.14.: Transition-enabled service triggering on delay between publications (workload C) with the delivery ratio (a), summarized traffic (b), and delivery delay (c) over time.

7.6.2 Transition Strategy Evaluation

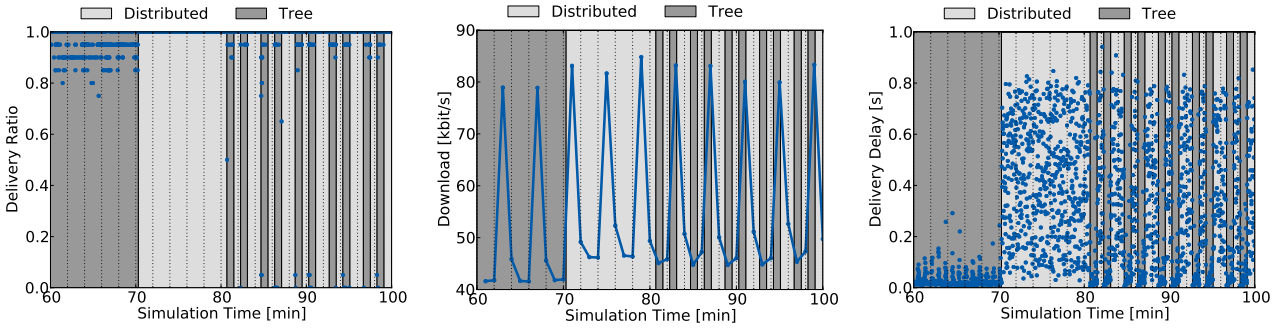
TRANSOS includes different transition strategies to accomplish unstable measured parameters of overlays and to avoid rapid transitions. For that, a simple, a hysteresis, and a adaptive transition strategy is implemented (see Chapter 6). For the evaluation of the different transition strategies, the second time workload (with rapid oscillating overlay triggers) is used. The service can react on it directly, since no further cashing or processing is required. As already mentioned, the movement speed parameter is not capable to record a rapid oscillation in the velocity of the nodes. Figure 7.16 – Figure 7.18 compares the different transition strategies under various performance metrics and simple monitoring. In the first plot it can be seen, that the simple transition strategy causes rapidly oscillating transitions. Even under global knowledge monitoring, the traffic is high fluctuating in this phase. This condition should be avoided, since it can cause many problems on the MANET itself. For example bandwidth cannot be reserved. At the first view, between the hysteresis and the adaptive transition strategy no big difference can be observed. However, it can be seen that the monitoring is discontinued for an interval after a transition occurred. If a deeper look is taken at the summarized traffic, see Figure 7.15, the adaptive strategy performs slightly better than the hysteresis strategy. This is caused by the reason that the adaptive strategy changes the monitoring interval depending on the history of transitions while the hysteresis strategy pauses the monitoring always for a predefined amount of time. Conclusively, the adaptive transition strategy only produces around half of the traffic than the simple transition strategy if the simple monitoring approach is used.



(a) Traffic Comparison (Incoming)

(b) Traffic Comparison (Outgoing)

Figure 7.15.: Traffic comparison of the different transition strategies and moitoring approaches based on the rapid oscillating time parameter.

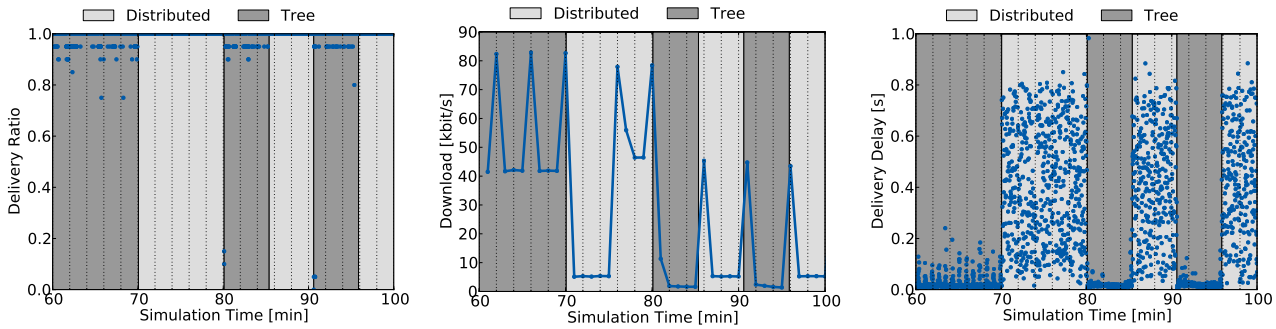


(a) Delivery Ratio (Simple)

(b) Traffic (Simple)

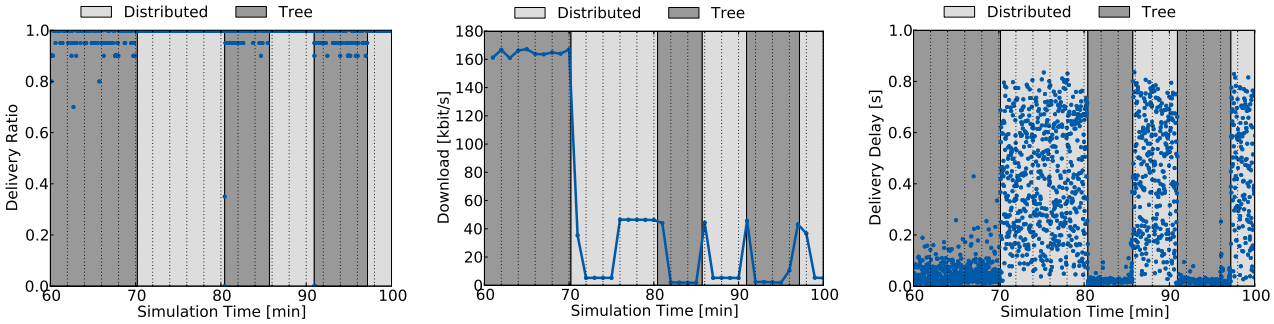
(c) Delivery Delay (Simple)

Figure 7.16.: Simple transition strategy for TRANSOS rated under various performance metrics.



(a) Delivery Ratio (Hysteresis) **(b)** Traffic (Hysteresis) **(c)** Delivery Delay (Hysteresis)

Figure 7.17.: Hysteresis transition strategy for TRANSOS rated under various performance metrics.



(a) Delivery Ratio (Adaptive) **(b)** Traffic (Adaptive) **(c)** Delivery Delay (Adaptive)

Figure 7.18.: Adaptive transition strategy for TRANSOS rated under various performance metrics.

7.7 Interpretation of the Data and Conclusive Analysis

In this chapter, the distinct pub/sub overlays, the transition-enabled pub/sub service, and different transition strategies under various conditions and workloads is evaluated. The evaluation results of the different pub/sub overlays have shown the behaviour and performance of the pub/sub overlays under different circumstances. What was seen is that the performance of one pub/sub overlay never depends on one metric, but it is a combination of all performance metrics defined at the beginning. A good result on one metric mostly comes with a drawback at another one. This can directly be seen if the traffic and the success rate of the distributed broker overlay is analyzed.

Based on these results, the transition-enabled pub/sub service was evaluated. A time parameter is used to trigger transitions manually in a controlled matter, since the triggering of a combination of parameters is not predictable and hard to evaluate. However, transition are also triggered on different movement speed and publication rate scenarios, which shows that the measurement of these parameters works as expected. In this context, the difference between the simple monitoring approach and a global knowledge monitoring component were evaluated and discussed. At the end, the simple, the hysteresis, and the adaptive transition strategy are compared to show their influence on the system. The results have shown that the more intelligent transition strategies actively prevent oscillations on TRANSOS.

8 Conclusions and Future Work

Pub/sub is a communication paradigm, which is widely used on todays Internet and social media. It also provides a solid scheme for P2P-based overlays on MANETs. In this thesis, the usage of this paradigm in a pub/sub overlay for MANETs is investigated. While one monolithic pub/sub overlay that satisfies all requirements under all environmental conditions is hard and cost intensive to develop, small lightweight overlays can be used. However, these simple overlays might only show a good performance in a special conditional range, depending on the scenario and the environment. A transition-enabled pub/sub service can use these pub/sub overlays and switch them to provide an overall system with better performance than the independent overlays.

This thesis provides a design for a transition-enabled pub/sub service, as well as three different pub/sub overlays ready to use with the system. The design is based on the extensive study of existing pub/sub systems on MANETs and fixed infrastructure, as well as on middleware infrastructure. The results of the study are used to define the requirements for transition support and the transition-enabled pub/sub service. Based on these requirements, TransOS, a transition-enabled pub/sub service, is designed and implemented. The definition of the system includes a single broker based pub/sub overlay, a distributed broker based pub/sub overlay, and a tree-based pub/sub overlay. The tree-based overlay is the most complex overlay, as it builds a connected broker substrate over the whole MANET with broker nodes and client nodes. The transition-enabled pub/sub service can monitor the MANET with generic defined parameters and can trigger transitions between these overlays based on accumulated conditions.

The distinct pub/sub overlays, as well as TransOS is evaluated under various workload definitions and scenarios. The results of this evaluation are presented and discussed. One important conclusion from the overlay evaluation is, that the performance on an overlay cannot be determined by a single performance metric. All metrics must be included in the measurement of the performance of an overlay. Since the performance under a certain environment cannot easy be predicted, the evaluation of TransOS was reduced to a functional evaluation of the triggering of transition, the evaluation of different transition strategies, and the evaluation of a simple monitoring approach and global knowledge monitoring. Future work on TransOS can include the evaluation of the system on combined conditions over a whole range of performance metrics, as it is already implemented in TransOS.

8.1 Future Work on TransOS

The presented system provides a first prototypical implementation of a transition-enabled pub/sub overlay. It serves as proof of concept and offers many directions for further research and improvements. In this section, possible future work on the TransOS system is discussed. Of course, one way to improve the system is by adding other pub/sub overlays to the transition service. The overlays presented and evaluated covered different extremes with the single broker and distributed broker approach. The tree based pub/sub overlay is a simple approach to strike a balance between these extremes. Since the system is designed generically, it is able to adapt every pub/sub overlay, regardless of its subscription model, the routing algorithm or the underlying topology. The existing overlays were implemented during the design of the transition system. It should be studied how complex pub/sub overlays with many node-types and message-types can be adapted to the API of the transition service.

Routing

For MANETs, location-aided routing (LAR) [KV00] is an interesting routing mechanism. Location information is used to predict a zone in which a message is routed, since the receiver is predicted in this

zone. If the receiver cannot be reached, a bigger zone around the last know position is drawn in which the message is tried to be delivered. This concept can also be used in the manner of pub/sub overlays by using subscription and broker information about clients in order to build an adaptive and environmental aware routing protocol. In [LJLF07] a location-aided flooding protocol for MANETs is presented which is an interesting concept for the routing protocol of the distributed broker overlay.

The presented pub/sub overlays in TransOS mostly are using custom routing protocols. The distributed broker overlay uses a flooding based routing algorithm with random early discard. This routing algorithm was chosen to simulate a very robust pub/sub overlay. The single broker overlay uses global knowledge routing as a synthetic optimal routing algorithm. The tree broker overlay routes notifications along the broker substrate. This concept can be particularly improved by the use of advertisements provided by the application. For the evaluation of the transition service, advertisements were ignored for better comparison with the simpler overlays, like the distributed broker overlay. However the use of advertisements in TransOS should be further studied.

Probabilistic Data Structures

In a system, were a large amount of data is carried around, it is important to store this information safely and efficiently. TransOS stores subscriptions and notifications in a generic data storage on every peer. This generic storage is needed to carry information about subscriptions and publications from one pub/sub overlay to another pub/sub overlay when a transition occurs. While clients only store data about their own subscriptions, brokers also store subscription and publication information about other clients, which can exceed the information about their own status by a large factor. For the evaluation of TransOS a dummy workload is applied with a relative small notification size. This is done, because the evaluation is focusing on the pub/sub functionality and the transition service. The small amount of data carried by notifications do not need a special data structure for storing or handling this data.

For a real application of the transition service or any pub/sub service, the data transmitted can have a much higher impact on the overlay and the MANET itself. A video streaming service for example may transmit many GByte of data per second through the overlay. Even while video codecs split large data into chunks, it may still use a large amount of the overall available capacity of the single peers and the bandwidth of the network. For such a scenario it is important to use intelligent algorithms to compress this data and store it efficiently. With the use of filters it is possible to categorize large amount of data and store the needed data on specific broker nodes to grant better access and reduce the impact on the network. Bloom filters, as they are described in [BM02], are a specific type of a probabilistic data structure. With the use of bloom filters it is possible to very fast derive if specific data was already sent in a data stream. The authors explicitly adduce reasons that bloom filters should be used in collaborating overlays and P2P networks. They are used to summarize content to aid collaborations in overlays. They are also used to improve the routing of the overlay by allowing probabilistic algorithms for locating resources and simplify packet routing protocols. In addition to that, they provide measurement capabilities of the infrastructure to create summaries of data in specific peers of the network. This is helpful for an efficient pub/sub overlay and once more in a transition-enabled pub/sub service. Simplified, a bloom filter functions as follows. A bit array is used for representing a set of data elements. Initially, all entries of the bit array are set to zero. Each item of the data set is hashed multiple times using fast hash functions. Simple hash methods can be used, since no cryptographic properties must be achieved. The output of the hash functions span uniform values over the size of the bit array, so each hash yields to a bit location. These bits are set to one of an item with this hash location exists. The data structure is checked for an item by also hashing it and checking the bit-location. If the location is zero, it is guaranteed that the item is not in the storage. Since hash functions may map multiple items in one location it is possible that a one in the bit-array may give a false positive result. In an pub/sub overlay, such a bloom filter can be used to decide if a full data item must be sent to a broker or not by hashing it on the entry point of the pub/sub overlay. The bloom filter of a client must be downloaded from the corresponding broker periodically. Further study of TransOS should consider and investigate the impact of bloom filter on the service. Optimized

pub/sub overlays may use the enhanced packet and resource routing capabilities, while the transition service uses the measurement infrastructure of bloom filters to improve the monitoring component.

As presented in Chapter 3, a XML-based representation and storing can also be advantageous. The topics in PeerfactSim.KOM are already modeled as URI. These can be easily adapted to follow a XML-based structure. The result of the use of such a structure is the better filtering using XML-based filtering languages. To implement such a storage, the existing storage in TransOS must be changed from an object-based structure towards a XML-based structure. A study can investigate the advantages of better filtering and storage capabilities.

Monitoring

For the prototype of the transition-enabled pub/sub service a simple monitoring module was implemented in order to provide a mechanism to observe the current state of the system. With the knowledge of the current system state, the service is able to decide which pub/sub should be used for the system. However, the provided monitoring module is a very simple approach. It just requests an update of the local observed state of the node and collects the data on a single node. Based on the collected data the an average of the whole system can be calculated. This approach also produces many overhead, since every node reports its state separately and the status request is also flooded through the network. The impact of this was evaluated by the comparison of the provided monitor approach and a optimal monitor approach which has no impact on the network. This optimal monitoring does only work in the simulated environment where a direct access on peers is possible without sending messages on the network layer. If the system is deployed on real mobile devises an optimal monitoring cannot be achieved.

Further work on TransOS can investigate the benefit of optimized monitoring approaches. There are many different ways to explore this. Typical monitoring approaches use a cascaded or an adaptive traversal of monitoring information through the network. In contrast, [SRR+14] presents a gossip based monitoring approach, were nodes talk which each other to exchange information instead of aggregating them. This system is especially trimmed for characteristics of wireless communication and mobility of nodes at a minimum of cost. Another approach is presented in [SGN+13]. This paper proposes a location-aware decentralized monitoring mechanism called BlockTree. As opposed to the centralized approach used in TransOS, a decentralized approach has to accomplish further requirements, like accuracy, timeliness, scalability, and robustness.

Another step towards a complex monitor and control mechanism is the use of different network layers for the transmission of internal used status and monitoring information. For example, the pub/sub related messages can be transmitted over the wifi interface, while status requests are transmitted via bluetooth. This of course comes with drawback of increased power consumption, due to the use of multiple radio sets. However, further work on this field may result in interesting results.

8.2 Future Work on Transition-Enabled Publish/Subscribe Systems

As mentioned in the beginning, a transition-enabled pub/sub overlay might serve as a foundation for future mobile Internet. Here, the idea of adaptivity and distributed nature is paramount. The shown transition-enabled pub/sub service is based on a monolytic concept with one node handling the control and monitoring mechanisms of the system. However, the design also allows distributed control mechanisms, within the decision process is distributed among multiple or even all nodes. Further work on this area has to investigate which agreements or rules must be placed between peers in order to make distributed decisions stable and predictable.

Self-Learning Transition-Enabled Publish/Subscribe Service

As shown during the evaluation, the values for the performance metrics of the different pub/sub overlays are chosen manually. By that, the transition-enabled pub/sub service is able to use the performance metrics of the network to calculate which specific pub/sub overlay is the best to use in the monitored

state. If the system administrator wants to enable the use of a new pub/sub overlay, he has to configure the condition for this new overlay manually. He might be forced to guess these values over the other overlays or try some values in order to find the right setup. One possible solution to this problem is to equip the transition service with self learning capabilities. This means that the administrator is able to put in every pub/sub overlay he wants and the system automatically ascertains at which state the new pub/sub overlay should be used.

One way of building a self learning service is the use of genetic programming. Genetic programming [BFKN98] allows an algorithm to find a good solution for a problem statement by optimizing overlay settings on each step of the algorithm. To measure the progress of the optimizing process, a fitness function determines the current performance of the whole transition service based on the already known performance metrics. A genetic algorithm can only be applied, if the model problem can be reduced to a single bit array, similar to a DNA string of a living organism. This is the case in our system, since the behaviour of the transition-enabled service is only determined by the settings and overlay conditions. Each step of the algorithm varies the settings by a small amount and then the performance of the system is measured again. A change in the bit array is called mutation. If the mutation rate is 0, no changes are applied to the bit array. If the mutation rate is 1, the outcome differs from the original by a maximum amount. Normally, with a mutation rate of 0.05 – 0.1, a good progress of the algorithm can be achieved. Each variation of the settings is called generation. The second mechanism used in a genetic algorithm is the crossover of different generations and paths of the search tree. On a crossover, the bit array is split into two or more parts and combinations out of two generations are constructed. A high crossover rate leads to premature convergence of the genetic algorithm. Normally a crossover rate of 20% leads to a good result. A genetic algorithm does find a good solution relatively fast if the fitness function can be fast climbed, but it is not guaranteed the best solution will be found. This however depends on the concrete modeling of the scenario in the mentioned bit array. The search is terminated if the solution satisfies a minimum criteria, a fixed number of generations is reached or further progress does not change the fitness of the performance metric significantly.

Another drawback is that the learning process can only find solutions for the current shaping of the world the system is placed in. So if the simulations are only running on one world model, the solution which will be found is only practically for this scenario. However, small variations defined by random events do not hugely change the outcome. The more different system state problems must be covered, the more expensive the learning process is.

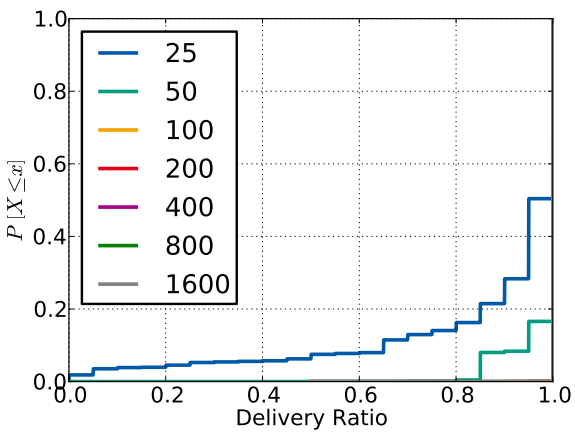
Machine learning can also be supported by providing rules. The rules define relations between variables of the pub/sub overlay and the condition it should be used in. A rule based on the amount of brokers on an overlay would discard the use of a single broker overlay if a specific threshold of nodes is exceeded, since it is expected that the high utilization of the broker will result in a bad performance of the overall system. However, this changes if the single broker is a server with suficient ressources, specifically optimized for this type of application. In conclusions, there are many ways of building a self-learning transition-enabled pub/sub service and it is worth to take a deeper look at the field of machine learning to optimize the automatization of the system.

The results presented in this thesis show, that a transition-enabled pub/sub service is realisable and provides superior performance using basic mechanisms and exchanging those based on environmental conditions. Next steps include the deployment of TRANSOS on a testbed with real devices, e.g. mobile phones, in order to evaluate the performance of a transition-enabled service under real-world conditions.

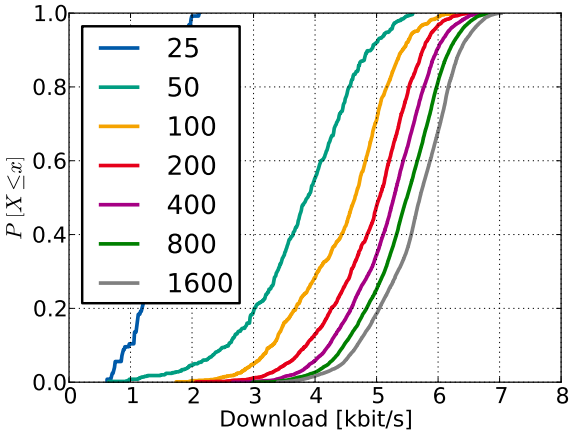
A Additional Evaluation Results

In this section, additional results from the evaluation presented in Chapter 7 are listed. This includes plots from the evaluation of the distinct pub/sub overlays, as well as performance metrics under specific conditions from TransOS that are not included in Chapter 7 due to space constraints.

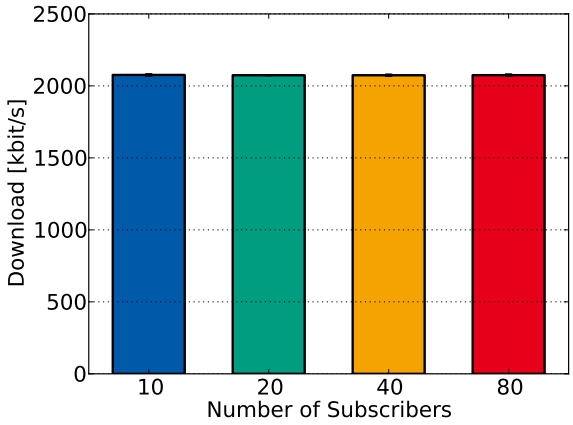
Additional Evaluation Results for the Distributed Broker Overlay



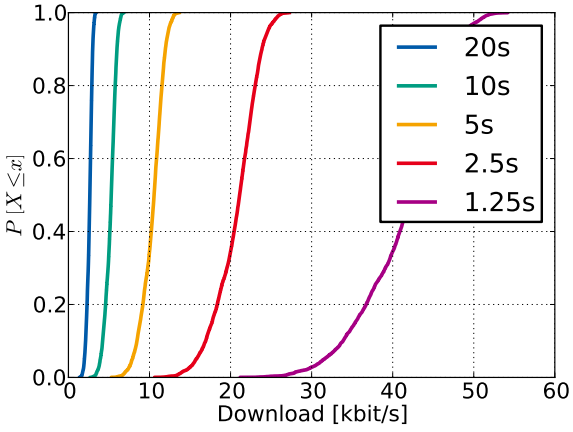
(a) Delivery Ratio Distribution vs. #Peers



(b) Traffic Distribution vs. #Peers



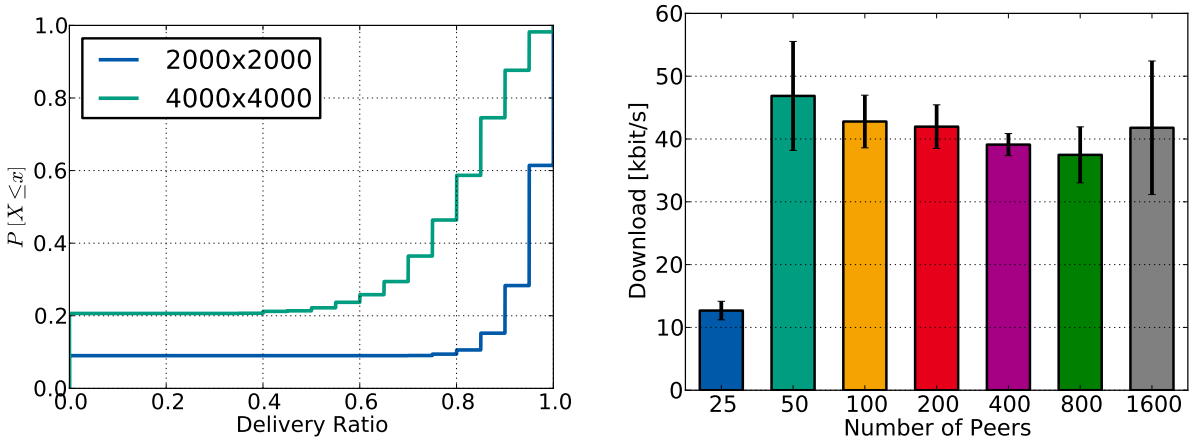
(c) Summarized Traffic vs. #Subscriber



(d) Traffic Distribution vs. Publication Delay

Figure A.1.: Additional evaluation results for the distributed broker overlay. The delivery ratio starts to drop at the lower end of the number of peers due to the formation of clusters (a). The traffic is fairly distributed over the peers regardless of the total number of peers (b). The number of subscriber does not have an influence in the summarized traffic (c). The traffic is fairly distributed over the peers regardless of the delay between publications (d).

Additional Evaluation Results for the Single Broker Overlay

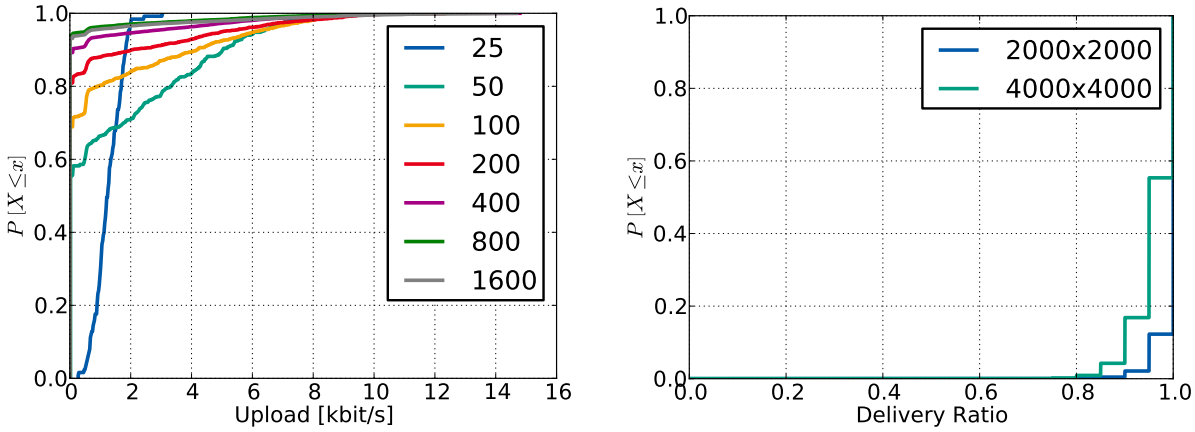


(a) Delivery Ratio vs. World Size

(b) Summarized Traffic vs. #Peers

Figure A.2.: Additional evaluation results for the single broker overlay. The delivery ratio is more likely to drop with a larger scenario size (a). The number of peers does not have a negative influence on the overlays used traffic (b).

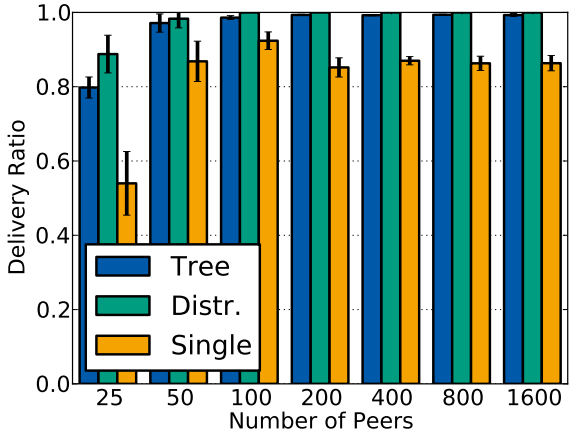
Additional Evaluation Results for the Tree Broker Overlay



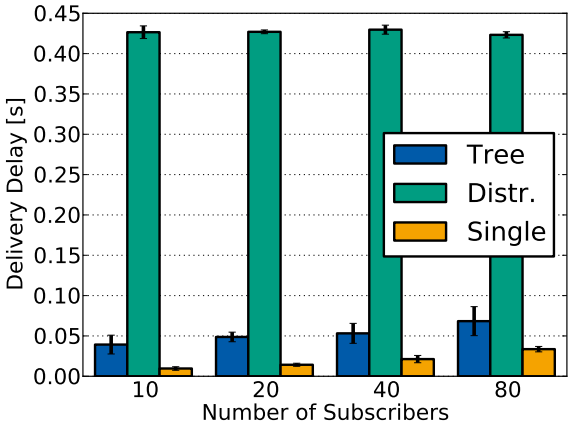
(a) Outgoing Traffic Distribution vs. #Peers

(b) Delivery Ratio Distribution vs. World Size

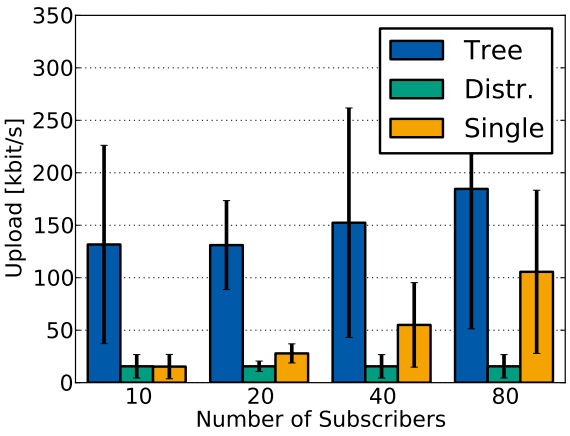
Figure A.3.: Additional evaluation results for the tree broker overlay. (a) shows the distribution of outgoing traffic over the peers. The delivery ratio in respect of the world size is shown in (b).



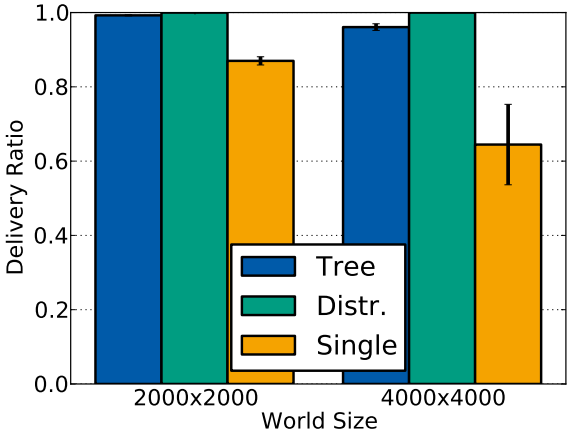
(a) Delivery Ratio vs. #Peers



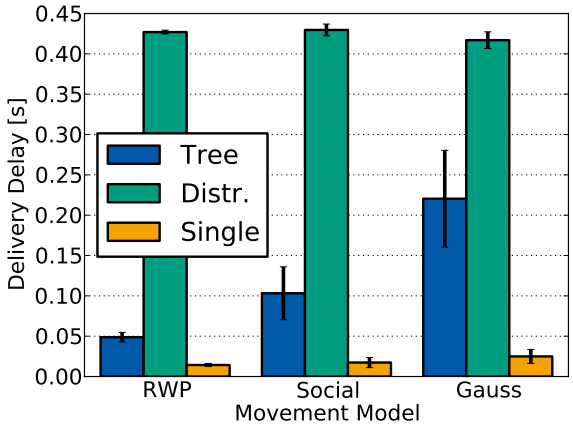
(b) Delivery Delay vs. #Subscriber



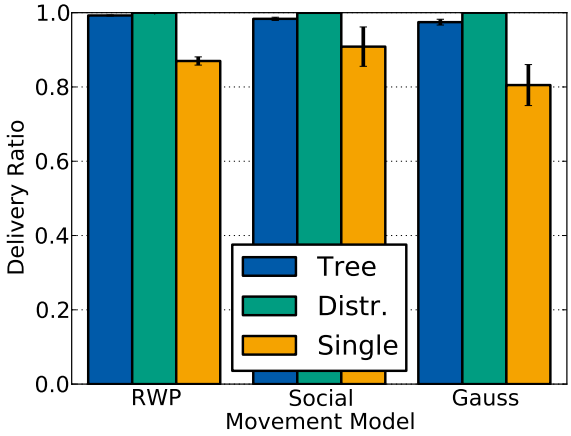
(c) Outgoing Traffic vs. #Subscriber



(d) Delivery Ratio vs. World Size

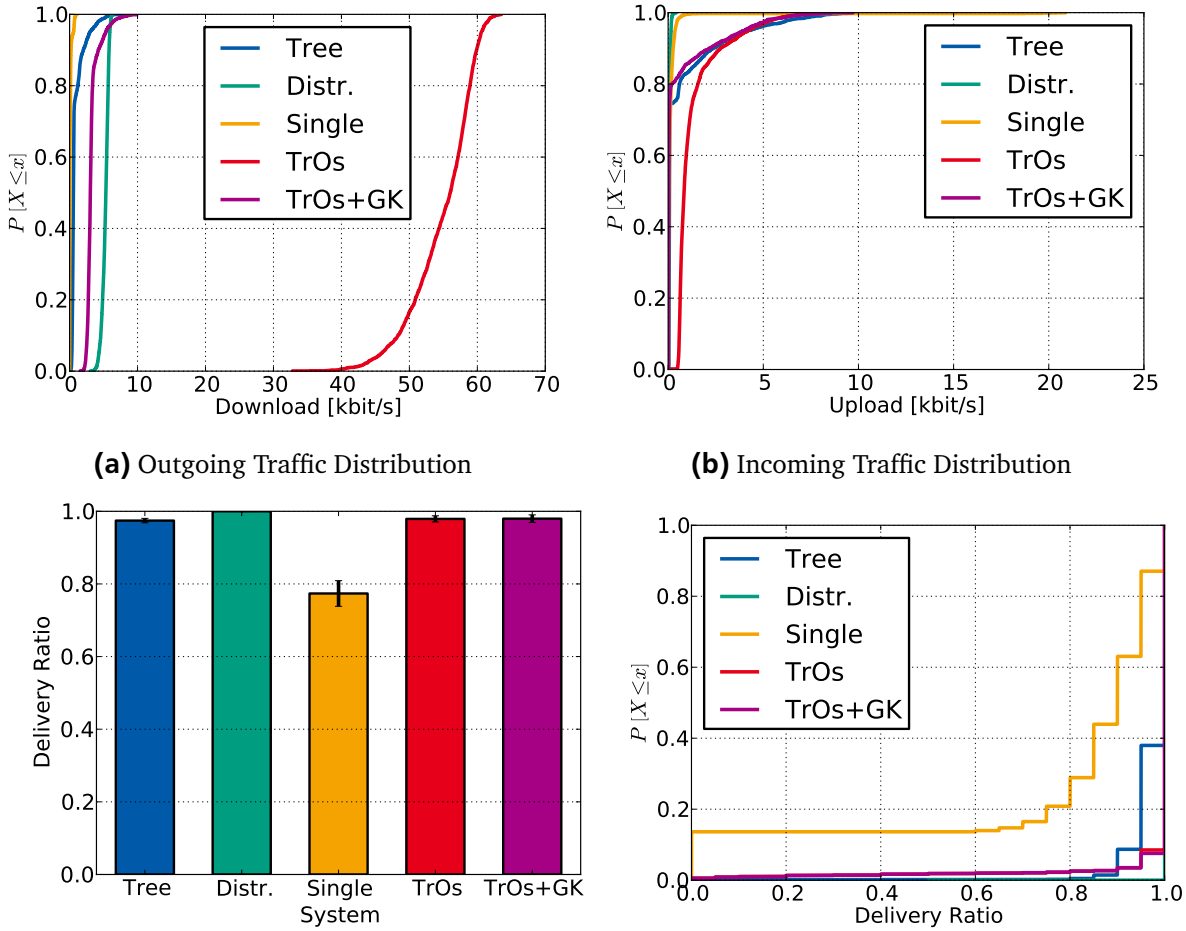


(e) Delivery Delay vs. Movement Model



(f) Delivery Ratio vs. Movement Model

Figure A.4.: Additional evaluation results for comparison of the distinct publish/subscribe overlays. Only the distributed broker overlays reaches delivery of 1.0 if no clustering occurs (a). The number of subscribers does not have a huge influence on the overlays in terms of traffic (b+c). The world size has a bit influence on the delivery ratio of the single broker overlay (d). Changing the movement model does not have a large negative effect on the overlays (e+f).



(a) Outgoing Traffic Distribution

(b) Incoming Traffic Distribution

(c) Delivery Ratio vs. System

(d) Delivery Ratio Distribution

Figure A.5.: Additional evaluation results for TRANSOS. Traffic distribution (outgoing and ingoing traffic) in comparison with the distinct overlays, TRANSOS with simple monitoring, and TRANSOS with global knowledge monitoring (a+b). The delivery ratio (c), as well as the distribution of the delivery ratio (d) shows that the transition service has no negative influence on the delivery ratio.

Acronyms

ACK Acknowledgement
AODV Ad hoc On-Demand Distance Vector
AP Access Point
API Application Programming Interface

DCF Distributed Coordination Function
DHT Distributed Hash Table
DSM Distributed Shared Memory

FDM Frequency Division Multiplexing

GSM Global System for Mobile Communications

ID Identifier
IoT Internet of Things
IP Internet Protocol

LAR Location-Aided Routing
LTE Long Term Evolution

MAC Medium Access Control
MAKI Multi-Mechanismen-Adaption für das künftige Internet
MANET Mobile Ad Hoc Network

NLRI Network Layer Reachability Information

OSGi Open Services Gateway initiative
OSI Open Systems Interconnection

P2P Peer-to-Peer
PNNI Private Network-to-Network Interface
pub/sub Publish/Subscribe

QoS Quality of Service

RED Random Early Discard
RIP Routing Information Protocol
RMI Remote Method Invocation
RPC Remote Procedure Call
RTO Retransmission Timeout

SOAP Simple Object Access Protocol

TCP Transmission Control Protocol

TDM Time Division Multiplexing
TTL Time To Live

UDP User Datagram Protocol
URI Uniform Resource Identifier

VM Virtual Machine

WAN Wide Area Network
WLAN Wireless Local Area Network

XML Extensible Markup Language

Bibliography

[ABB+09] E. Aitenbichler, E. Behring, D. Bradler, M. Hartmann, D. Schreiber, J. Steimle, and T. Strufe. Shaping the Future Internet. *Future internet of people, things and services (iopts) eco-systems workshop*, 2009.

[ADGS02] E. Anceaume, A. K. Datta, M. Gradinariu, and G. Simon. Publish/subscribe scheme for mobile networks. In *Proceedings of the 2nd ACM international workshop on Principles of mobile computing (POMC)*, pages 74–81. ACM Press, 2002.

[AEH05] E. Ahmed, A. S. Elgazzar, and A. S. Hegazi. *An Overview of Complex Adaptive Systems*. Mansoura J. Math, 32 edition, 2005.

[AJP08] N. Antunes, G. Jacinto, and A. Pacheco. On the minimum hop count and connectivity in one-dimensional ad hoc wireless networks. *Telecommunication Systems*, 39(2):137–143, June 2008.

[BBC+05] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, and L. Querzoni. Structure-less content-based routing in mobile ad hoc networks. In *International Conference on Pervasive Services (ICPS)*, pages 37–46, 2005.

[BCG04] S. Baehni, C. S. Chhabra, and R. Guerraoui. Mobility Friendly Publish/Subscribe. In *École polytechnique fédérale de Lausanne, EPFL*, 2004.

[BCM+99] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings of the 19th International Conference on Distributed Computing Systems*, pages 262–272, 1999.

[BFI+07] R. Boivie, N. Feldman, Y. Imai, W. Livenes, and D. Ooms. Explicit Multicast (Xcast) Concepts and Options. *IETF RFC 5058*, 2007.

[BFKN98] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin. Genetic programming: an introduction: on the automatic evolution of computer programs and its applications. 1998.

[BHL95] B. Blakeley, H. Harris, and R. Lewis. *Messaging and Queuing Using the MQI*. McGraw-Hill Inc., US, 1995.

[BM02] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. In *Internet Mathematics*, pages 636–646, 2002.

[BN84] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. In *ACM Transactions on Computer Systems (TOCS)*, volume 2, pages 39–59. ACM, February 1984.

[BQV05] R. Baldoni, L. Querzoni, and A. Virgillito. Distributed Event Routing in Publish/Subscribe Communication Systems: a Survey. 5(15), 2005.

[CB04] G. Coulson and G. Blair. A Component Model for Building Systems Software. In *Proceedings of IASTED Software Engineering and Applications (SEA)*, 2004.

[CBD02] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC)*, 2(Mobile Ad Hoc Networking: Research, Trends and Applications):483–502, 2002.

[CCL03] I. Chlamtac, M. Conti, and J.-N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, 2003.

[CDF01] G. Cugola, E. Di Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. In *IEEE Transactions on Software Engineering*, volume 27, pages 827–850, 2001.

[CRW00] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an Internet-scale event notification service. In *Proceedings of the 19th annual ACM symposium on Principles of distributed computing (PODC)*, pages 219–227, New York, USA, 2000. ACM Press.

[DPSB10] E. Dahlman, S. Parkvall, J. Skold, and P. Beming. *3G Evolution: HSPA and LTE for Mobile Broadband*. Academic Press, 2010.

[ECC+99] Telefonaktiebolaget LM Ericson, International Business Machines Corporation, Intel Corporation, Nokia Corporation, and Toshiba Corporation. Specification of the Bluetooth System, 1999.

[EFGK03] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/-Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[FJL+01] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *ACM SIGMOD Record*, 30(2):115–126, June 2001.

[FK12] R. Friedman and A. Kaplun Shulman. A density-driven publish subscribe service for mobile ad-hoc networks. *Ad Hoc Networks*, 11(1):522–540, September 2012.

[HGM03] Y. Huang and H. Garcia-Molina. Publish/Subscribe Tree Construction in Wireless Ad-Hoc Networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 122–140, London, UK, 2003. Springer-Verlag.

[HGM04] Y. Huang and H. Garcia-Molina. Publish/Subscribe in a Mobile Environment. *Wireless Networks*, 10(6):643–652, November 2004.

[HN11] K. S. Hazra and K. Nahrstedt. Group formation and communication in mobile wireless environments. In *Military Communications Conference (MILCOM)*, pages 1645–1650. IEEE, November 2011.

[HOTV99] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, 1999.

[IEE97] IEEE. IEEE standard for Wireless LAN-Medium Access Control and Physical Layer Specification. *P802.11*, 1997.

[KCC05] S. Kurkowski, T. Camp, and M. Colagrosso. Manet Simulation Studies: The Incredibles. *Mobile Computing and Communications Review - Special Issue on Medium Access and Call Admission Control Algorithms for Next Generation Wireless Networks (SIGMOBILE)*, 9(4):50–61, 2005.

[KR95] B. Krishnamurthy and D. S. Rosenblum. Yeast: a general purpose event-action system. *IEEE Transactions on Software Engineering*, 21(10):845–857, 1995.

[KV00] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, September 2000.

[LH89] K. Li and P. Hudak. Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems*, 7(4):321–359, November 1989.

[LJLF07] X. Liu, X. Jia, H. Liu, and L. Feng. A location aided flooding protocol for wireless ad hoc networks. In *Proceedings of the 3rd international conference on Mobile ad-hoc and sensor networks (MSN)*, pages 302–313. Springer-Verlag, December 2007.

[MC02] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Network. In *Proceedings of the International Workshop on Distributed Event-Based Systems*, pages 639–644, July 2002.

[MCP08] L. Mottola, G. Cugola, and G. P. Picco. A self-repairing tree topology enabling content-based routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(8):946–960, 2008.

[Mic00] S. Microsystems. Java Remote Method Invocation Specification. *Santa Clara, CA.*, 2000.

[MVA10] J. McAffer, P. VanderLei, and S. Archer. *OSGi and Equinox: Creating Highly Modular Java Systems*. Addison-Wesley Professional, 2010.

[PB02] P. R. Pietzuch and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. *Proceedings of the 1th International Workshop on Distributed Event-Based Systems*, pages 611–618, July 2002.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *Internet Engineering Task Force*, 2003.

[PDJ03] V. Paruchuri, A. Durresi, and R. Jain. Optimized Flooding Protocol for Ad hoc Networks. November 2003.

[PNW07] T. Pongthawornkamol, K. Nahrstedt, and G. Wang. The Analysis of Publish/Subscribe Systems over Mobile Wireless Ad Hoc Networks. In *Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*, pages 1–8. IEEE, 2007.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, Middleware*, pages 329–350. Springer-Verlag, 2001.

[RKCD01] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. In *Proceedings of the 3rd International Workshop on Networked Group Communication*, pages 30–43. Springer-Verlag, November 2001.

[RRL08] C. G. Rezende, B. P. S. Rocha, and A. A. F. Loureiro. Publish/subscribe architecture for mobile ad hoc networks. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC)*, pages 1913–1917, New York, USA, March 2008. ACM Press.

[RS13] B. Richerzhagen and R. Steinmetz. Pub/Sub for the Future Internet: Towards an Adaptive Approach Supporting Transitions. In *Proceedings of the 7th international conference on Autonomous Infrastructure, Management, and Security*, pages 84–87. Springer-Verlag, 2013.

[RWO95] S. H. Redl, M. K. Weber, and M. W. Oliphant. *An Introduction to Gsm*. Artech House Inc, 1995.

[SB05] T. Sivaharan and G. Blair. GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. In *Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems (OTM)*, pages 732–749. Springer-Verlag, 2005.

[SB07] K. Scott and S. Burleigh. *Bundle Protocol Specification*. Network Working Group, November 2007.

[Sch01] R. Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P)*, page 101, Washington, DC, USA, August 2001. IEEE Computer Society.

[Scr82] L. W. Scruben. *Detecting initialization bias in simulation output*. Cornell University, Ithaca, New York, 1982.

[SGN+13] D. Stingl, C. Gross, L. Nobach, R. Steinmetz, and D. Hausheer. BlockTree: Location-Aware Decentralized Monitoring in Mobile Ad Hoc networks. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.

[SGR+11] D. Stingl, C. Gross, J. Ruckert, L. Nobach, A. Kovacevic, and R. Steinmetz. PeerfactSim.KOM: A simulation framework for Peer-to-Peer systems. In *2011 International Conference on High Performance Computing & Simulation*, pages 577–584. IEEE, July 2011.

[SMO06] V. Srinivasan, M. Motani, and W. T. Ooi. Analysis and implications of student contact patterns derived from campus schedules. In *Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom)*, pages 86–97, New York, USA, September 2006. ACM Press.

[SRR+14] D. Stingl, R. Retz, B. Richerzhagen, C. Gross, and R. Steinmetz. Mobi-G: Gossip-based Monitoring in MANETs. In *Proceedings of IEEE/IFIP Network Operations and Management Symposium*, 2014.

[TA90] B. H. Tay and A. L. Ananda. A survey of remote procedure calls. *Operating Systems Review (SIGOPS)*, 24(3):68–79, July 1990.

[TA04] P. Triantafillou and I. Aekaterinidis. Content-based Publish/Subscribe over Structured P2P Networks. In *Proceedings of the 3rd International Workshop of Distributed Event-based Systems (DEBS)*, pages 24–25, 2004.

[Tan81] A. S. Tannenbaum. Network Protocols. *Computing Surveys (CSUR)*, 13(4):453–489, December 1981.

[TSF90] M.-C. Tam, J. M. Smith, and D. J. Farber. A taxonomy-based comparison of several distributed shared memory systems. *Operating Systems Review (SIGOPS)*, 24(3):40–67, July 1990.

[TW11] A. S. Tannenbaum and D. J. Wetherall. *Computer Networks*. Pearson Education, Inc., 5th edition, 2011.

[YB04a] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 92–97, Washington, DC, USA, 2004. IEEE Computer Society.

[YB04b] E. Yoneki and J. Bacon. Content-Based Routing with On-Demand Multicast. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 788–793, Washington, DC, USA, March 2004. IEEE Computer Society.