Engineering a Suburban Ad-Hoc Network

Mike Tyson, Ronald D Pose, Carlo Kopp, Sk. Mohammad Rokonuzzaman, Muhammad Mahmudul Islam
Faculty of Information Technology, Monash University
{mtyson,rdp,carlo,rokon,sislam}@csse.monash.edu.au

Abstract

Ad-hoc wireless networks are networks of co-operating users, with no reliance upon fixed infrastructure. Such networks are growing in popularity, as wireless communication hardware, both fixed and mobile, becomes more common and affordable.

The Monash Suburban Ad-Hoc Network (SAHN) project has devised a system that provides a highly secure and survivable ad-hoc network, capable of delivering broadband speeds to co-operating users within a fixed environment, such as a residential neighbourhood, or a campus.

The SAHN can be used by residents within a community to exchange information, to share access to the Internet, providing last-mile access, or for local telephony and video conferencing. SAHN nodes are designed to be self-configuring and self-managing, relying on no experienced user intervention. Thus, they are suitable for use by the general public, in 'plug-and-play' fashion.

This paper investigates possible architectures for an implementation of the SAHN (Tyson 2005), and presents a real-world prototype. The prototype presented takes the form of a Linux kernel module, and a user-space daemon.

Keywords

Algorithms, Performance, Design, Security

INTRODUCTION

In our modern, information-centric society, the availability of adequate connectivity is of high importance. Not only is more pressure placed upon the existing communications infrastructure, but the need to extend connectivity to areas previously not covered is growing. In many areas, communications services are expensive and often unreliable, if not entirely unavailable.

Suburban Ad-Hoc Networks (SAHNs) are economical, high-speed connectivity alternatives for communities of co-operating users in a suburban environment (Islam, Pose, Kopp 2003a) (see Figure 1). They offer affordable broadband access where other techniques may not be viable. Importantly, the SAHN offers a viable alternative

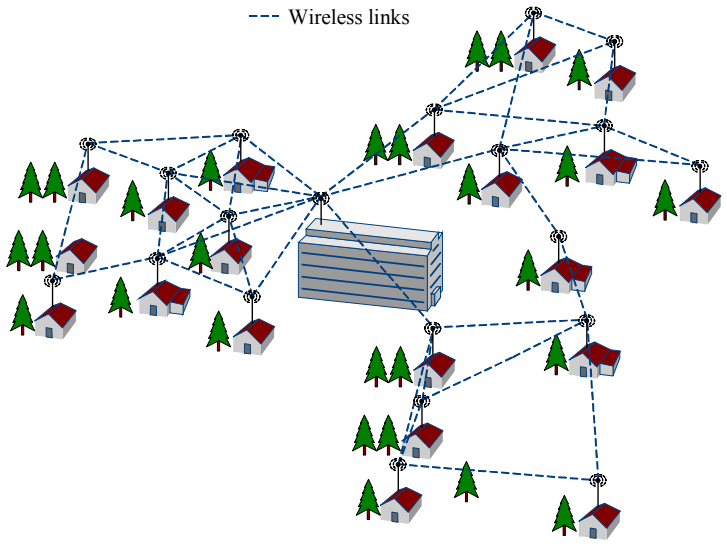

Figure 1: Sample SAHN Scenario

to fixed broadband services in low population density areas, where distance to exchanges, or low demand, make such fixed services overly expensive, or entirely unavailable. SAHNs provide for community interaction, or to provide last-mile Internet connectivity. They can be used for local video and audio communication, or community bulletin boards. Research is being done into localised virtual reality systems, using the SAHN.

The SAHN system has no dependence on existing infrastructure, as the network participants themselves co-operatively form the network and route traffic. Consequently, recurring costs are low, making the technology more accessible to the public (Pose, Kopp 1998).

The SAHN is designed to be independent of particular technologies, and is intended to be built with commodity, off-the-shelf hardware, such as PCs, or embedded systems, to keep initial costs down.

SAHNs are self-forming, self-healing networks which operate over a typically wireless medium, although they can incorporate many different physical media. They operate in arbitrary, dynamic topologies. Consequently, highly-connected networks can be formed, providing high redundancy and the capacity for load-sharing over multiple links. SAHNs can be used to integrate and aggregate many different physical links, enhancing the availability and value of a network architecture. SAHN nodes are designed to be self-configuring and self-managing, resulting in the ability for them to be owned and maintained by members of the general public, without particular expertise.

In an age where information systems are becoming increasingly exposed to attack, in the form of direct attacks against the network or its users, or passive monitoring for espionage or other illegal purposes, security is critical. This becomes particularly true when using an inherently insecure channel, as in wireless networks. Additionally, controlling access to network resources is important for a modern network architecture.

In the SAHN, security is built in at a low level (Islam, Pose, Kopp 2004a, 2005, Islam 2006). Fully self-organised key and access control management facilities are available, and network layer packets are encrypted at each hop. For this reason, the SAHN does not use the standard TCP/IP stack form, and instead provides a re-designed layer structure (see Figure 2 (Islam 2006)). Thus, instead of being limited by the security of the TCP/IP stack, this allows optimisation for SAHN-style network environments, where encryption should encapsulate all data, and thus sit below other layers.

The SAHN Security Protocol (SSP) resides between the network and the link layer, and protects packets originating from upper layers. Thus, it forms an underlying secured layer for protocols that operate above SSP.

SSP protects users from malicious entities by both encrypting the user's data, and providing authentication of the source of all packets (Figure 3 (Islam 2006)). SSP is effective against most common attacks, such as broadcast storm or piggyback attacks, tunnelling and identity theft, as detailed in (Islam 2006).

SSP uses asymmetric cryptography for signing and encrypting, or decrypting and verifying SSP control packets. For all other network packets, SSP uses symmetric cryptography. No centralised certification authority or centralised trusted entity is necessary for distributing and managing capabilities and keys; a fully self-organised capability and key management system is provided.

Performance advantages are gained by only requiring certificate verification when a node joins the SAHN, or when a certificate is revoked. Ordinary network packets are encrypted and authenticated with symmetric cryptography, with less intensive processing requirements.

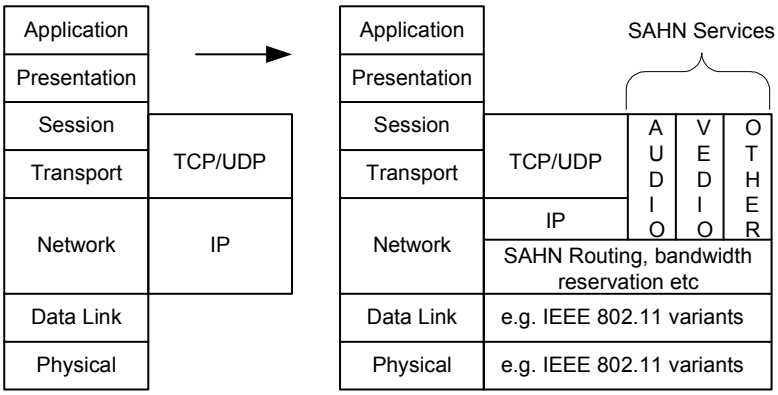

Figure 2: SAHN/OSI Comparison

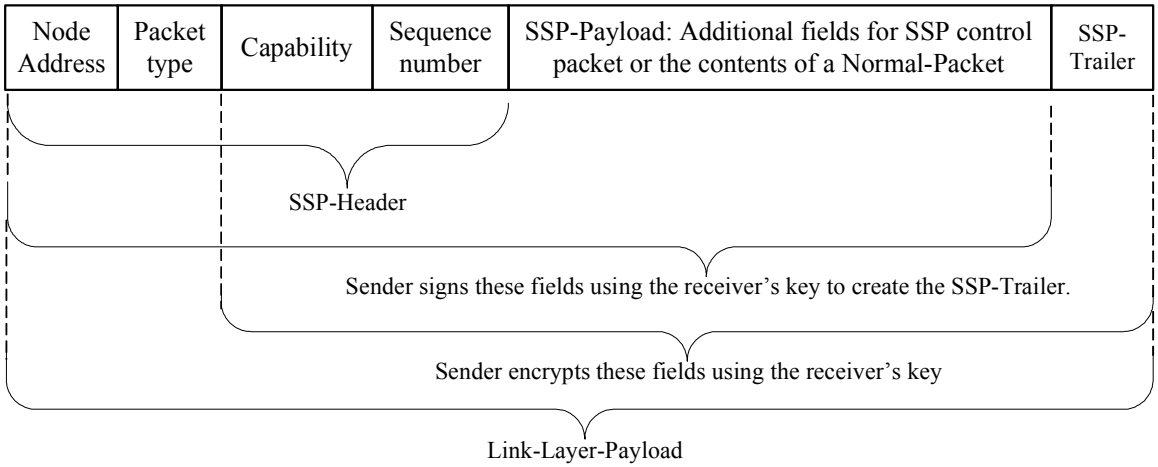

Figure 3: A Link-Layer Payload produced by SSP

SSP is based on the password-capability model (Anderson et al., 1986, Castro, 1996, Kopp, 1997, Pose, 2001). A capability is a token that identifies an object or resource, and grants some level of access to that resource. Capabilities can be granted or revoked by any entity for any other entity in the system. Such a model provides for a simple and robust access control management scheme.

SSP provides services, identified and controlled by capability tokens, to authenticate a new neighbour for local (single hop) communications, authenticate a network membership for network-wide communications, perform secured routing, and further propagate or revoke capabilities.

A new node wishing to join a SAHN initially communicates out-of-band information to an existing SAHN node to authenticate the new node, or is invited by an existing node. Provided the existing node possesses a new neighbour authentication capability, it authenticates the entering node for local, single hop communications. This permits the entering node to communicate with the existing node only.

The entering node is then required to obtain a network membership certificate from more than one network member. This operation can be performed with a fully distributed certification authority (Zhou and Hass, 1999), and can be executed by the existing SAHN node (which has network membership allowing communication with other nodes) on behalf of the entering node.

Once equipped with a network membership certificate, the entering node requests a Routing Capability from the existing node, using the network membership as authentication. Once granted, the new node can begin routing through the SAHN, and is accepted as a network member. A network membership certificate can be revoked if necessary, thereby revoking a node's network membership.

Network membership authentication yields significantly more security than protocols based only on transitive trust. Two levels of protection are offered with the two-stage authentication procedure, requiring an initial authentication to perform single-hop communications with a single neighbour, followed by a network membership authentication procedure, to begin communicating with the entire network.

Packet integrity is assured by the inclusion of a digital signature, encrypted with the rest of the packet contents. This signature is formed by a one-way hash, using the packet contents and the receiver's encryption key as input. This both protects against packet manipulation or fabrication by a third party, and provides a robust error detection mechanism. Each packet is verified against the digital signature when it is received. Packets that fail the verification procedure are dropped. Thus, identity theft, protocol field modification and message fabrication attacks are prevented.

Replay attacks are prevented by the inclusion of a monotonically increasing encrypted sequence number for each neighbour. As the sequence number is encapsulated within the packet's encryption layer, this cannot be fabricated.

Additionally, encryption keys are updated and capabilities are re-issued at pseudo random intervals in order to prevent keys being compromised using brute force cracking methods. Updates are performed more rapidly than an external party could recreate the encryption keys and capabilities.

In addition to SSP, an intrusion detection system, SAHN-IDS, is built into the SAHN, which detects and responds to anomalies caused by misbehaviour or faults. This is achieved by monitoring channel usage of neigh-

bouring nodes, and monitoring of nodes along each routing path. Nodes within the local node's transmission footprint are monitored by taking note of the time each neighbouring node spends using the common channel. Nodes along a routing path are monitored in this fashion by neighbouring nodes, and statistics are periodically unicast back to the source node. This two-fold monitoring approach is capable of detecting most network anomalies such as non-compliance with the MAC protocol or bandwidth reservation scheme, jamming, black- and grey-hole attacks, and selfish packet forwarding. The network can then respond by either automatically excluding misbehaving participants from network operation, and/or informing a human operator (Islam 2006).

The SAHN also possesses a distributed admission control and bandwidth reservation protocol, to offer QoS facilities. This provides for the use of the SAHN for real-time applications, such a video or audio conferencing.

Supporting the admission control and bandwidth reservation protocol, as well as the security protocol, the SAHN possesses a capability-based resource access control mechanism (Bickerstaffe 2001, Gunawan 2003), which provides for fine-grained control over access to network resources.

Where many traditional ad hoc networks rely on omnidirectional antennas, SAHN is capable of operation with directional and electronically steered antennas (Islam, Pose, Kopp 2004b), giving the SAHN the ability to maximise range, enhance security, and maximise usage of the radio spectrum. The use of directional antennas results in a very different network environment, requiring new protocols to operate and to exploit the nature of directional antennas.

As suggested by the name, Suburban Ad-Hoc Networks are designed for a suburban setting. Consequently, they are optimised for a 'quasi-static' environment, where nodes rarely move or disappear. This allows for higher transfer rates, particularly, than in traditional mobile ad-hoc networks. In addition, the SAHN is symmetric in upstream and downstream channels, making viable a wide range of higher-level protocols such as video/audio conferencing and file-sharing protocols.

As mentioned above, the SAHN is designed to be independent of any particular technology configuration. Thus, many implementation options are available. For example, a monolithic implementation of the SAHN system running on a single-board embedded system is suitable for a modem-type router device. An installable operating system driver could allow users to run the SAHN software on their existing PC hardware. A 'SAHN-OS' self-contained operating system could be installed on commodity PC hardware for a dedicated 'SAHN gateway'.

An embedded SAHN device could provide for mass-production, offering a self-contained solution. Operating system drivers or self-contained operating system software can be distributed to keep costs down, using existing hardware.

To demonstrate one such engineering method, we have developed a sample architecture for a SAHN implementation, and produced a working prototype implementation under Linux (Tyson, 2005). While no assertions are made that the presented implementation is optimal for the SAHN, a prototype under Linux can be produced cheaply, quickly and easily, as the Linux O.S. is an open system. This provides a rapid proof-of-concept to provide further verification and profiling.

The SAHN prototype needed to be extensible, and thus modular, with well-defined interfaces. In addition, it needed to avoid making assumptions about the underlying platform as much as possible, to maximise portability. This allows the prototype to be moved between platforms easily, a useful property for an initial prototype, when it is desirable to gather information on hardware choices. Performance was another consideration, as the prototype needed to provide relatively realistic behaviour.

A Linux kernel module provides an interface to the system, and interacts with a user-space daemon, which provides all processing and logic functionality. This is logically placed between the physical network device layer, and higher layers of the network stack. This results in the SAHN software appearing to higher layers, such as TCP/IP, as an ordinary physical device.

The presented architecture offers portability across platforms, transparency to higher layers, modularity and extensibility. Performance profiling of the developed implementation reveals that the SAHN prototype implementation provides reasonable performance.

As the implementation resides in user-space, de-coupled from operating system specifics, it is independent of operating system-level details, and can be moved to other operating systems with relative ease. Additionally, future changes to the kernel will have a minimal impact on the SAHN implementation: a large advantage. The architecture makes use of widely supported POSIX libraries, and thus requires little modification to be used on other operating systems.

The design provides similar interfaces to the IEEE802.11 and IEEE802.3 Linux implementations, thereby providing complete protocol transparency to higher level protocols. The architecture is highly modular, and is easily extensible.

The format of this paper is as follows. The 'Sample Architecture' section shall describe the proposed architecture of the SAHN system. The 'Implementation' section shall discuss a few implementation details. 'Testing and Results' will discuss the testing and profiling procedures, and examine the implications of the results. Finally, the conclusion will summarise the presented material and discuss some potential further research directions.

SAMPLE ARCHITECTURE

The functionality of the SAHN is logically broken up into five separate modules (Figure 4), offering routing, access control, flow control and error management, security, and Quality of Service (QoS) support.

The five modules are defined below.

Suburban Ad-Hoc Network Routing (SAHNR)

An ad-hoc routing mechanism that maintains tables of routes to all known nodes for each network node. For the purposes of this implementation, SAHN Routing uses concepts from Dynamic Source Routing (DSR) (Johnson, ,Maltz, 1996) and Ad-Hoc On-Demand Distance Vector Routing (AODV) (Perkins, Royer 1999), with modifications and optimisations to suit a 'quasi-static' suburban environment (Makalic 2001, Islam, Pose, Kopp 2003a).

Access Control/Management

A decentralised system for authenticating joining nodes, and managing access to network resources. The access control/management system is primarily built upon ideas from the Walnut kernel, a password-capability based operating system (Castro 1996)), which operates by granting privileges by passing tokens subsequently used to gain access to resources. It permits nodes to propagate and revoke privileged access to network resources in an entirely decentralised and robust fashion.

Flow Control/Error Management

A sliding-window based mechanism to control transfer rates, with an ACK/NACK-based system for management of transmission errors, and provisions for out-of-order packet delivery. This involves keeping buffers of transmitted or received packets, and transmission/reception counters for synchronisation. Communication of local counter values keeps sender and receiver synchronised.

Quality of Service

A system to provide a guarantee of certain link characteristics. Provides a mechanism to request and reserve bandwidth for a variety of critical applications, such as video/audio conferencing, or multimedia streaming. Interoperates with IPv6 and ATM's QoS systems, by way of providing a similar interface, with similar functionality. This involves supporting similar system calls on connection handles, via functions such as *setsockopt()*.

Security

An encryption mechanism, both end-to-end, and node-to-node, implemented at the Network layer. This ensures the safety of sensitive data within the network, as well as securing network management data, which could potentially be used by hostile outside parties to break into the network.

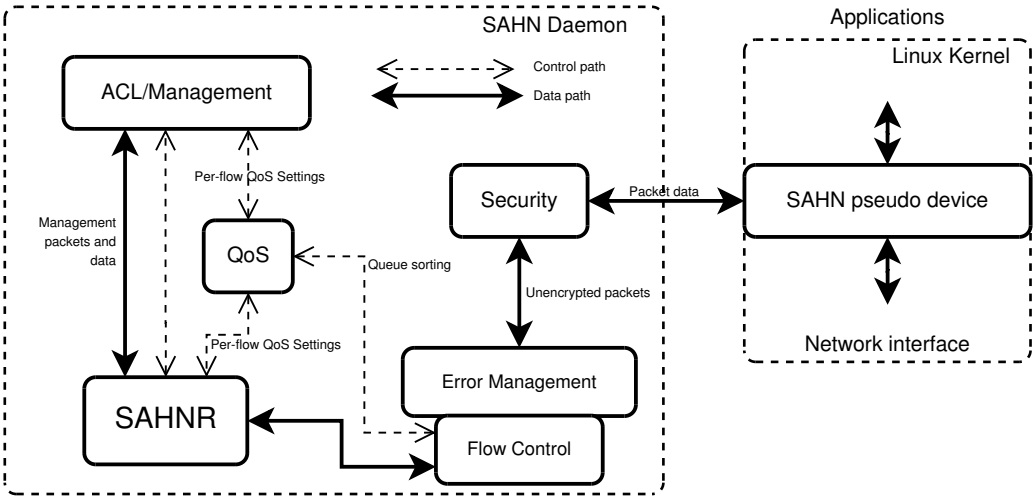

Figure 4: Single user-space process model

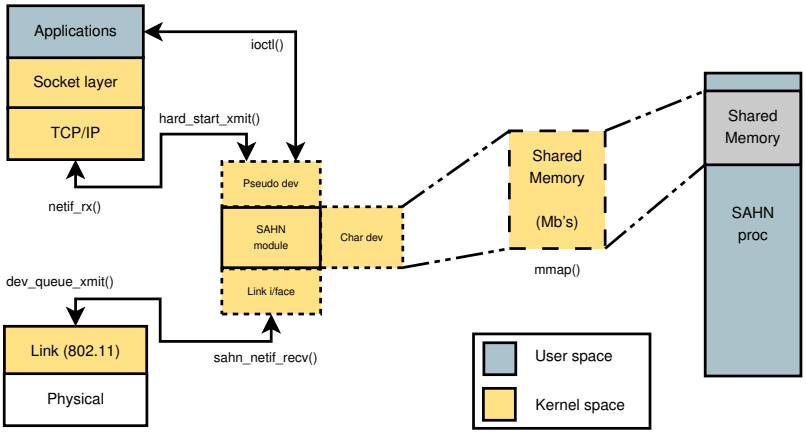

Figure 5: System overview

The SAHN implementation presented takes the form of a user-space daemon and kernel module (Figure 5), communicating via a character device (such as */dev/sahn0*), mapped to a segment of the SAHN daemon's address space via the *mmap()* system call. A similar concept, using shared memory, was introduced by (Makalic 2001, Bickerstaffe 2001, Garic 2001).

A pseudo network device is employed as the primary interface to the SAHN driver. This appears as a standard network device to the rest of the system (named, for example, *sahn0*).

The SAHN daemon handles all SAHN-related activity, and communicates with a real network interface, such as an IEEE802.11 device, via the kernel module, through the shared memory area. All packets are placed within this shared area, and control is transferred between the kernel module and the daemon. See Figure 6 for a representation of data flow through the system.

A consequence of having a user-space process responsible for SAHN operations is that a large number of context switches will take place in times of high activity. This represents a potentially serious performance bottleneck, with the potential to bring down the speed of the entire system. However, as wireless interface speeds are relatively low, due to the limitations of the medium, the quantity of data to be processed in any given time interval will be small enough for this bottleneck to have minimal impact.

Thus, although complexity is increased, negative effects are minimal, while the architecture provides a highly portable and easily maintainable implementation.

A custom kernel module is used to provide maximum interaction with the system, typically through the *ioctl()* system call. This allows the implementation to support configuration of the QoS or Access Control subsystems from user-space.

IMPLEMENTATION

The SAHN kernel module encapsulates all operating system-specific elements of the SAHN system, and presents an interface to the SAHN daemon. The module drives a series of character devices, which are file-like structures that support a variety of operations upon them, such as reading and writing. The module also controls virtual network devices, each of which is associated with a character device. In addition, the module controls a monitoring interface, which drives a set of items within the */proc* file system, which can be read for statistics.

The SAHN module provides the interface to the operating system, thereby de-coupling the SAHN software from operating system-specifics. A character device forms the interface to the SAHN daemon. Multiple SAHN daemons can run concurrently, each with a separate character device, and a separate virtual network device.

As soon as a character device is opened, which occurs during SAHN daemon start-up, the module performs the following tasks:

- A virtual network device is allocated and registered.
- Associated structures are initialised, including the transmission and reception queues.
- A large amount of memory is allocated, ready to be mapped into the daemon's memory space via the *mmap()* system call.
- The buffer space is initialised, ready for packet space allocations.

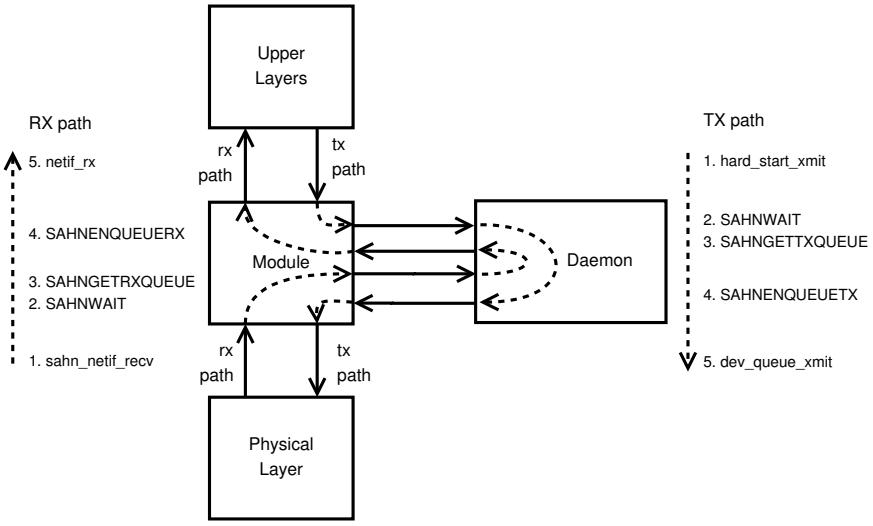

Figure 6: Data paths

The daemon typically then maps the shared memory into its address space by way of the *mmap()* system call on the character device, and begins node start-up procedures.

The SAHN system allows for an arbitrary number of SAHN devices to be used concurrently. When the SAHN daemon is started, it connects to the kernel module via the character device, and causes a SAHN device to be registered. Each device is numbered incrementally: *sahn0*, *sahn1*, *sahn2*, up to the maximum allowed number of devices.

To create a new interface, administrators launch a new SAHN daemon process, which will attach to the first available SAHN character device, and begin operating the newly created virtual network device.

The SAHN daemon does all the work of the SAHN, and interfaces with the SAHN module. Upon start-up, the daemon identifies a free SAHN character device, maps the shared memory into its memory space with the *mmap()* call, and begins performing management functions and processing packets.

Each SAHN device has an associated pool of physical network devices, through which the SAHN node operates. This pool is implemented as a linked list of Linux network device structures, which is manipulated by the SAHN daemon via *ioctl()* system calls to add or remove a device. On node start-up, the daemon specifies which physical devices are to be used for the relevant SAHN device, and the SAHN module begins monitoring the given devices. When an incoming packet arrives on a physical network device, the kernel module identifies the associated SAHN device, and uses its structures to en-queue the packet.

The daemon main event loop comprises the following steps:

1. Wait for activity.
2. Acquire the active queue segment. This locks off the queue segment that contains packets, and returns a reference to the beginning and end of this segment.
3. Sort the queue segment in order of priority (applying QoS and traffic shaping).
4. Process each packet in queue segment, possibly submitting for transmission or reception, passing upwards to the Linux network stack.
5. Release the queue segment.

This loop operates while the SAHN node is online. When packets are awaiting processing, they are passed into the daemon, and ultimately passed back for transmission or reception. To ensure fairness while processing, so that large volumes of outgoing packets do not dominate the processing of incoming packets, or vice versa, incoming/outgoing packets are processed in small lots, and control is returned to the loop which processes the next queue. Thus, if a large number of outgoing packets are awaiting processing, the incoming packets will still be processed in a timely fashion.

In order to ensure that the implementation does not rely on any packet structure, only the MAC header is used for obtaining addressing information.

Consequently, SAHN addresses are mapped to Ethernet addresses, and vice versa. This is accomplished by simply converting the three-byte SAHN address to a six-byte Ethernet address by padding the Ethernet address field with zeros, and copying the SAHN address into the last three bytes of the Ethernet address. IP and other higher network layers see only an Ethernet address.

In order to achieve such functionality, a cached mapping between SAHN addresses and MAC addresses has to be maintained. This permits outgoing SAHN addresses to be resolved to MAC addresses for physical addressing. Without the availability of such mapping, packets could not be received by destination hardware.

This mapping is achieved through the 'neighbour discovery' process, whereby an entering SAHN node broadcasts a 'hello' packet to its neighbours, and in turn get a 'hello' reply from each neighbour. Incoming 'hello' or 'hello reply' packets from neighbouring nodes have a physical MAC header, which contains the node MAC address. This address is stored, in combination with the node's SAHN address, within a database for future lookups.

TESTING AND RESULTS

Testing and performance profiling of the SAHN prototype implementation was performed using a series of 200MHz Pentium 1 PCs, connected in a chain of three nodes. The machines used each had three network cards. The first network card of every system was connected to a common channel, used for administration and baseline testing, and configured for a 192.168.1.x network. The second and third network cards connected to the nearest neighbours in each direction. These network cards were configured for a 192.168.2.x and 192.168.3.x network, respectively.

Performance profiling of the SAHN software reveals that even on the obsolete SAHN test bed hardware, performance of the SAHN implementation is relatively high. In throughput tests over a single hop, the SAHN offered performance in the order of 5.87 Mbit/second. This represents 82% of the speed obtained in the baseline tests, using the common channel instead of the SAHN link. In throughput tests over two hops, the SAHN implementation gave 5.59 Mbit/second, 74% of the baseline test speed.

Latency tests revealed significant differences in the latencies of the 'baseline' common channel and the SAHN link. *Ping* tests over a single hop revealed latencies over the SAHN link in the order of 3.7 ms, over 630% of the latency of the baseline test. Tests over two hops revealed 4.5 ms latencies, 770% of the baseline test.

Such large differences in link latencies are likely to be due to the paths that packet data take through the SAHN software, traversing several queues and undergoing processing within the SAHN daemon. On more modern hardware than the SAHN test bed, it is expected that these latencies would be significantly lower.

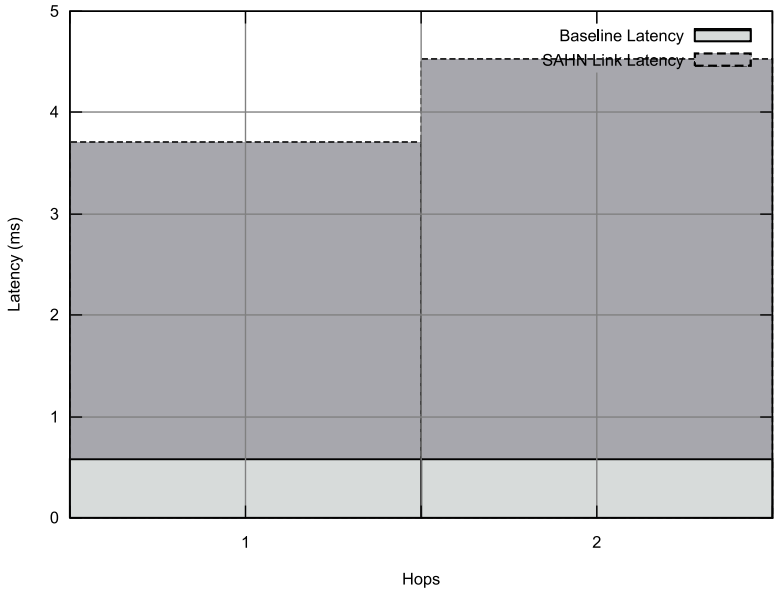

Figure 7: Latency test results

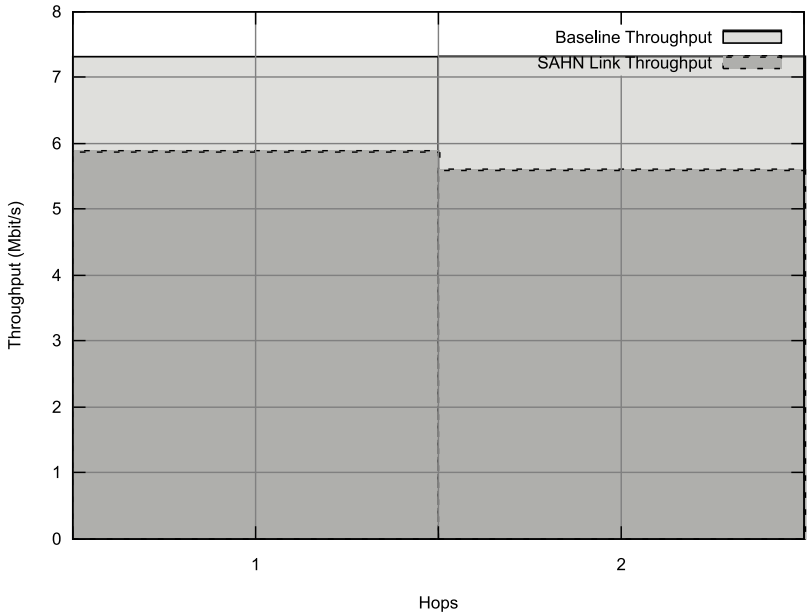

Figure 8: Throughput test results

These test results are promising, and reveal that the SAHN software is indeed capable of offering near-link speed performance. Due to the obsolete hardware comprising the SAHN test bed, throughput results are significantly lower than would be experienced on more modern hardware. Processor speeds of the test bed are low, and the available RAM severely limits the buffer size of the SAHN software. On more modern hardware, throughput is expected to closely approach link speeds.

Further testing would be desirable to gain a firmer understanding of the prototype implementation's performance. Such tests could typically involve more complex arrangements of nodes. A longer chain of nodes would give a better idea of the throughput and latency of the prototype, as well as experimentation with different lengths of chains (different hop counts). Varied topologies could be used to test route discovery performance.

Finally, experimenting with different hardware for the prototype nodes would yield more information about the processing performance of the prototype.

CONCLUSION

The presented Monash Suburban Ad-Hoc Network (SAHN) provides a highly secure and survivable broadband alternative to co-operating users in a suburban environment. The SAHN can aggregate many different physical media, and offers a broadband-class connectivity solution with low ongoing costs.

This paper has presented a sample architecture and prototype implementation of the SAHN, as a Linux kernel module and a daemon application written in C.

The SAHN kernel module provides an interface to the operating system kernel, and provides for the de-coupling of operating system functionality from SAHN functionality. For interaction between the SAHN daemon and the SAHN kernel module, a shared memory interface is defined. Accompanying this interface, a number of control routines are provided via the *ioctl()* system call interface.

Protocol transparency is achieved through emulation of a physical network device, and the use of MAC address caching and mapping to SAHN addresses. This provides a mechanism whereby SAHN addresses are represented as MAC addresses, so that standard interfaces and routines can be used with the SAHN implementation. Consequently, the SAHN implementation can be used with arbitrary upper level protocols, such as TCP/IP.

Possible future research directions involve further implementation of the SAHN software presented in this paper. The remaining SAHN modules need to be completed within the implementation: Quality of Service, security, management/access control, and error management. Additionally, further development of the SAHN is possible, to improve its security and access control systems, and to extend its use to more diverse environments.

REFERENCES

Anderson, M., Pose, R., and Wallace, C. S. (1996). A Password-Capability System. The Computer Journal, **29**(1): 1-8.

Bickerstaffe, A. (2001) Suburban Area Networks: Access Control and Management. Computer Science honours thesis, Monash Uni. Available at http://www.csse.monash.edu.au/research/san/. Last accessed March 4 2005.

Castro, M. D. (1996) The Walnut Kernel: A Password-Capability Based Operating System. PhD thesis, Department of Computer Science, Monash University.

Garic, S. (2001) Suburban Area Networks: Security. Computer Science honours thesis, Monash Uni. Available at http://www.csse.monash.edu.au/research/san/. Last accessed March 4 2005.

Gunawan, S. (2003) Resource Charging in Ad-hoc Networks by Password Capabilities. Computer Science honours thesis, Monash Uni. Available at http://www.csse.monash.edu.au/research/san/. Last accessed April 9 2005.

M. M. Islam. (2006) Design and Analysis of Low Level Protocols for Suburban Ad-Hoc Networks (SAHNs). PhD thesis, Monash University.

Islam, M. M., Pose, R. and Kopp, C. (2003a) Efficient Routing in Suburban Ad-Hoc Networks (SAHN). The 2003 International Conference on Communications in Computing (CIC 2003). Monte Carlo Resort, Las Vegas, USA.

Islam, M. M., Pose, R. and Kopp, C. (2003b) Routing In Suburban Ad-Hoc Networks. The 2003 International Conference on Computer Science and its Applications (ICCSA 2003). San Diego, USA.

Islam, M. M., Pose, R. and Kopp, C. (2004a) A Link Layer Security Protocol for Suburban Ad-Hoc Networks. Australian Telecommunication Networks and Applications Conference (ATNAC 2004). Pages 174-177. Available at http://www.csse.monash.edu.au/research/san/A_Link_Layer_Security_Protocol_for_Suburban_Ad-Hoc_Networks.pdf. Last accessed April 4 2005.

Islam, M. M., Pose, R. and Kopp, C. (2004b) Multiple directional antennas in suburban ad-hoc networks. Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on, 2:385–389.

Islam, M. M., Pose, R. and Kopp, C. (2005) Link layer security for SAHN protocols. Third IEEE PerCom International Workshops on Pervasive Wireless Networking (PWN 2005). Pages 279-283. Available at http://www.csse.monash.edu.au/research/san/Link_Layer_Security_for_SAHN_Protocols.pdf. Last accessed October 4 2005.

Johnson, D. B. and Maltz, D. A. (1996) Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, Mobile Computing, volume 353. Kluwer Academic Publishers.

Kopp, C. (1997). An I/O and Stream Inter-Process Communications Library for a Password Capability System. Masters Thesis, Monash University, Clayton, Australia.

Makalic, E. (2001) Suburban Area Networks: Route Discovery and Maintenance. Computer Science honours thesis, Monash Uni. Available at http://www.csse.monash.edu.au/research/san/. Last accessed March 4 2005.

Perkins, C. E. and Royer, E. M.. (1999) Ad-hoc On-Demand Distance Vector Routing. Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications.

Pose, R. (2001). Password-Capabilities: Their Evolution from the Password-Capability System into Walnut and Beyond. Proceedings of the 6th Australasian Computer Systems Architecture Conference (ACSAC), vol. 23, pp 105-113.

Pose, R. and Kopp, C. (1998) Bypassing the home computing bottleneck: The suburban area network. 3rd Australasian Comp. Architecture Conf. (ACAC). Pages 87-100.

Tyson, M. (2005) The Implementation of a SAHN protocol on an Open Source Platform. Computer Science honours thesis, Monash Uni. Available at http://www.csse.monash.edu.au/research/san/. Last accessed November 22 2006.

Zhou, L., and Hass, Z. J. (1999) Securing Ad Hoc Networks. IEEE Networks, **13**(6): 24-30.

COPYRIGHT