

HIGH COST ELIMINATION METHOD FOR BEST CLASS PERMUTATION IN ACCESS LISTS

Faheem Bukhatwa

Department of Computer Science

University College Dublin

Belfield, Dublin 4, Ireland.

Communication email: faheemfb@gmail.com

ABSTRACT

As communication is greatly expanding and the number of users continues to increase, the number of attacks on the Internet is also increasing. All this places more pressure on packet classifiers and filters to provide more filtering and greater security at higher performance levels without causing a bottleneck. Organising the rules in access lists according to their class is a way of improving performance for access list-based packet filters. One problem with this method is the large number of different permutations of the rules' classes in an access list for each individual pattern of arriving packets. "The Packet-Rule Cost Weighing" method (PRCW) is proposed which obtains the best organisation of the classes of rules in an access list that will guarantee the best performance. This method is based on classifying the rules in the access list and obtaining the relative processing cost of each rule. By obtaining the necessary parameters for the access list and the packet stream, the average processing cost is calculated for all packets in the packet stream needed to pass through the access list in every possible permutation of the rules' classes. The access list class permutation that yields the lowest average packet processing time is the best permutation possible for the particular packet stream. The problem with this method is the large number of computations required to calculate the cost of each and every permutation. In this paper we propose a method, which will reduce the time required. The calculation is hugely reduced from calculating every permutation to calculating less than the number of arrangements of the classes taken two at a time. The new method is based on identifying and eliminating the high processing cost steps of filtering.

KEYWORDS

Packet filtering, access lists, packet classification, cost weighing, PRCW.

1. INTRODUCTION

More organisations and users are using the Internet, more data is being kept and exchanged on and through the Internet. More communication means more packets being exchanged. Communication devices with an operation based on packet filtering are facing more packets to filter than ever. On the other hand, more security attacks are experienced on the Internet. Higher security levels are needed in most communication systems. On access list-based packet filtering devices, this means longer access lists. An access list consists of the rules, which are used to test packets. More packets and more rules have a negative impact on access list-based packet filters. It means more packets are tested, and each single packet is tested by a longer list of rules. Should this trend continue as expected, then access list-based packet filters as known today will not be efficient enough to perform a high level of security at a higher load of communication with the same level of performance. Unless improvements are achieved or a change in approach is adopted in access list packet filtering, either performance or security will have to suffer.

Firewalls are used to protect networks, by being situated strategically at a single security screening station where the private network or the Intranet connects to the public Internet. Incoming or outgoing packets are filtered to *allow* or *deny* each packet access to the network. Firewalls are also used to isolate and protect sub-networks. A firewall is a computer, router or other communication device that filters access to the protected network (Schreiner, 1998). Cheswick & Bellovin (1994) define a firewall as a collection of components or a system that is placed between two networks and possesses the following properties:

- All traffic from inside to outside, and vice-versa, must pass through it.
- Only authorised traffic, as defined by the local security policy, is allowed to pass through it.
- The firewall itself is immune to penetration.

In packet filtering firewalls, packet filtering refers to the basic operation performed by the firewall to inspect the packet header, verifying any number of the fields in the packet header i.e. the IP address, the port or both and then accepting or rejecting the packet. Filtering can be applied to incoming or outgoing packets or both. Packet filtering is transparent to the users or independent of the user's knowledge or intervention. Habtamu (2000) states that firewalls of this type are cheap, simple, fast, efficient and provide a good level of security. Packet filtering has proved to be efficient and effective at improving system security (Schuba & Spafford, 1997). A single access list rule can help protect an entire network by prohibiting connections between specific Internet sources and internal computers. Packet filters do not require client computers to be specifically configured; the packet filters do all of the work. But the cost of filtering involved may still be a significant bottleneck when a lot of inspections need to be performed on many packets (Ballew, 1997).

2. BACKGROUND OF REORGANISING ACCESS LIST RULES

All implementations of access list-based firewalls comprise a list of rules that are applied to inspect the incoming or outgoing packets. The ultimate aim is to reduce to a minimum the processing time needed for each packet to be inspected, with a reasonable cost in memory requirements.

The “critical rule” for a packet is defined as “the rule in an access list that identifies the packet so that some action is taken, such as accept or refuse the packet”. There is a processing time cost added every time a packet, going through a list is inspected by an individual rule until the critical rule is found (if ever). From a performance point of view, it is desirable that each packet meets that critical rule at or near the start of the list. This would reduce the time required for packet inspection and consequently improve performance. If the critical rule for a packet is met at or near the end of the list, or the rule is never met, the processing time for the packet will be very high and performance will degrade. The concept of rearranging the access list rules is to have most packets meet their critical rule at or near the start of the list of rules.

It is feasible to classify access list rules into different classes based on what fields the rules inspect in a packet (Bukhatwa, 2003). For example, the rules can be divided into those that are intended to filter packets by inspecting a single field such as the source address, target address or port number or inspecting any combination of these fields. Ultimately, access list rules are first classified into different classes and secondly, the rules are arranged in the list in such a way that rules of a particular class are grouped and placed in some order in relation to other classes. The number of classes within a list is based on 8 fields in the packets, which can be inspected. Other studies have shown that individual access lists on the Web use only three combinations, one field, three and four fields combinations. The maximum number of unique combinations for one given number of fields (k) selected from any one given available number of fields (n) can be calculated as:

$$c_n^k = \frac{n!}{k!(n-k)!}$$

Where c is the maximum number of combinations
 n is the number of fields to select from.
 k is the size of each combination.

It can be shown that the maximum number of unique combinations (c) that can be obtained using 8 fields (n) to select from in order to extract 1, 3 or 4 fields (k) is 134.

Consider that rules are ordered in a particular way such that all rules of a certain class, let us say class “X” are made to be located at the top of the access list. When packets start arriving, and if all arriving packets or most of them require the critical rule for acceptance or refusal to be of class “X”, then our ordering of the rules list was a successful choice and performance is expected to improve, see the right side of Figure 1. The

reason is that all or most packets need only a few rules to be checked at the start of the list before their relevant critical rule is reached. While, if the list was ordered in such a way that “X” class rules are placed at the end of the list, see the left-hand side of Figure 1; then for each packet arriving, most of the rules in the list have to be checked before eventually reaching the needed “X” class rule at the end of the list. Remember, each rule used to inspect a packet has a time cost attached to it, and the more rules the packet has to pass through the more time is wasted. The aim is to reduce this time required for processing each packet.

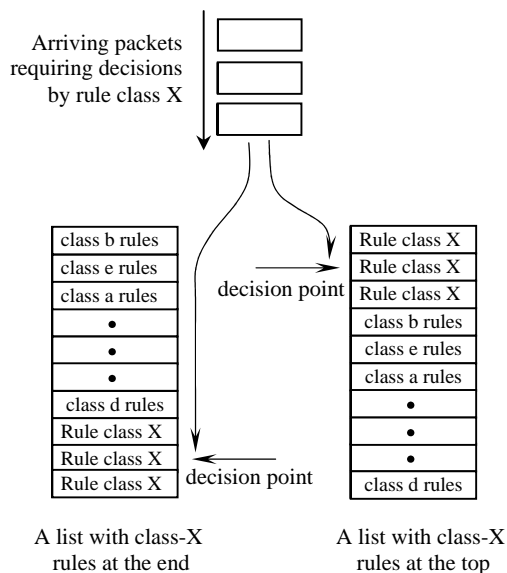


Figure 1, Effects of class arrangement of the rules in a list on processing time for arriving packets.

In order to arrange the access list in some order, the characteristics of the arriving packets must be known in advance. A profile of arriving packets can be developed from past observations for a network device, and it is possible to make a prediction about the expected pattern of these packets in the future. Profile development is possible and is easily established (Gupta & McKewon, 2001). Classification of packets arriving in the past can help to determine expected classes and numbers of packets arriving in the future. On the other hand, real time pattern recognition can be done based on classifying arriving packets. Packet patterns can periodically be inspected to determine the most effective order of the rules list.

3. PACKET-RULES COST WEIGHT METHOD (PRCW)

Consider a packet arriving into an access list. It is not entirely true to say that the total processing time for such a packet is only dependent on the number of rules. The processing time of each rule must be taken into consideration for a more accurate calculation. An arriving packet getting filtered by 5 rules, which require 1 microsecond each, will have the same processing time if it was being filtered by 10 rules requiring ½ microsecond each. Therefore, the processing time of a packet depends on the number of rules as well as the processing time for each rule (Bukhatwa, 2004).

Consider a packet passing through a list of rules. If there were m rules of one class in the list with a processing time (or an average processing time) r for each rule, then the total cost of processing C , for a single packet passing through the rules of a particular class in the list would be calculated as: $C = m \cdot r$. If the packet has to pass through more rules belonging to different classes (s) then the total cost can be expressed as:

$$c_{total} = \sum_{s=1}^n m_s \cdot r_s \quad \text{where } n \text{ is the number of classes}$$

But, for the class that contains the critical rule for the packet, processing is different. That is where the rules' class is the same as the packet's class. The critical rule may be located as the first rule in the class, last or somewhere in the middle. Let us assume that on average the critical rule will be in the middle of the class's rules. This means processing time for the class, which will accept or refuse the packet, will be presented as: $C = \frac{1}{2} m \cdot r$

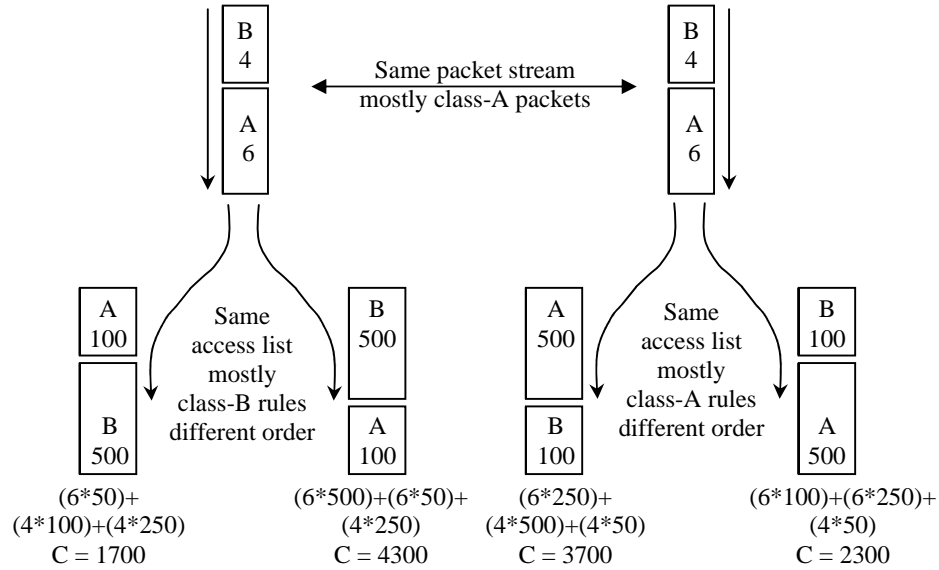


Figure 2, The same packet stream arriving into two different access lists (left and right), each is organized in two different arrangements.

Let us consider an example of a packet stream and an access list both having only two classes A and B. The packet stream consists of only 10 packets. It is mostly a class-A stream, 6 packets of class-A and 4 packets of class-B. For the access list, the number of rules and processing time of each has been taken into consideration to produce a total cost ($m_s \cdot r_s$) for each of the classes A and B. The left-hand side of Figure 2 shows the list with the cost for classes A and B as 100 and 500 units of time respectively, while on the right-hand side of the figure, the cost for classes A and B is reversed to 500 and 100 respectively. Based on what we have seen previously, to obtain the better performance it is best to place rules of class A on top of the list because there are more class A packets in the packet stream. With class A rules on top of the list, remember that class A packets passing through the list will only go through class A rules and will not get to class B rules, while class B packets will be processed by class A rules followed by class B rules. In other words, every packet will continue processing until it is processed by rules of its class. The example on the left-hand side shows that the total cost is less if class A rules are placed on top of the list ($C=1700$). While on the right-hand side of Figure 2, the total processing cost is less when class A rules are placed at the bottom of the list ($C=2300$).

4. TOTAL COST CALCULATION FOR A PACKET STREAM THROUGH AN ACCESS LIST

It is suggested that it is possible to calculate the total cost for all the different possibilities of class combinations of an access list for a given packet stream (Bukhatwa, 2003). Combinations of the rules classes

in this sense are best referred to as the permutations of the access list classes. Consider a packet stream made up of 3 classes of packets (p_1 , p_2 and p_3) and an access list of 3 classes of rules with a calculated cost of (r_1 , r_2 and r_3). Notice that r_x now refers to the cost for all the rules in a class rather than the cost of one rule. The maximum number of permutations for the 3 access list classes would be $3! = 6$ different possibilities. The best permutation for our purpose; would be the one that would produce the lowest processing time (or least cost) for a given packet stream. The cost for each permutation is the total cost for each class of packets passing through the list, as was seen in Figure 2. The cost can be calculated as follows with reference to Figure 3 and the following formulae. Remember a p_1 class packet is processed through the rules and will stop at rule class r_1 . Similarly, a p_2 packet will stop at r_2 and a p_3 will stop at r_3 . Also, $C(r_1, r_2, r_3)$ refers to the cost when the rules classes are arranged with r_1 class on top, followed by r_2 and then r_3 .

$$c_{(r_1, r_2, r_3)} = 0.5 p_1 r_1 + p_2 r_1 + 0.5 p_2 r_2 + p_3 r_1 + p_3 r_2 + 0.5 p_3 r_3$$

$$c_{(r_1, r_3, r_2)} = 0.5 p_1 r_1 + p_2 r_1 + p_2 r_3 + 0.5 p_2 r_2 + p_3 r_1 + 0.5 p_3 r_3$$

$$c_{(r_2, r_1, r_3)} = p_1 r_2 + 0.5 p_1 r_1 + 0.5 p_2 r_2 + p_3 r_2 + p_3 r_1 + 0.5 p_3 r_3$$

$$c_{(r_2, r_3, r_1)} = p_1 r_2 + p_1 r_3 + 0.5 p_1 r_1 + 0.5 p_2 r_2 + p_3 r_2 + 0.5 p_3 r_3$$

$$c_{(r_3, r_1, r_2)} = p_1 r_3 + 0.5 p_1 r_1 + p_2 r_3 + p_2 r_1 + 0.5 p_2 r_2 + 0.5 p_3 r_3$$

$$c_{(r_3, r_2, r_1)} = p_1 r_3 + p_1 r_2 + 0.5 p_1 r_1 + p_2 r_3 + 0.5 p_2 r_2 + 0.5 p_3 r_3$$

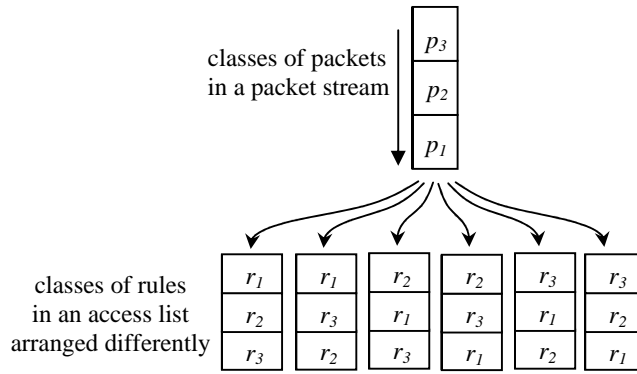


Figure 3, Calculation of cost for all different permutations.

The above formulae can be simplified to extract a general format that calculates the cost for any single permutation by subtracting the cost values appearing in all of them: $p_1 \frac{1}{2} r_1$, $p_2 \frac{1}{2} r_2$ and $p_3 \frac{1}{2} r_3$. As each value will be taken from every formula, they will have no effect on the final comparison to find the formula with the smallest value. Now, the general formula to calculate the cost for any number of classes of rules and packet streams can be expressed as follows:

$$\text{The time cost for a permutation } t \text{ of rules classes} = \sum_{j=1}^{n-1} \left(r_{tj} \cdot \sum_{z=j+1}^n p_z \right)$$

- Where
- n is the number of classes.
 - r_{tj} is the time cost for rule class in position j in the combination t .
 - p_z is the number of packets of class z .

The PRWC algorithm may be expressed in two parts. The first part produces all the different permutations possible for the access list classes. This is done by recursively shifting the numbers that make up the classes. The second part of the algorithm actually calculates the total cost of processing the packets through the access list using each of those permutations. For this purpose two more lists of information are needed, the first is the number of packets in each class in the tested packet stream. The second is the total processing cost of rules for each class in the list. This processing cost is simplification of two values namely the number of rules in the class multiplied by the average processing cost of the rules in that class. The actual algorithms are outlined in detail in Bukhatwa (2003). This algorithm works in producing the permutation of classes in an access list which yields the lowest processing cost for a given packet stream. The major problem is the amount of processing and length of time required to find such permutation. In fact the cost calculation for every permutation needs to be performed. A 16-class access list can have more than 2×10^{13} different permutations. A 30-class access list will have over 2×10^{32} .

5. THE HIGH COST ELIMINATION IMPROVEMENT TO PRWC

The new approach is based on the elimination of unnecessary processing of packet stream classes with large numbers of packets by the access list classes with high cost rules. We shall call it the High Cost Elimination method (HCE). Assuming the largest number of packets in a packet stream belong to class B, and that the highest cost class of rules in an access list are of class A. This means that of all the processing of the different classes, the highest processing cost is done when packets of class B are processed by rules of class A. It is also a fact that this processing is an extra cost of no value to the filtering process; we know that class B packets are only filtered by class B rules. Ideally, it is not desirable for class B packets to reach class A rules in the list. Therefore, class B rules must be located before class A rules in the access list. That is provided that on the other hand, packets of class A meeting class B rules in the list would have less unnecessary processing cost. In such a case, it is better for class A rules to come before class B rules in the list saving this unnecessary processing from ever occurring.

The following algorithm can implement the High Cost Elimination method (HCE):

- 1) The classes in the packet stream are arranged in a descending order based on the number of packets in each class.
- 2) The access list classes are arranged in a descending order based on the cost weight of the rules in each class.
- 3) Each of the packet stream classes is checked by calculating the cost of filtering with each of the classes in the access list. Two cost calculations are made for a given packet class number and a rule class number. In other words, cost is calculated when packets of class A is filtered by rules of class B. Then they are swapped so that packets of class B are filtered by rules of class A and the cost is calculated.
- 4) The number of times are noted when a higher cost value is calculated for a class when it is used as a packet stream class than when it is an access list class. The number of times will indicate how high up the class will move in the desired access list desired permutation.

To describe this algorithm using an example, assume a 4-class packet stream containing the following number of packets: 10, 30, 40 and 10 for classes 1, 2, 3 and 4. Also a 4-class access list containing the following rules' cost values: 8, 2, 4 and 10 for classes 1, 2, 3 and 4 consecutively. Table 1 shows the packet stream classes ordered based on the number of packets in each class; and also shows the access list classes ordered based on the costing of rules in each class. Taking the packet stream classes one at a time and calculating the cost against each of the access list classes, the left part of Table 2, shows the packet stream classes and the cost calculations with each of the remaining classes of the access list. In the right side of the table, the roles of packets and access list classes are reversed. For example, packet class 3 filtered by rules class 4 becomes packet class 4 filtered by rules class 3, and the cost is calculated. When the two cost columns in the table are compared; the higher values of the cost in the left side of the table are noted and marked in the column [*position in list*] with the (*) sign. The number of (*) signs for a given class indicates the index of its place in the list. The higher the index number is; the higher the position of the relevant class in the list. A

value of 0 indicates that the class is placed at the bottom of the access list. The position refers to the class shown in the [Rules class] column on the right side of the table. In the example in Table 2, the permutation obtained for the access list class is: class 2, 3, 1, 4. Class 2 being on top of the list and class 4 at the bottom.

Table 1, The ordered packet stream and access list classes.

Packet stream			Access list	
Packet classes	Number of packets		Access list classes	Number of rules
3	40		4	10
2	30		1	8
1	10		3	4
4	10		2	2

When performing the cost calculations, no calculation needs to be done if the packet class and rules class are the same. On the other hand, classes of rules in an access list for which there are no matching classes of packets in the packet stream shall be ignored, for they will be placed at the bottom of the list. They would unfortunately be extra luggage carried on the access list, but will have no effect on the arriving packets with one exception. The exception is those classes of packets, which do not have matching classes in the list. Such class of packets will be ignored because their cost is going to be constant for the same access list regardless of the rules permutation. A packet of such a class will simply pass through each and every rule in the list all the time. With a better prediction of the packet stream, such a problem can be reduced or eliminated.

Table 2, Cost calculations and defining the position in the access list order.

Packet class into access list class				Position in list	Reversed roll classes			
Packet class	Rules class	Cost calculation	cost		Rules class	Packet class	Cost calculation	cost
3	4	40 x 10	400	*	3	4	04 x 10	040
3	1	40 x 08	320	* 2	3	1	04 x 10	040
3	2	40 x 02	080		3	2	04 x 30	120
2	4	30 x 10	300	*	2	4	02 x 10	020
2	1	30 x 08	240	* 3	2	1	02 x 10	020
2	3	30 x 04	120	*	2	3	02 x 40	080
1	4	10 x 10	100	*	1	4	08 x 10	080
1	3	10 x 04	040	1	1	3	08 x 40	320
1	2	10 x 02	020		1	2	08 x 30	240
4	1	10 x 08	080	0	4	1	10 x 10	100
4	3	10 x 04	040		4	3	10 x 40	400
4	2	10 x 02	020		4	2	10 x 30	300

One more reduction is for the calculations of processing time for any packet class to be filtered by the same class rules in the list. This would be the same value for all different permutations of the access list when filtering the same packet stream. Eliminating that from all permutations will not affect the final output.

6. CONCLUSION

Reorganising access list rules based on their classification is one way of improving performance particularly where there are a large number of rules. Arrangement of rules is only relevant to one particular characteristic of an incoming stream of packets. Finding out the best arrangement of the classes in an access list for a particular packet stream means calculation of processing costs of the packets through all possible permutations of the access list classes. The Packet-Rule Cost Weighing (PRCW) method was proposed in a previous research work by (Bukhatwa 2004) to perform the calculation and produce the permutation that will

yield the best (lowest) processing cost. The PRCW method is particularly helpful for automating the solution when considering that the number of possible permutations of the classes in the access list can be extremely large. An access list with n -classes of 16 and has $n!$ or more than $2 \cdot 10^{13}$ different permutations. Each permutation contains 136 calculations (the sum of all the numbers from n to 1). The High Cost Elimination (HCE) new method would only have $n \cdot (n-1) = 16 \cdot 15 = 240$ permutations of two elements. Each permutation has only two calculations.

Yet, the drawback with the PRWC algorithm is in that it produces the cost for each and every permutation possible for a particular characteristic of a packet stream. Though, it removes guessing or randomly testing the different possibilities, the algorithm still carries out a large number of computations. The High Cost Elimination (HCE) method is a new improvement of the same algorithm (PRWC) and will limit the calculations to a fraction of that needed by the original algorithm. This new improved method has been tested and verified in tests against the original PRCW algorithm.

REFERENCES

- Ballew, Scott M. 1997. *Managing IP Networks with Cisco routers*. O'Reilly, 1st edition. October 1997.
- Bukhatwa, F., 2004. *Packet-Rule Cost Weighting Method for Best Organisation of Access Lists in Packet Filtering*. The 2004 International Conference on Computers, Communication and Control Technologies CCCT'04, August 14-17, 2004 - Austin, Texas, USA.
- Bukhatwa, F., and Patel, A., 2003. *Effects of Ordered Access Lists in Firewalls*, paper accepted at the IADIS International Conference WWW/Internet 2003 WWW/Internet 2003, Algarve, Portugal, 5-8 November 2003, (<http://www.iadis.org/icwi2003/>)
- Cheswick, W.R. and Bellovin, S.M. 1994. *Firewalls and Internet Security, Repelling the Wily Hacker*, Addison-Wesley.
- Gupta, P., and McKewon, N., 2001. *Algorithms for Packet Classification* IEEE Network Special Issue, March/April 2001, vol. 15, no. 2, pp 24-32
- Habtamu, Abie, 2000. *An Overview of Firewall Technologies* Norwegian Computing Centre January 2000.
- Hazelhurst, S., 1999. *Algorithms for Analysing Firewall and Router Access Lists*", Technical Report TR-Wits-Cs-1999-5, Dept. of Computer Science, University of Witwatersand, South Africa July 1999.
- Hazelhurst, S., 2001. A proposal for Dynamic Access Lists for TCP/IP Packet Filtering. *SAICSIT 2001*. Pretoria, South Africa.
- Schreiner, R., 1998, *CORBA Firewall White Papers*, TechnoSec Ltd.
- Schuba, C.L. and Spafford, E.H., 1997, A Reference Model for Firewall Technology. *In Proceedings of the Thirteenth Annual Computer Security Applications Conference*, December 1997.