BSMX - A Prototype Implementation for Distributed Aggregation of Sensor Data

Sascha Schnaufer
Computer Science IV
University of Mannheim
schnaufer@informatik.uni-mannheim.de

Stephan Kopf
Computer Science IV
University of Mannheim
kopf@informatik.uni-mannheim.de

Wolfgang Effelsberg
Computer Science IV
University of Mannheim
effelsberg@informatik.uni-mannheim.de

ABSTRACT

Beacon-based Short Message eXchange (BSMX) is a system to exchange small-sized messages between unassociated WLAN devices like smartphones or access points. In this paper, we describe our proof of concept implementation of BSMX for Linux and the Android operating system. Furthermore, we introduce a novel probabilistic data structure that BSMX utilizes to provide application developers methods for distributed data aggregation.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

1. INTRODUCTION

Vehicular Ad-Hoc Networks (VANETs) are a special class of Mobile Ad-Hoc Networks (MANETs) where road vehicles with WLAN equipment form a network without additional infrastructure. In such a network each vehicle can communicate directly with all other vehicles in radio range. Typical applications for such a network try to increase the driver's safety and convenience by exchanging sensor values. For instance, SOTIS [8] and TrafficView [5] exchange information like speed and position among vehicles in order to enable users to access the current traffic conditions. Furthermore, [1] propose to equip ticket machines with WLAN so that they can inform vehicles about the capacity utilization of their parking lots. These approaches have in common that vehicles automatically aggregate received information and exchange these aggregates among each other. If a received aggregate is outdated or the distance between the vehicle and the position of the data origination is too large, the information is dropped and the dissemination is stopped. Instead of querying special information, the system works as a best effort service and automatically exchanges aggregates about the situation in the proximity of a vehicle. Some approaches also propose a hierarchical aggregation system. For instance, the authors in [1] subdivide the plane into a grid and utilize a quad tree mechanism to aggregate the utilization of parking lots. This system enables the driver to receive information about available parking lots in the proximity, but also about the situation in other districts.

The main purpose of VANETs is to increase the security by warning other drivers of dangerous situations like an emergency braking. It is obvious that due to the high risk of misuse, such a network does not use an open architecture that can be utilized to develop novel applications by everyone. The main contribution of this paper is the presentation of our technical solution to build such systems based on already deployed IEEE 802.11 devices like access points and smartphones that primarily have another intended use.

The remainder of this paper is structured as follows: The next section analyzes the mechanisms to exchange data packets between mobile devices like smartphones. Section 3 describes our novel approach called BSMX which allows the easy data exchange in a 1-hop neighborhood. A proof of concept implementation is described in Section 4. Section 5 explains an existing probabilistic data structure that can be used for the in-network aggregation of sensor values. Our novel data structure that is used in the BSMX system is described in Section 6. Section 7 illustrates the results of our simulation study. Finally, Section 8 summarizes the paper.

2. WIRELESS COMMUNICATION

Several mainstream wireless communication technologies for handheld mobile devices like smartphones are available on the market today. However, there is no established and easy to use method available to exchange packets between such devices without any using infrastructure. On the one hand, smartphones are equipped with technologies like UMTS and GSM, both using an area-wide infrastructure which is controlled by telecommunication companies that offer the access to the phone network and to the internet via their infrastructure. On the other hand, smartphones are typically also equipped with Wireless LAN (WLAN) and Bluetooth. The telecommunication companies have no incentive that devices of their customers communicate directly with each other without the usage of the companies' infrastructure and outside of their control. The situation of the second group of technologies differs clearly, because in the most cases WLAN and Bluetooth are self-governed by the device owners.

Most smartphones are equipped with a Bluetooth device of class 2 or class 3 and have a very limited radio range. Two devices need to be paired to communicate with each other. The pairing process is typically triggered automatically the

PhoneSense 2010 Zurich, Switzerland

first time a device receives a connection request. After both users have entered the identical pin the two devices can exchange files or contact information. Multicast or broadcast communication is only available if the devices operate in a Bluetooth ad-hoc network which is called piconet. However, the Bluetooth stacks of smartphones usually do not support the required profiles to operate in ad-hoc mode. Therefore, we focus the further discussion on IEEE 802.11 which is intended as a general replacement for wired networks and allows a more flexible configuration.

Although supported by IEEE 802.11, the ad-hoc mode that allows a direct communication from one device to other devices in radio range is rarely used in practice. We assume that the complex configuration of ad-hoc networks is the main reason for the current situation. IEEE 802.11 client devices can start a search over all radio channels to find access points and ad-hoc networks in radio range. If the user wants to join a discovered network the device can adopt the required network properties like SSID, radio channel and encryption method from the selected network. However, to create a new ad-hoc network the user has to configure the device manually. Furthermore, the network requires a mechanism to assign unique IP addresses to all devices, and if multi-hop communication is wanted a special ad-hoc routing protocol is required. Another issue is that many devices cannot connect to an access point for Internet access and communicate at the same time with other ad-hoc devices. We are confident that most of these configuration problems could be solved with the adoption and extension of existing technologies, but the tools the devices provide today are not sufficient or far too complex. Another reason why ad-hoc communication did not find the way to the market yet is that no unique feature or killer application is available which requires it. Our conclusion is that the already widely-used mobile Internet connections via UMTS are sufficient to enable end-to-end communication between mobile devices. However, we assume that an easy to use 1-hop data exchange mechanism can generate a significant user benefit and enable new types of applications.

3. BEACON-BASED SHORT MESSAGE EXCHANGE

The IEEE 802.11 standard defines a set of procedures to create, join and maintain WLAN networks. These mechanisms require additional packets that are not forwarded to the operating system. Furthermore, the standard allows manufacturer-specific components in most of these management packets. We have developed an IEEE 802.11-compliant extension that allows user space applications to add short messages to these management layer packets. Moreover, we also developed a mechanism that forwards this user data to the related application. A major advantage of our approach is that it does neither require a common SSID nor negotiate encryption settings or routing layer configurations. The idea is that WLAN devices can operate without changing their network and security configurations. We call this novel communication method *Beacon-based Short Message eXchange* (BSMX).

Access points and ad-hoc network nodes send unencrypted beacon packets periodically, typically every 100 ms. Our approach can be utilized to add small-sized messages to these beacon packets which are sent independently from the current network load, anyway. This exchange method leads to a heterogeneous radio channel configuration. However, the distance between the IEEE 802.11b/g channels is 5 MHz only, and the used channel width is 22 MHz. This overlap and the used encoding technique allow the successful decoding of some packets which are transmitted on adjacent channels by using a technique called overhearing. Network devices drop such packets by default, but our BSMX system uses this channel overlap to monitor a part of the frequency band without changing the radio channel of the device. In a comprehensive measurement study we analyzed the expected connectivity between indoor access points and a mobile device on the street in an inner city environment [7]. The average reception rate is 53% for packets that are sent on the same channel, 35% for packets that are sent on adjacent channels and 4% for packets with a distance of two channels. The low rate of 53% is caused by the fact that most access points should only cover indoor areas, whereas the measurement was performed on the street.

One drawback of this mechanism is the fact that mobile devices like smartphones typically run in client mode and hence they do not send beacon packets continuously. Thus, a mobile device running in client mode can receive messages from access points which operate on the same radio channel but cannot send messages back or even communicate with other client devices. Therefore, the extended version of our approach utilizes the active scan procedure which is defined in the IEEE 802.11 standard to discover access points in the proximity. During such an active scan the device passes through all radio channels to send probe request packets that are answered from receiving access points by returning a probe response packet. The BSMX system can add messages to both probe request and probe response packets and can this way exchange small-sized messages between unassociated WLAN devices that do not operate on the same radio channel. This approach has the advantage that most devices can conduct an active scan while they are connected to an access point and smoothly resume the connection after the scan.

4. IMPLEMENTATION

The chance that BSMX can find a way to the consumer market highly depends on the complexity of its appropriation. Therefore, a main design goal of the BSMX system is to minimize required changes of existing software components and work principles. This consideration is the reason why BSMX utilizes the so called tagged parameter mechanism of the IEEE 802.11 standard to extend already existing management packets. This strategy has two important advantages: First, devices without BSMX support just ignore unknown tags and discard the additional data without any drawback or failure. The second advantage is that the extension of existing device drivers is comparatively easily and can be done by including the source code of our extension in less than hundred lines of code.

Figure 1 shows the system architecture. The BSMX extension is linked to the device driver and is running in kernel space. We use Netlink (RFC3549) as communication method between the extension and user space applications. The advantage of Netlink compared to other communication methods like ioctls is that it allows bidirectional communication and implements a multicast mechanism. The former one is necessary to forward received messages to user space

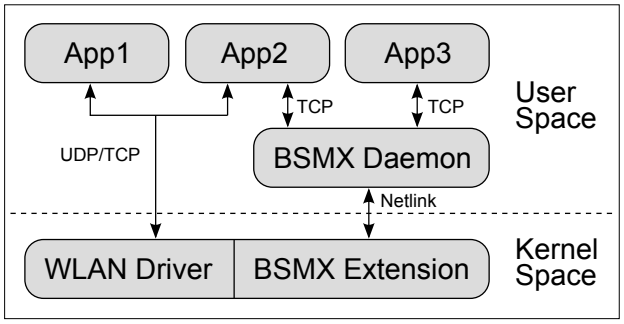

Figure 1: BSMX System Architecture

without polling. We have implemented the extension for the TNETW driver that is used by the majority of Android devices and also for the open source driver MadWifi which supports Atheros based chipsets.

The BSMX header specifies the message type as integer value but does not implement an addressing schema. We assume that most applications decide on the receiver's side whether they are a recipient. With respect to the broadcast characteristics of the ether every transmitted message can be received by every device in radio range. A new aspect is that we move the decision whether a packet should be dropped from the operating system to the application. However, it is possible that more than one application is interested in receiving this message. For this purpose we developed a central daemon that implements the publish/subscribe pattern and enables applications to subscribe the reception of messages of a specified type via TCP. Applications can also utilize the daemon to send messages over WLAN or to other local applications. Furthermore, the daemon maintains a neighbor table that can be accessed by all applications. The central instance is also required to check the security settings and deny unauthorized access to the WLAN device. Additionally, the daemon provides a standardized interface to several probabilistic data structures that can be utilized by application developers.

The intention of this approach is to provide a complete set of tools to create new types of applications. For instance, an application can aggregate the sensor values of several devises in the proximity before transferring the result to a central server for further processing. This procedure would help protecting the privacy of the application users. We also assume that for many applications it would be sufficient to know that two devices are close to each other without knowing where they are.

5. FLAJOLET-MARTIN SKETCHES

A Flajolet-Martin sketch is a data structure for probabilistic counting of distinct elements [3]. In this context an *element* represents everything for that a hash value can be calculated, e.g., a text string or a file. Please consider that the data structure does not store the element itself. The advantage is that the required storage size of the sketch only depends on the selected configuration parameters and does not depend on the number of inserted elements. A higher storage size leads to a better approximation of the number of distinct inserted elements. Two parameters influence the required storage size. M defines the accuracy and L the number of distinct elements that can be inserted. For instance, $M = 128$ and $L = 16$ leads to a storage size of 256 Bytes ($128 \cdot \frac{16}{8}$). The sketch of this example can count about 530000 distinct elements and provides a standard error of approximately 7% ($0.78/\sqrt{M}$).

An application X can add the integer value Y to a sketch by inserting Y different elements. For instance, the application can use the hash values of the strings "$X : 1$" ... "$X : Y$". In this scenario, the number of distinct values of the sketch is equivalent to the sum of all inserted integer values. For example, if two applications add values of $Y_1 = 25$ and $Y_2 = 50$, the distinct number of inserted elements is 75. The calculation of the average requires a sketch that stores the sum of all integer values and one that stores the number of inserted integer values. In the example of the parking lot application named in the introduction, the capacity utilization can be calculated by using two sketches. One contains the number of available parking lots and the other one stores the total number of parking lots.

The insertion process is deterministic and completely independent from the current state of the sketch. If one element is inserted several times, the identical bit is set to one each time. Hence, sketches are duplicate-insensitive and the order of insert operations does not affect the estimation process. These probabilities allow that sketches can be merged by a simple bit-wise OR operation. The merged sketch can be used to estimate the total number of distinct elements that are added to any of the source sketches. This behavior and the compact storage size are useful for the distributed calculation of aggregation functions like COUNT, SUM and AVG inside VANETs or sensor networks [2]. [6] introduce a compression schema for sketches and [4] provides an aging strategy that removes outdated elements from a sketch.

These approaches are suitable for the in-network aggregation of sensor values. For example, a smartphone application wants to estimate the number of other devices in the 5-hop proximity that also runs an instance of the software. In this scenario every device maintains a sketch, adds the name of the device itself, and sends the data structure via broadcast to other devices in radio range. Received sketches are added to the own sketch via a bit-wise OR operation. Then the application use an aging strategy (e.g., [4]) to remove outdated entries and send the sketch again via broadcast to other devices; these steps are repeated iteratively. After several iterations each device can use the sketch to estimate the number of running instances in the 5-hop proximity. The number of hops is a setup parameter of the aging approach that increases the required storage size of a sketch.

The described procedure is very simple and robust, and does not require any knowledge about the topology. However, the same working principle can be used without sketches by exchanging a list that contains tuples with the name of a device and a time to live (TTL) counter. In this case, each device maintains such a list and adds its own name with the maximum TTL counter. The device decrements the TTL counter of all entries beside its own entry and removes entries with a TTL counter below zero before it sends the list via broadcast to its neighbors. If a device receives a list, it adds all unknown devices to its own list and updates the TTL counter of already known devices. This way each device can also estimate how many other devices in the proximity run the application. However, this approach is very limited because the size of the list will grow very fast. For instance, if every device adds its own MAC address and

a TTL counter that can store four hops, every entry would require $6 \cdot 8 + 2 = 50$ bits. With respect to the typically used maximum transfer unit a list with 22 entries can be transferred without fragmentation at most. Compared to the sketch approach that can store several thousand entries, a capacity of 22 is extremely small.

A serious drawback of the probabilistic approach is that it is only suitable if the application can accept a standard error of at least 5%. In our simulations, values of $M > 256$ do no longer improve the accuracy significantly. Furthermore, the accuracy can only be achieved if more than $15 \cdot M$ elements are inserted. In other words the sketch we named before requires $128 \cdot 15 = 1920$ entries to achieve the standard error of about 7%. This behavior makes the configuration complex and depends on the number of participating devices. Another drawback is that the range of integer values that can be inserted is very limited. The number of elements that can be inserted to a sketch depends on its setup parameters. However, it is obvious that the insertion of 64-bit values exceed the available range clearly. For these reasons, we present in the next section a novel approach to calculate and exchange aggregates.

6. BLOOM FILTER MAPS

Flajolet-Martin sketches are suitable to estimate aggregation functions in large scale networks. However, in environments with a low number of participants the accuracy can decrease significantly. Another limitation is that a more complex aggregation can only be reproduced by using several sketches. This section presents a novel approach that overcomes these limitations and offers further advantages.

In the last section, we briefly described the distributed estimation of the number of participating devices by exchanging a list with device names and TTL counters. The sum or the average can be estimates by adding the sensor values to the related list entries. If we assume that it is typically suitable to use a range of $0 \ldots 255$ as sensor value and $1 \ldots 8$ as hop range, each entry needs $6 \cdot 8 + 8 + 3 = 59$ bits. It is obvious that the largest fraction of storage is used to identify the owner of the entry. The main idea of our approach is to substitute this identifier by a probabilistic replacement. Keep in mind that the hashing of the MAC address is a probabilistic method that can lead to hash collisions. However, the reduction of the storage amount achieved by hashing is not sufficient.

Bloom filters are a well-known data structure that is used to test whether an element is a member of a set. False positives are possible, but false negatives not. A Bloom filter is bit field $B = b_0 \ldots b_{N-1}$ of a length $N > 0$ and is initialized with zeros on every bit position. An element E is added by setting bit b_X with $X = hash(E)\ mod\ N$ to one. A similar procedure is also used to test whether an element was inserted before. If the tested bit is one, the element is member of the set or a collision had occurred (false positives). However, if the bit is zero, the element was definitely not inserted. The false positive rate can be reduced by utilizing more than one hash function, but we will focus on the approach with one hash function only.

It is possible to extend the array positions from a single bit to an n-bit counter. In our approach, a counter of zero means that there is no entry; otherwise the counter represents the TTL of the entry. To come back to the uncompressed list from the beginning of this section, the identifier

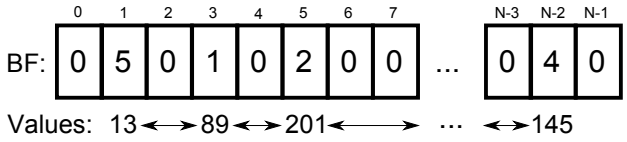

Figure 2: Bloom Filter Map

corresponds to the position inside the Bloom filter and the TTL counter corresponds to the counter at the related position inside the Bloom filter. The order of non-zero counters is used to maintain an additional list which contains the content of the entries, e.g., the sensor vales. We call the Bloom filter with TTL counters and value list *Bloom Filter Map* (BFMap). Two BFMaps can be merged by performing a MAX operation on every position of the related Bloom filters and arranging the corresponding values to a new list. If two positions have a non-zero counter, this process prefers the newer one. Figure 2 shows an exemplary BFMap. The upper part of the figure shows the Bloom filter with TTL counters and the lower part the list of related values.

It is obvious that several entries could be mapped to the identical position inside the Bloom filter. This represents the probabilistic part of our approach. The probability of such a collision increases with the number of already inserted entries and decreases if the size of the Bloom filter is increased. The interesting aspect here is that the enlargement of the Bloom filter only slightly increases its entropy. If the probability of each symbol of a data stream is known, the well-known arithmetic coding approach [9] can be utilized to compress the data stream entropy-optimal. A Bloom filter with T-bits TTL counters consists of the symbols $0 \ldots (2^T - 1)$. Assume that $count(X)$ is a function that returns the number of occurrences of the symbol X inside a Bloom filter, then the probability of each symbol can be calculated by $count(X)/N$. If a device receives a compressed Bloom filter, N and a list of $count(X), X \in 0 \ldots (2^T - 1)$, it can decompress the data structure. It is sufficient to use 16-bit unsigned integer values to store and transmit N and the counters.

7. EVALUATION

In [7] we estimated the connectivity of access point among each other in the inner city of Mannheim, Germany. In the following, we will use this very dense network to conduct a simulation study to estimate the accuracy and the required storage size of the BFMap approach. The simulator creates for each of the 3797 simulation nodes a BFMap and inserts a tuple based on a random integer value and the maximum TTL T_{max}. In the next step, the simulator adds to the BFMap of each node the values of their neighbors with a shortest path of x hops ($x \in 1 \ldots T_{max}$) with a TTL of $T - x$. This procedure emulates the TTL reduction, the exchange, and the joining process of BFMaps. Finally, the simulator compares the created BFMaps with the real situation. The comparison includes the number of neighbors (COUNT), the sum of random values (SUM) and the calculation of the average value (AVG). Furthermore, the simulator compresses each BFMap and determines its compressed size. Figure 3 shows extracts of the average results of 150 simulation runs each using 3797 nodes.

The average size of a compressed Bloom Filter entry de-

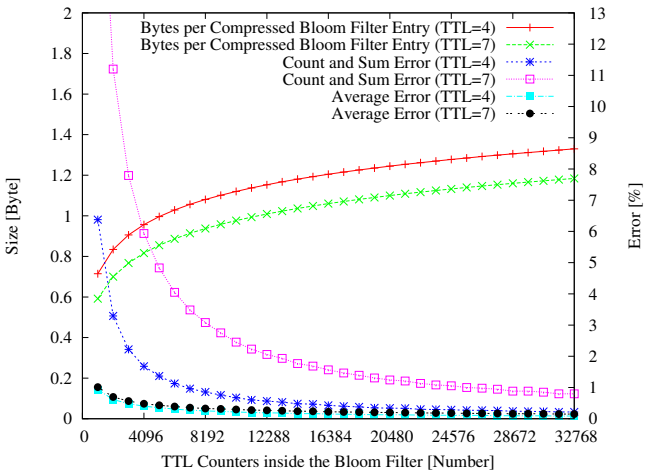

Figure 3: Simulation Results

pends on the number of inserted elements, the capacity N of the filter, and the range of the TTL counters. In our simulation environment the maximum TTL determines the number of neighbors inside the TTL limit and thus it also defines the number of inserted elements. Figure 3 shows the average entry size for a maximum TTL of 4 (141 entries) and 7 (520 entries) respectively. This size includes the identifier and the TTL counter of a node but not its value. Thus, if one byte per value is used, the full size of the BFMap entry is also increased by one byte. Although the compression of a higher maximum TTL requires more symbols, the average entry size of the TTL=7 example is lower compared to TTL=4. The reason for this behavior is that the required storage space of the zero TTL counters is distributed across more inserted entries.

The results show that the usage of a BFMap with 8192 TTL counters is a good trade-off between size, computational complexity, and accuracy. The average count and sum error of this setup is about 1% with 141 entries and 3% with 520 entries. An interesting aspect of the BFMap approach is that even in the case of collisions the newly inserted element only overwrites an existing entry without influencing the other elements. In other words, the number of collected samples to calculate an aggregation is reduced by one, but all remaining samples are unchanged. This effect causes the small error of the average calculation. We run simulations based on uniformly and normally distributed random values, but the differences were marginal, and therefore the figure shows the former one only.

The presented BFMap data structure is well suited for the distributed calculation of aggregation functions. The advantage is that a developer can use real values for the calculation and is not limited to basic functions like COUNT, SUM and AVG. In addition, the developer controls the size and type of the values used and decides which accuracy the application requires. For instance, the developer can use a quantization method to reduce the required size per value. Furthermore, the BFMap can be used to maintain and exchange applications states or other kind of data that is not intended for aggregation purposes. Contrary to Flajolet-Martin sketches BFMaps do not require a minimum of inserted elements to work properly. However, the size of a BFMap increases with number of its entries in a linear way, and thus they are not suited to calculate and exchange aggregates of several thousand entries.

8. CONCLUSION

We present our BSMX system that can be utilized to create novel applications without complex device configuration or significant impairment of the device's main functions. Our approach also allows the development of hybrid applications that communicate via WLAN and Internet. A sample application could be to exchange public IP addresses via WLAN for further communication or to detect the closeness of other devices without GPS and central server. Furthermore, we presented a novel probabilistic data structure that can be utilized to aggregate sensor data in a distributed manner. The data structure is also used by our prototype to maintain a multi-hop neighbor table which is accessible to subscribed applications. In future work, we will improve our prototype implementation and make it available under an open source license.

9. REFERENCES

[1] M. Caliskan, D. Graupner, and M. Mauve. Decentralized discovery of free parking places. In *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 30–39, New York, NY, USA, 2006. ACM.

[2] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 449–460, April 2004.

[3] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[4] C. Lochert, B. Scheuermann, and M. Mauve. Probabilistic aggregation for data dissemination in vanets. In *VANET '07: Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pages 1–8, New York, NY, USA, 2007. ACM.

[5] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: traffic data dissemination using car-to-car communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(3):6–19, 2004.

[6] B. Scheuermann and M. Mauve. Near-optimal compression of probabilistic counting sketches for networking applications. In *Dial M-POMC 2007: Proceedings of the 4th ACM SIGACT-SIGOPS International Workshop on Foundation of Mobile Computing*, Aug. 2007.

[7] S. Schnaufer, S. Kopf, H. Lemelson, and W. Effelsberg. Beacon-based short message exchange in an inner city environment. In *9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2010)*, June 2010.

[8] L. Wischhof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis - a self-organizing traffic information system. *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, pages 2442–2446 vol.4, April 2003.

[9] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.