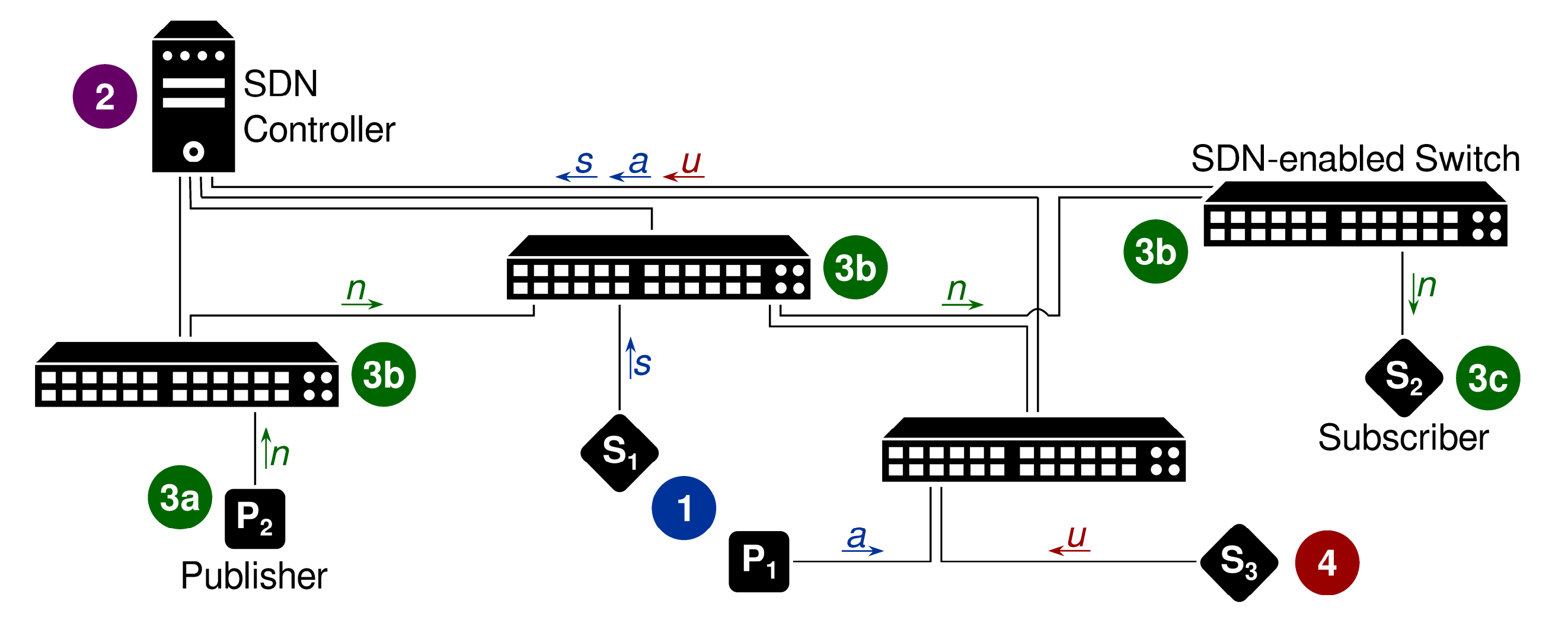
Implementing Content-based Publish/Subscribe with OpenFlow

Helge Parzyjegla[1], Christian Wernecke[1], Gero Mühl[1], Eike Schweissguth[2] and Dirk Timmermann[2]

Motivation

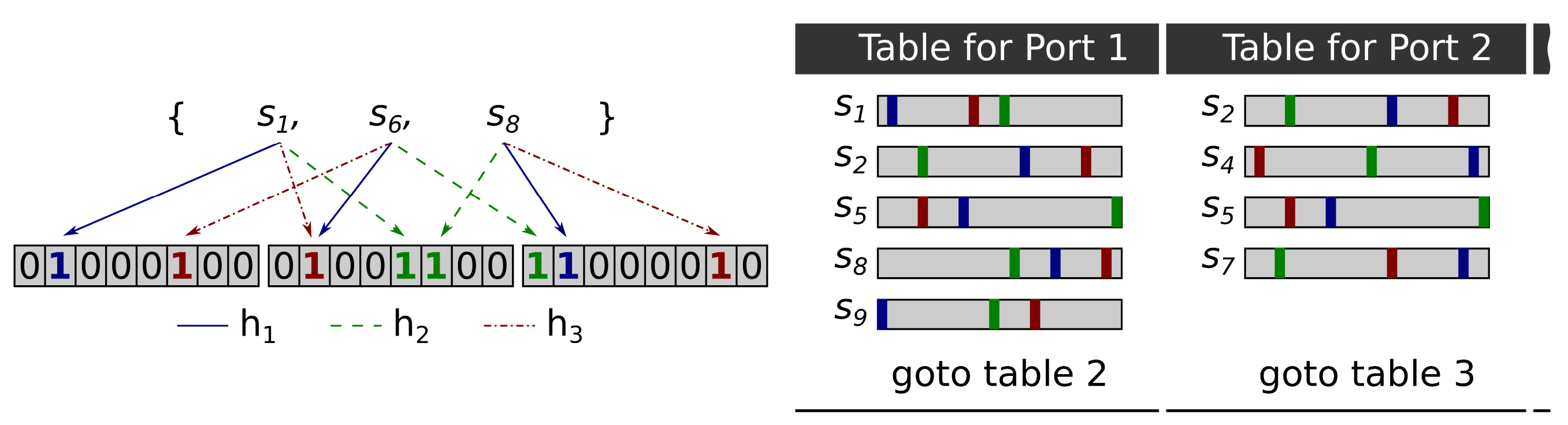
Content-based publish/subscribe is a versatile communication mechanism for building loosely coupled distributed applications. In the past, publish/subscribe systems often required an overlay network of cooperating brokers that inspect the content of a published notification on the application layer in order to flexibly make forwarding decisions. Leveraging OpenFlow-enabled network switches, forwarding decisions can now be flexibly taken within the network layer significantly reducing latency and jitter.

Publish/Subscribe Lifecycle



1. Advertising and subscribing. Publisher P_1 and subscriber S_1 send advertisement a and subscription s describing the notifications to produce and to consume, respectively.
2. Installing the distribution rules. The SDN controller determines and installs forwarding rules and tells publishers about overlapping subscriptions and their Bloom filter encoding.
3. Notification distribution. (a) Preprocessing (i.e., matching and encoding) of notification n at P_2, (b) forwarding of n, (c) postprocessing (i.e., filtering out false positives) at S_1.
4. Unadvertising and unsubscribing. Subscriber S_3 sends unsubscription u to the controller restarting step 2.

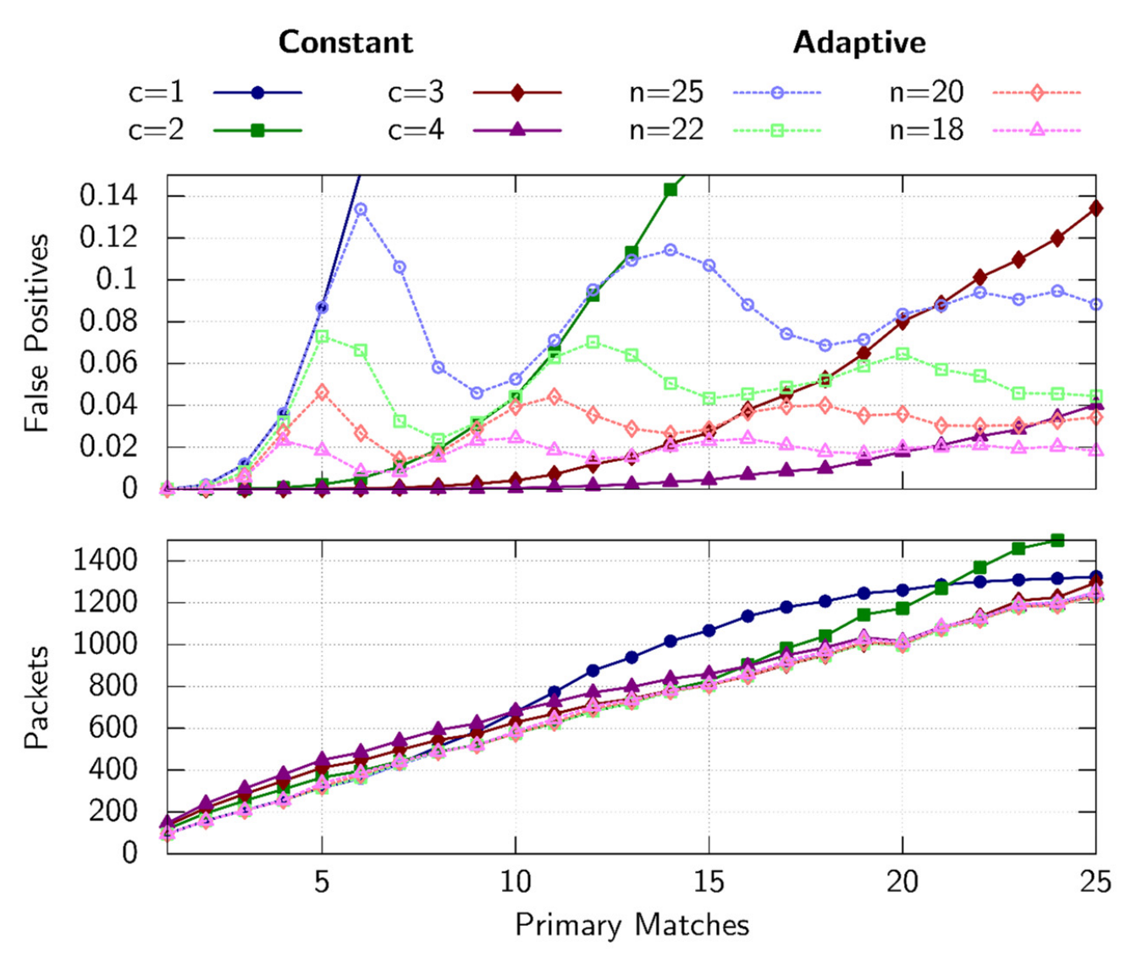
Content-based Forwarding with Bloom Filters



The key idea of our SDN-based publish/subscribe approach is to label and forward notifications by the help of Bloom filters, which encode the subscriptions matched by a notification. A Bloom filter consists of a bit array of m bits, which are initially all set to 0, and $k \ll m$ hash functions that uniformly map an element to one of the array positions. Each matching subscription is hashed by the publisher using all k hash functions and the resulting array positions are set to 1. The final bit array is embedded into the notification's IPv6 source and destination address to be inspected by the network switches. Each switch has an associated flow table for every switch port that contains an entry for every active subscription from a client reachable over this port. A flow entry matches a received packet, if the bits belonging to the associated subscription are set. In this case, a copy of the packet is forwarded on the corresponding switch port. Please note that due to the probabilistic nature of a Bloom filter, a packet may also falsely be forwarded with a certain probability.

Optimization and Evaluation

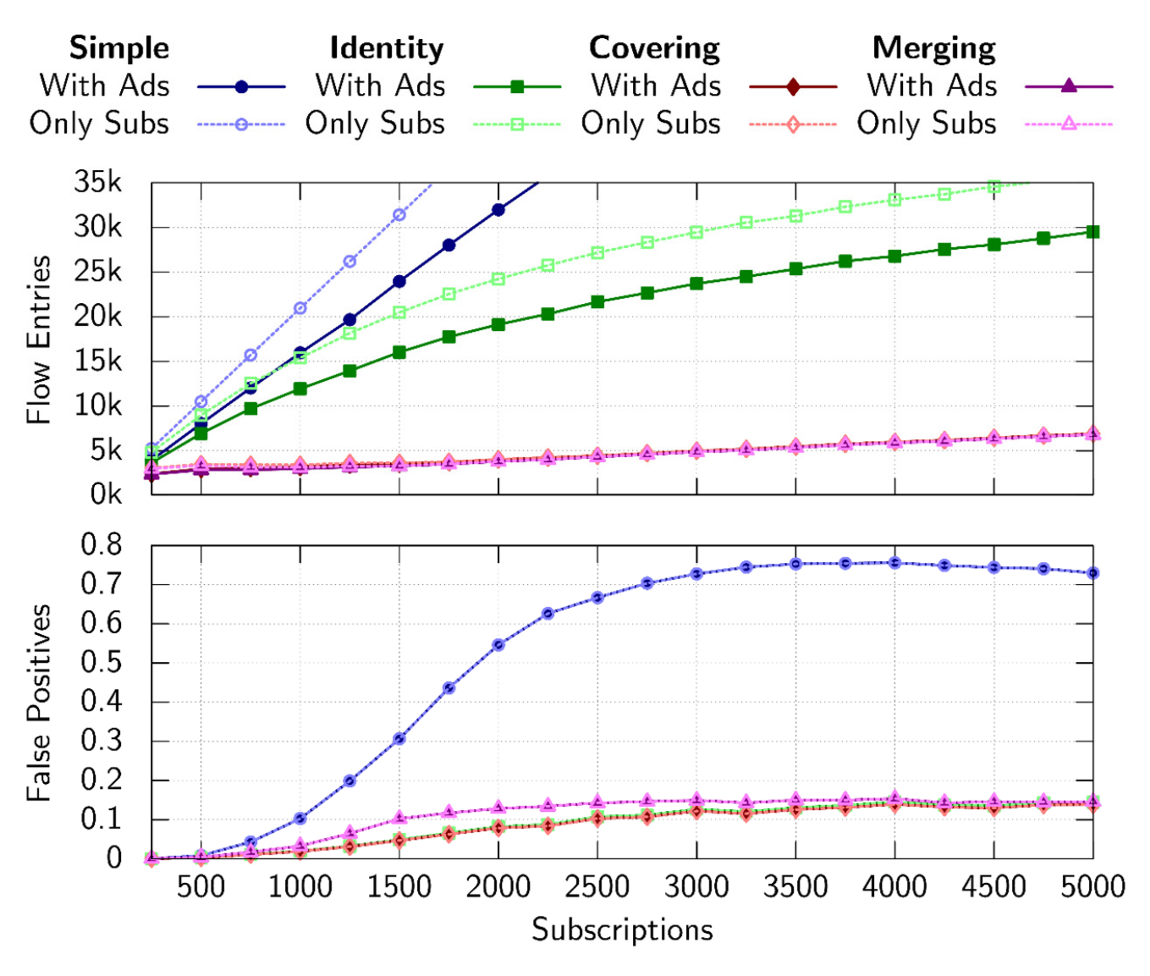
Parameters (i.e., number of hash functions k and elements n) can be tuned to keep the rate of false positives low. An adaptive scheme splits the set of matching subscriptions and distributes it over multiple notification copies if the Bloom filter gets overloaded. By exploiting similarities in the content-based filter expressions (e.g., removing identical and/or covered filters or merging new covers), the number of subscriptions is effectively reduced for which forwarding rules must be created. The smaller rule set both saves switch memory and decreases the probability of false positives. If the size of the rule set still exceeds the storage capacity of individual switches, flow entries with similar bit patterns can be merged repeatedly within each flow table. However, this reduction of forwarding rules comes at the expense of an increasing number of false positives.

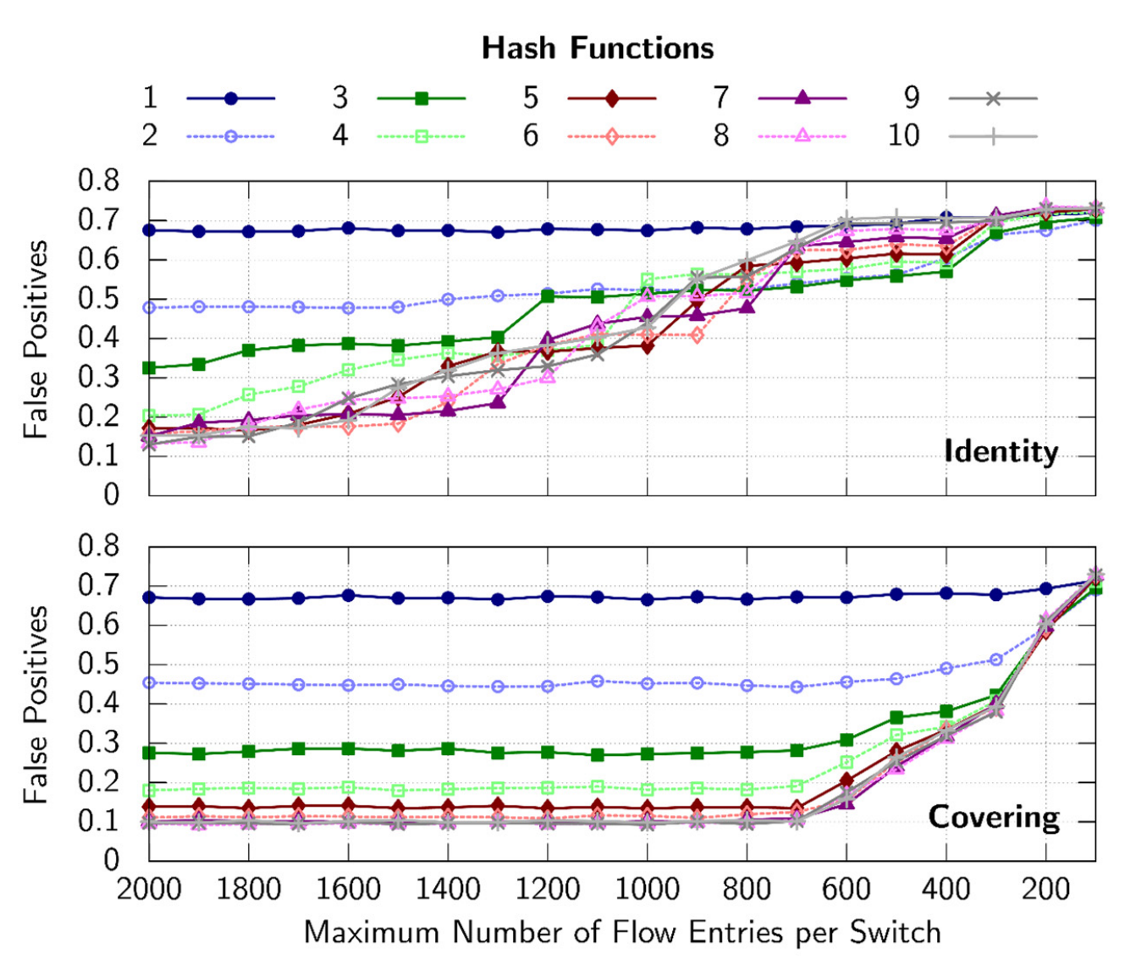


◄ Top left: Sending a constant vs. an adaptive number of notification copies; fraction of false positives and network traffic.

▼ Bottom left: Effectiveness of optimizations; number of flow entries and fraction of false positives.

▼ Bottom right: Merging of flow entries after pruning of identical subscriptions and covered subscriptions.





Conclusions

The presented approach to implement content-based publish/subscribe with OpenFlow combines efficiency and flexibility. Publishers encode matching subscriptions as Bloom filters in the IPv6 source and destination address so that the network switches can derive forwarding decisions using standard OpenFlow bit vector operations only. Furthermore, we developed and evaluated strategies to effectively limit the number of falsely delivered notifications as well as to deal with the switches' restricted capacity to store forwarding rules.

[1] Institut für Informatik | Architektur von Anwendungssystemen | Universität Rostock
Albert-Einstein-Straße 22 | 18059 Rostock | Germany

[2] Institut für Angewandte Mikroelektronik und Datentechnik | Rechner in Technischen Systemen | Universität Rostock
Richard-Wagner-Straße 31 | 18119 Rostock-Warnemünde | Germany