# Evaluating Layer 3 Device Tunneling and Access Control List Security Bandwidths Using B-Node Theory

**S P Maj, D Veal**

Edith Cowan University, Perth, Western Australia

**Summary**

Computer networks of today consist of a multitude of devices, technologies and protocols that each in themselves may impact the performance as experienced by the end user. A number of techniques exist to predict and/or model these variations, each with their own relative merit. Bandwidth-Nodes (B-Nodes) are one such technique that allows devices and/or technologies to be modelled as a single node or as a collection of nodes, each exhibiting their own performance characteristics. A simple formula is used to calculate the theoretical maximum bandwidth of a node which allows for efficiency decomposition. This incorporates device sub-optimal operation ($e_{Di}$) and using empirically derived results, $e_{Di}$ for an individual process on a particular device can be calculated. This paper focuses on evaluating the impact the choice of device will have on network performance. By empirically evaluating Access Control Lists (ACLs) with a varying number of statements, as well as assessing different tunneling techniques, the specific device efficiencies for these can be calculated. Using this information, the anticipated performance of an ACL network given a technical specification can be easily and quickly determined.

***Key words:***
*Network Security, B-Nodes, Access Control Lists, Tunneling.*

## 1. Introduction

The addition and subtraction of devices, technologies and protocols within a computer network may come at an additional cost, not only in terms of expenditure for extra system infrastructure, but also in terms of overheads which may reduce overall system performance. This is especially the case for complex security protocols ranging from simple Access Control Lists (ACLs) to the IPSec framework. These overheads can include additional information, such as "control packet" or "data packet" overheads [1] as well as those in the form of additional processing required by a device in order to achieve the desired network administration goal. A variety of methods exists to measure and/or predict this anticipated system performance variation. Such methods include [1]:

1. Rule-of-Thumb: a subjective method typically based on prior experience, usually with little mathematical rigor.
2. Stochastic modeling: the use of complex mathematics which can be problematic and difficult to understand by the typical network administrator, and
3. Benchmarking: commonly use different scales and units, with comparative results possibly requiring further interpretation.

B-Nodes are a method which uses the notion of a networks' bandwidth to provide an *"unbiased, empirical performance analysis that is simple to use and conceptualise and be based on the user perception of performance [1]"*. B-Nodes are a conceptually simple model that use the concept of abstraction to control the detail and hide the complexity of a system [2]. A simple formula is used to determine the anticipated performance which can be used for individual components or for networks as a whole [1]. Through the use of recursive decomposition, device sub-optimal operation can be assessed for a particular device and this information used in the B-Node formula to predict the anticipated performance of a network for given a configuration. This research focuses on empirically evaluating device sub-optimal operation for Access Control Lists and a variety of tunneling techniques, and calculating the $e_{Di}$ parameter of these for use in the B-Node formula.

## 2. Network Performance

There are a variety of terms, units and metrics that are employed to describe performance including delay, packet loss and bandwidth [3]. As each of these typically use differing units of measure, there are numerous metrics and measurement methodologies employed to express these quantities [1]. One such metric is bandwidth, and in a network centric context, it is defined as the data rate at which a network link or path can transfer information [4]. Specific application data, including any overheads incurred in its transportation, may impact the performance as perceived by the user i.e. the overall elapsed time an application takes to execute over the underlying network [5], and this in turn impacts the performance as it is perceived by the user [5]. Bandwidth Nodes (or B-Nodes), as described by Maj, Veal, et. al [2, 6-13], is a *bandwidth-centric concept used to describe and model the "performance of every node and data path …assessed by a*

Manuscript received September 5, 2010
Manuscript revised September 20, 2010

*simple, common measurement- bandwidth… with the common units of …Mbytes/s"* [7]. This bandwidth can be described using Bulk Transfer Capacity (BTC), which is defined as the long term average data rate over the path in question [5]. It is calculated by dividing the amount of data that was sent by the elapsed time it took to send this information [5], and this also uses the common units of Mbytes/s. Applications that rely on a networks' capacity to transfer significant quantities of data rely on this BTC. B-Node theory has been expanded by to account for possible reductions in efficiency due to additional network protocols and the associated overheads they may incur [1]. By decomposing the simple B-Node formula,

$$(B = C \text{ x } D \text{ x } E) \tag{1}$$

The "Efficiency" component (E) was shown to be a product of all efficiencies ($e_i$) contained within the B-Node, with the equation thus becoming:

$$B = C \times D \times \prod_{i=1}^{n} e_i \tag{2}$$

This accounted for Data Packet efficiencies (overheads directly added to packets that are transmitting application data) and Control Packet efficiencies (packets that carry no user data and are used to control link flow) [1]. Devices themselves can add latency and processing overhead to a system that may not be accounted for in equation 2. As such, the concept of device sub-optimal operation ($e_{Di}$) was introduced and the B-Node equation further modified to account for this [1].

$$B = C \times D \times \prod_{i=1}^{n} e_i e_{Di} \tag{3}$$

Empirically derived results must be evaluated for an individual process on a particular device [1]. From this, the specific device efficiency, $e_{Di}$, for a particular network protocol or technique can be evaluated, and the effect the device has on BTC can be determined.

## 3. Access Control Lists

Access Control Lists are a Layer 3 technique which potentially enhances security by preventing traffic that a network administrator defines as unnecessary, undesirable, or unauthorized to traverse a network or internetwork [14], [15], [16]. They are a sequential collection of permit or deny statements that are applied to addresses or upper layer protocols that instruct the device to block or forward packets based on source or destination address, TCP/UDP port numbers or combinations of these [17-19]. ACLs are defined on a per-protocol basis [17, 19] and can be applied per interface to inbound or outbound traffic, or both [18, 20]. They limit network traffic and can increase network performance, provide traffic flow control, and provide a basic level of security for network access [17]. Due to the sequential nature of the technology, the order of access list statements is significant as the first matching statement decides what to do with the packet [21]. If a packet is not matched against a statement, it will continue to be checked against every statement one after the other until a match is found, or until the end of the ACL. If no match is found, the packet is implicitly denied by the device [21]. Cisco Systems Inc rule of thumb is one ACL per interface per direction, as they consume CPU resources in the device as every packet has to be processed by the CPU [17]. Morrissey notes that ACLs may slow down your system [20], as does Davis who states that *"…typically your router uses net flow or fast processors for fast switching. When you use route filtering, you use the slowest mode of process switching…"* [19]. Morrissey also notes ACL support in hardware ASICs which can support true wire speed [20]. Davis also observes that *"Using access lists for route filtering is CPU intensive. Overuse of routing filtering can slow the flow of packets...[and] the router usually has but one processor available for process switching"* [19]. In the article entitled "The Cost of Security on Cisco Routers" it is stated that "*there are significant performance penalties once you enable ACLs … because an access list cannot always take advantage of the fastest switching technique that might otherwise be available on the router"* [22] . Veal et.al uses the concept of "Negative Bandwidth" to quantify the performance reduction after implementing ACLs [23]. Using ACLs for filtering is entirely CPU intensive [19], and as such there are no control packets or additional data packet overheads being sent on the wire to consume bandwidth. Control and data packets in this case are defined by Cikara et al [1]. As such, the Efficiency parameter pertaining to ACL statements will refer to device sub-optimal operation and the figure ($e_{Di}$) should be obtained for the statements on a specific device.

## 4. Tunneling

Tunneling is a technique which provides a method of encapsulating arbitrary packets inside of a transport protocol [24]. It is designed to provide the necessary services to implement any standard point-to-point encapsulation scheme, and as tunnels are point-to-point links, a separate tunnel must be configured for each link [24]. Tunnels have three primary components, those being [24]:

1. Passenger protocol- is the protocol that is to be encapsulated, such as Appletalk, DECNet, IP, IPX, etc

2. Carrier protocol- is the encapsulation protocol which may include Generic Route Encapsulation (GRE), Network Operating System (NOS), etc.
3. Transport Protocol- is the protocol used to carry the encapsulated protocol, which is IP only.

The advantages of using tunnels can include [24]:

1. Reductions in bandwidth, due to the directed nature of the technology
2. The provision of multiprotocol local networks over a single-protocol backbone
3. Alleviation of protocol limitations (such as limited hop counts, default routes and load balancing), and
4. Increased security through the use of virtual private networks across wide area networks

Tunnels do however come with disadvantages, and precaution must be used. Such considerations include [24]:

1. Encapsulation and decapsulation in general, is a processor switched operation, which are slow operations
2. Tunneling can violate security policy where the source and destination is not restricted.
3. Transport protocols with limited timers may create problems
4. Multiple point-to-point tunnels can saturate links with routing information
5. Routing protocols that make decisions based solely on hop count will often prefer a tunnel over multipoint real link (i.e. tunnels may appear to be "one hop"), and
6. The routing information of the tunnel network mixes with the transport network's information create a recursive routes (i.e. the best path to the tunnel is via the tunnel itself)

## 5. Tunneling Protocol Categories

A number of tunneling protocols exist that allow encapsulation of one protocol inside another. For the purposes of this paper and due to the limitations of the technology evaluated, in this case the Cisco IOS version, we will be focusing on IPv4-to-IPv4 and IPv6-to-IPv4 transition mechanisms.

## 6. IPv4-to-IPv4 Encapsulation Protocols

### *6.1 I P-over-IP Tunnel*

IP-over-IP tunnels are a method where an IP datagram is carried as the payload within an IP datagram and is defined by RFC 2003 [25]. This is achieved by the addition of an outer header before the existing IP header [25]. It is the outer header source and destination addresses which identify the endpoints of the tunnel, while the existing IP header identifies the original sender and recipient of the packet [25]. The additional 20 bytes of overhead incurred by the additional IPv4 header (which precedes the existing IPv4 header) impacts efficiency (table 1).

Table 1: IPv4-over-IPV4 encapsulation efficiency overhead

| Protocol | Overhead (B) | TCP/UDP Payload (B) | Efficiency |
|---|---|---|---|
| No Tunneling | | | |
| IPv4 TCP | 78 | 1460 | 94.93% |
| IPv4 UDP | 66 | 1472 | 95.71% |
| With IP-over-IP Tunneling | | | |
| IPv4 TCP | 98 | 1440 | 93.63% |
| IPv4 UDP | 86 | 1452 | 94.41% |

This encapsulation method translates to a reduction in efficiency of 1.3% for both TCP and UDP.

### *6.2 IP-over-IP KA9Q/NOS Compatible Tunnel*

The KA9Q Network Operating System (NOS) TCP/IP package was only the second known implementation of the Internet protocols for low-end computers which could simultaneously act as a network client, server and an IP packet router, including support for multiple client and server sessions [26, 27]. There are numerous implementation and variations of NOS including but not limited to the MS-DOS executable "net.exe" [28], WNOS, JNOS, PMNOS and TNOS [29]. Cisco provides an IPv4 to IPv4 NOS tunneling implementation which is compatible with the KA9Q program [30]. Although KA9Q still exists in various forms such as embedded versions of Linux [27], the Cisco technical implementation is difficult to ascertain, so as such for the purposes of this research it is assumed that NOS has the same overheads as IPv4 as defined by RFC 791 [31]. This would incur the same decrease in efficiency as IP-over-IP tunnels of 1.3%

### *6.3 IP-over-IP GRE Tunnel*

Generic Routing Encapsulation (GRE), defined by RFC 2784 [32], uses IP Protocol 47 and the Cisco implementation adds an additional 24 bytes of overhead to each packet [33, 34]. GRE tunnels were designed to be completely stateless, meaning that each tunnel end-point doesn't keep any information about the state or availability of the remote tunnel end-point [35]. However, new

implementations of the protocol allow for "keepalives" to allow interfaces to shut down if the keepalives fail for a certain period of time [36]. From and efficiency perspective, the additional 24 bytes of overhead decreases the efficiency of IPv4 TCP by 1.56% to 93.37%, and likewise by 1.56% to 94.15% for IPv4 UDP.

# 7. IPv6-to-IPv4 Encapsulation Protocols

## *7.1 Automatic 6to4 Tunnel*

6to4 tunnels are defined in RFC 3056 [37] and are an automatic tunneling method where tunnels are created dynamically on a per-packet basis [38] based on an embedded IPv4 address [39]. The tunnel endpoint is determined by the IPv4 address embedded in the 6to4 address and it is a combination of the of the unique routing prefix, 2002::/16 and the unique 32-bit IPv4 address [40]. The format of the packet is typically 2002:*Layer-3-IPv4-Address*::/48 and 16 bits following the IPv4 address can be used to number the networks within the site [38]. This type of tunnel is used for less-permanent, transient connectivity [40] and can be used in point-to-multipoint configurations [38]. Dual stack layer 3 devices are not configured in pairs as the IPv4 infrastructure is treated as a virtual non-broadcast multi-access (NBMA) link, where the IPv4 address embedded in the IPv6 address is used to find the other end of the automatic tunnel [38]. By intending to encapsulate an IPv6 packet and transport it over an IPv4 backbone, an additional 20 bytes of overhead must be added to each packet, which is the IPv4 header attached to the front of the IPv6 packet (table 2).

Table 2: Automatic 6-to-4 encapsulation efficiency overhead

| Protocol | Overhead (B) | TCP/UDP Payload (B) | Efficiency |
|---|---|---|---|
| No Tunneling | | | |
| IPv6 TCP | 98 | 1440 | 93.63% |
| IPv6 UDP | 86 | 1452 | 94.41% |
| With Automatic 6-to-4 Tunneling | | | |
| IPv6-to-IPv4 TCP | 118 | 1420 | 92.33% |
| IPv6-to-IPv4 UDP | 106 | 1432 | 93.11% |

Therefore the additional 20 bytes of overhead leads to a decrease of 1.3% for TCP and for UDP using this tunneling technique.

## *7.2 6to4 GRE Tunnel*

IPv6 over IPv4 GRE tunnels use the standard tunneling technique as defined is Section 0. They link two points in the standard point-to-point encapsulation scheme [38]. As indicated previously, the tunnels are not tied by any specific passenger or transport protocol, however in this case they carry IPv6 as the passenger protocol over GRE as the carrier protocol [38]. As the GRE encapsulation is an IPv4 encapsulation technique, it has the same overheads as defined in Section 0. In this case, it leads to a decrease of 1.56% to 92.07% for TCP, and 1.56% to 92.85% for UDP

## *7.3 IPv4 Compatible Tunnel*

IPv4 compatible tunnels use IPv4-compatible IPv6 addresses. This address comprises of a "*IPv6 unicast address that have zeros in the high order 96 bits of the address, and an IPv4 address in the low-order 32 bits"* It has the format "*0:0:0:0:0:0:A.B.C.D or ::A.B.C.D , where "A.B.C.D" represents the embedded IPv4 address"*. [38]. This method, as with Automatic 6to4 tunneling, adds an additional 20 bytes of overhead. Therefore the efficiencies are identical to table 2.

## *7.4 6to4 Manually Configured Tunnel*

As the name suggests and as defined in RFC 2893 [39, 41], manually configured tunnels is a technique where the IPv6 address is manually configured on the tunnel interface, and the IPv4 address is also manually configured on the interfaces of both the tunnel source and destination [40]. It requires definite specification of the IPv4 tunnel source and the IPv4 tunnel destination [42]. The tunneling technique requires a dual-stack node (IPv4 and IPv6) [39] and is generally used for dedicated, permanent connectivity [40]. The IPv6 packets are encapsulated within IPv4 packets, with IPv6 being the passenger protocol and IPv4 the encapsulating protocol [43]. As such, the additional overhead of this technique is 20 bytes. This translates to efficiencies as identified in table 2.

# 8. Empirical Validation

### 8.1 Access Control List Empirical Validation

As ACL statements are a CPU intensive process that does not produce any control packet or data packet overheads, they rely on the processing capacity of the device to filter and apply the appropriate measures on the packet being processed (ie discarding or permitting the packet to pass). Hence, ACL statements are a device specific efficiency ($e_{Di}$) and as differing devices have different overheads, and the efficiencies exhibited by each may vary. An experiment to calculate specific device efficiencies for a varying number of ACL statements was conducted and arranged as in Figure 1. The router was used in its default configuration (with CEF enabled), and ACL statements were first applied and tested on the ingress interface for both TCP and UDP data streams. Then, a differing number of ACL statements were progressively applied to the

ingress and egress interfaces and evaluated. It was also noted that due to UDP not being a congestion-aware transport protocol and in order to maximize the throughput, the data streams were manually limited (to approximately 1Mbps of the maximum) to achieve the greatest device throughput. As such, the maximum UDP throughput achieved in this experiment was greater than in Cikara et. als' other experimentation and research [1]. It was considered that this would give a more true indication of the overheads incurred by implementing ACL statements. The direction of flow in the experiments conducted was from PC 2 which was connected to the fast Ethernet interface 0/1 of the router, to PC 1 which was connected to the routers' Fast Ethernet interface 0/0.

The PCs used were identical dual stack IBM compatible Celeron 800MHz machines running Windows 2003 Server with not service packs or hot fixes installed. The router utilised was a Cisco 2621XM router running Cisco IOS version 12.3(6). The methodology employed in this research was identical to that employed in [1].

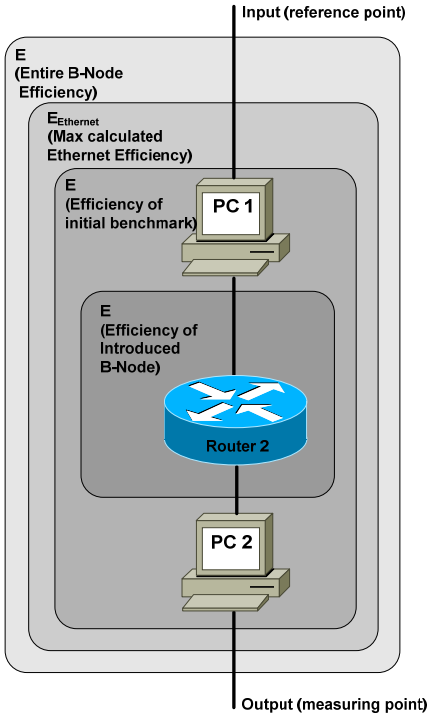


Figure 1: ACL evaluation experimentation setup

For the TCP IPv4 experiments, extended ACLs were used, and these were a collection of TCP deny statements from a specific host to another specific host. An example is shown below:
*access-list 101 deny tcp host 192.168.201.1 host 192.168.202.1 eq telnet*

Each line of the ACL would return a negative match, forcing each subsequent line of the list to be checked until the final statement,

*access-list 101 permit ip any any*

provided the match that permitted the traffic flow. The same approach was used for UDP, with the statements used shown below:

*access-list 101 deny udp host 192.168.201.1 host 192.168.202.1 eq tftp*

The identical methodology as described above was used for IPv6, with the exception that the IPv4 address as shown above was replaced with an IPv6 address.

## 8.2 IPv4 ACL Results

For TCP with the ACL applied to the in direction of Fast Ethernet (FA) 0/1, we experience an average linear decrease of 25kB per ACL statement. This corresponds with the result obtained by applying the ACL to FA0/1 to the out direction (25.1kB), which was as expected as this is the main direction of flow of the transfer. By applying the ACL to the opposite direction of the data transfer flow (for experiments conducted on FA 0/0 on the in direction and FA 0/1 on the out direction), we experience a lower effect in the decrease in performance. This is most likely due to the fact that the TCP acknowledgments PC 1 is sending to PC 2 travel opposite to the main direction of flow, and it is these acknowledgment packets to which the ACL has to filter. The average decrease experienced in this configuration was 11.1kB per ACL statement. By applying the ACL to both the in and out direction (for experiments with FA0/0 and then FA0/1), it was anticipated that the decrease in performance would be approximately 35kB per statement, which was a combination of the in and out direction results. This however proved to be incorrect, with an average of 28kB per ACL statement decrease observed. It was also anticipated that the results measured for TCP would have a very similar correlation for the experiments conducted with UDP. With the ACL applied to FA0/0 out and then FA0/1 in, an increase in BTC of 8% and 6% respectively was experienced compared to their TCP counterparts (27kB and 26.5kB per ACL statement). Applying the ACL statements in the opposite direction of the main traffic flow was expected to have no impact on network performance. The reasoning behind this was that as UDP is a connectionless technology, no acknowledgments from PC 1 will be sent to PC 2. This hypothesis proved correct with a negligible decrease in performance experienced from the maximum. The decrease in performance by applying the ACL to both the in and out directions was anticipated to be the same as applying the ACL to the main

direction of flow of traffic. Again this proved to be correct with the same decrease for FA0/0 (27kB) and FA0/1 (26.5kB) encountered. The B-Node device efficiency ($e_{Di}$) for the interface and hence the main direction of flow for ACLs can easily be extrapolated from the equations of the lines using the experimental data obtained (equation 4).

$$e_{Di} = \frac{-(decrease\ per\ ACL)(\#of\ ACL\ \ statements)\times(\max imum\ BTC)}{(\max imum\ BTC)} \quad (4)$$

For example, using the same experimental methodology as described above, applying 167 ACL statements to interface FA0/0 out, with 25.1kB per ACL statement decrease, the calculated efficiency, $e_{Di}$, that can be used in the B-Node formula is 45.58% (table 3).

Table 3: IPv4 ACL Summary

| Protocol | Interface | Direction | ACL |
|---|---|---|---|
| TCP | Fa0/0 | In | 11.10 |
| TCP | Fa0/0 | Out | 25.10 |
| TCP | Fa0/0 | Inout | 28.10 |
| TCP | Fa0/1 | In | 25.00 |
| TCP | Fa0/1 | Out | 11.10 |
| TCP | Fa0/1 | Inout | 27.90 |
| UDP | Fa0/0 | In | 0.00 |
| UDP | Fa0/0 | Out | 27.00 |
| UDP | Fa0/0 | Inout | 27.00 |
| UDP | Fa0/1 | In | 26.50 |
| UDP | Fa0/1 | Out | 0.00 |
| UDP | Fa0/1 | inout | 26.50 |

## 8.3 IPv6 ACL Results

For TCP ACL processing of IPv6 packets in the main direction of traffic flow had a relatively comparable decrease in the BTC (75.6kB for FA0/0 out and 84.2kB for FA0/1 in). This represents an approximate 300% to 336% increase in cost as opposed to IPv4. This relationship was only true for ACL statements less than 25. Interestingly after this number of statements (25), the BTC settled at a constant metric with a very slight, almost negligible, increase in performance (settling at 5.44MB/s with a 1kB increase per ACL for FA0/0 out and 5.25MB/s with a 0.6kB increase for FA0/1). In the opposite direction of main traffic flow, FA0/0 in experienced a 33.6kB decrease in bandwidth and FA0/1 had a 30.2kB decrease per statement. This represents a 272% to 303% increase on IPv4. Again after 25 ACLs the results showed a settling of BTC. FA0/0 in had a value of 6.55MB/s with a 0.3kB upward drift, and FA0/1 out had a value of 6.30MB/s with also a 0.3kB upward drift per statement.

Appling IPv6 ACLs in both directions (in and out) again followed the above trends, with FA0/0 demonstrating an 80.2kB decrease and FA0/1 88.9kB (which was up 286% to 317% on IPv4). For after 25 ACLs, FA0 settled at 5.33MB/s with a 0.8kB upward drift, and FA0/1 at 5.14MB/s with a 0.6kB upward drift. For FA0/0 out for UDP, the cost per ACL statement was 67.3kB opposed to 84.9kB for FA0/1 in. This was again significantly higher than IPv4 by about 249% and 315% respectively. For more than 25 ACLs, max BTC was 6.22MB/s with a 0.6kB upward drift for FA0/0 out 5.81MB/s with a 0.1kB upward drift fir FA0/1 in. For the direction opposite the main traffic flow, FA0/0 experienced a 0.1kB decrease per ACL statement and FA0/1 had 0.3kB. For more than 25 ACLs, FA0/0 had a 7.95MB/s transfer rate with zero drift, and likewise FA0/1 was 7.94MB/s with zero drift as well. In and out for both FA0/0 and FA0/1 was both 84.7 (about 300% worse than IPv4) with 6.22MB/s max with 0.6kB drift and 5.81MB/s with zero drift (table 4).

Table 4: IPv6 ACL summary

| Protocol | Interface | Direction | Cost per <25 ACLs> ((kB) | Cost per <25 ACLs> (kB) |
|---|---|---|---|---|
| TCP | Fa0/0 | In | 33.60 | 0.30 |
| TCP | Fa0/0 | Out | 75.60 | 1.00 |
| TCP | Fa0/0 | Inout | 80.20 | 0.80 |
| TCP | Fa0/1 | In | 84.20 | 0.60 |
| TCP | Fa0/1 | Out | 30.20 | 0.30 |
| TCP | Fa0/1 | Inout | 0.60 | 0.60 |
| UDP | Fa0/0 | In | 0.10 | 0.00 |
| UDP | Fa0/0 | Out | 67.30 | 0.60 |
| UDP | Fa0/0 | Inout | 84.70 | 0.60 |
| UDP | Fa0/1 | In | 84.90 | 0.10 |
| UDP | Fa0/1 | Out | 0.30 | 0.00 |
| UDP | Fa0/1 | inout | 84.70 | 0.00 |

## 8.4 ACL Conclusion

Overall, the cost of ACLs on IPv6 compared to IPv4 performed worse, with on average at least a 2 to 3 fold increase in the cost per ACL statement. This is however only true for up to 25 ACL statements. IPv4 costs are a linear decrease, dependent on the number of ACL statements in the list. This is also true for IPv6, however only to 25 statements. After 25 statements, the performance does not deviate from a constant value, except for a slight upward drift. Further investigation into these phenomena is required to explain this received outcome. The results obtained from the empirical validation (the equations of the lines) can easily be used to create and evaluate the performance degradation a desired number of ACL statements will have on network performance. This is hence used to determine the $e_{Di}$ figure in the B-Node equation for ACLs. Further investigation is required to ascertain as to why IPv6 ACL BTC is linear after 25 statements for both TCP and UDP.

## 9. Tunneling Empirical Validation

To empirically evaluate the effect the choice of tunneling protocol has on network performance (the BTC of the system) and how the choice of device impacts this ($e_{Di}$) the configuration was set up as shown in Figure 2. The same equipment was used as in Section 0, with the addition of another identical 2621XM Cisco Router. The tunnels established for the purposes of this experimentation were between the two routers, and they were configured as required by the variable being tested. As with all experimentation, the tunnels were up and established prior to experimentation to ensure no negotiations (if any) would impact the measured performance.

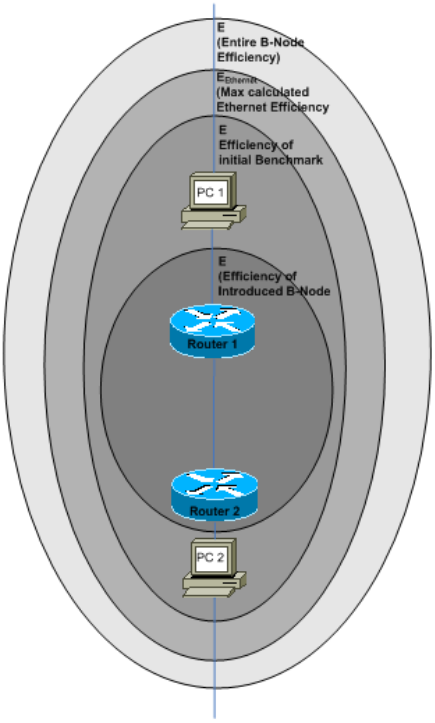


Figure 2: Tunneling Mode baseline experimental setup

## 10. Tunneling Results and Conclusion

The Bulk Transfer Capacity for various tunneling techniques was evaluated, and the results are shown in Table . Overall for IPv4-toIPv4 tunneling, the BTC experienced by the system was close to the baseline result. However, this was not the case for NOS compatible tunnels, where the BTC was significantly lower than others (NOS 1.06MB/s opposed to the Baseline 7.90MB/s for TCP and NOS 1.37MB/s contrasting with Baseline 7.11MB/s). This was both true for TCP and UDP. IPv6-toIPv4 tunneling contrasted markedly. Regardless of the method of tunneling, the BTC experienced was extremely low as compared to the Baseline. For TCP, BTC varied by 14KB/s between the three tunneling methods, and for UDP the variation was only marginally less at 13KB/s (table 5).

Table 5: Tunneling Technique Bulk Transfer Capacity

| | BTC (MB/s) | Theoretical max | Actual Entire B-node efficiency (%) | Efficiency of introduced B-node | Efficiency of introduced sub-B-node (%) |
|---|---|---|---|---|---|
| TCP baseline | 7.92 | 11.87 | 63.36 | 66.87 | Ref. value |
| 4to4 TCP IPIP | 7.91 | 11.70 | 63.28 | 66.79 | 99.87 |
| 4to4 TCP GRE | 7.90 | 11.67 | 63.20 | 66.70 | 99.75 |
| 4to4 TCP IPIP NOS | 1.06 | 11.70 | 8.48 | 8.95 | 13.38 |
| 6to4 TCP Manual | 1.22 | 11.54 | 9.76 | 10.30 | 15.40 |
| 6to4 TCP GRE | 1.30 | 11.51 | 10.40 | 10.98 | 16.41 |
| 6to4 IPv4 | 1.16 | 11.54 | 9.28 | 9.79 | 14 |
| 6to4 TCP overlay | 1.17 | 11.54 | 9.36 | 9.88 | 65 |
| UDP baseline | 7.11 | 11.96 | 56.88 | 60.54 | 14.77 |
| 4to4 TCP IPIP | 7.16 | 11.80 | 57.28 | 60.97 | Ref. value |
| 4to4 TCP GRE | 7.06 | 11.77 | 56.48 | 60.12 | 100.70 |
| 4to4 TCP IPIP NOS | 1.37 | 11.80 | 10.96 | 11.67 | 99.30 |
| 6to4 TCP Manual | 1.62 | 11.74 | 12.96 | 13.79 | 19.27 |
| 6to4 TCP GRE | 1.68 | 11.61 | 13.44 | 14.31 | 22.78 |
| 6to4 IPv4 | 1.55 | 11.64 | 12.40 | 13.20 | 23.63 |
| 6to4 TCP overlay | 1.68 | 11.64 | 13.44 | 14.31 | 21.80 |

# Conclusions

This paper has presented the empirical validation and quantified the impact of using techniques that are typically used in computer networks today- those being Access Control Lists and Tunneling. Even though ACLs do not add any overhead in terms of extra control or data packets, they do incur processing overhead due to the nature of the technology. For IPv4, these results are linear and can be easily modeled as an equation of a line, with the gradient of the line giving the cost per ACL statement. This is also true for IPv6 ACLs, but only for less than 25 statements. When this number is exceeded, the BTC of the system is constant regardless of the number of ACL statements processed. Further investigation is required to investigate this phenomenon. Tunneling is opposite to ACLs, in the fact that additional packet overheads are required to

implement the technology. These can be quantified and the estimated degradation in performance can be calculated. However, as many devices are a compromise between price versus performance, this is not always the case. This was experimentally evaluated, and demonstrated that IPv4-to-IPv4 tunneling has far superior performance that IPv6-toIPv4 tunneling. Again, this information can be used, and the corresponding device efficiency, $e_{Di}$, can be calculated, and used in further experimentation. Further related work to tunneling is to include the evaluation of IPv6-toIPv6 transitions mechanisms, and its impact on B-Node Theory.

## Acknowledgements

Mr S Cikara

## References

[1] Cikara, S., D.T. Shaw, and S.P. Maj. Modelling Layer 2 and Layer 3 Device Bandwidths using B-Node Theory. in 29th Australasian Computer Science Conference (ACSC 2006). 2006. Hobart, Australia.

[2] Maj, S.P. and D. Veal. Controlling Complexity in Information Technology: Systems and Solutions. in IASTED Conference on Computers and Advanced Technology in Education (CATE). 2001. Banff, Canada.

[3] Coccetti, F. and R. Percacci, Bandwidth Measurement and Router Queues. 2002, Sezione de Trieste: Trieste.

[4] Prasad, R., et al., Bandwidth Estimation: Metrics, Measurement Techniques, and Tools, in IEEE Network. 2003. p. 27- 35.

[5] Mathis, M. and M. Allman. RFC: 3148 A Framework for Defining Empirical Bulk Transfer Capacity Metrics. 2001 [cited; Available from: www.rfc-editor.org.

[6] Maj, S.P., D. Veal, and P. Charlesworth. Is Computer Technology Taught Upside Down? in ASEE Computers in Education Division. 2000. St Louis, Missouri, USA.

[7] Maj, S.P. and D. Veal. Architecture Abstraction as an Aid to Computer Technology Education. in ASEE Computers in Education Division. 2000. St Louis, Missouri, USA.

[8] Maj, S.P., D. Veal, and P. Charlesworth. Is Computer Technology Taught Upside Down? in 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education. 2000. Helsinki, Finland.

[9] Maj, S.P. and D. Veal. B-Nodes: A proposed new method for modelling information system technology. in International Conference on Computing and Information Technologies. 2001. Montclair State University, NJ, USA.

[10] Maj, S.P., D. Veal, and R. Duley. A Proposed New High Level Abstraction for Computer Technology. in ACM Special Interest Group for Computing Science Education (SIGCSE) 2nd Technical Symposium in Computer Science Education. 2001. Charlotte, North Carolina, USA.

[11] Maj, S.P., D. Veal, and A. Boyanich. A New Abstraction Model for Engineering Students. in UNESCO 4th UICEE Annual Engineering Education Conference. 2001. Bangkok, Thailand.

[12] Maj, S.P. and G. Kohli. Modelling Global IT Structures using B-Nodes. in 3rd Annual GITM World Conference. 2002. New York, USA.

[13] Veal, D., et al. A Framework for a Bandwidth Based Network Performance Model for CS Students. in 2005 ASEE Annual Conference and Exposition "The Changing Landscape of Engineering and Technology Education in a Global World". 2005. Portland, Oregon.

[14] Hudson, K., K. Caudle, and K. Cannon, CCNA quide to Cisco Networking (2nd Edition). 2003, Boston, MA: Thomson Course Technology.

[15] Odom, W., Cisco CCNA exam 640-607 certification Guide, C. Press, Editor. 2002: Indianapolis, IN.

[16] McGregor, M., CCNP Cisco networking Academy Program: Semester Five Companion Guide Advanced Routing, ed. A. Vito. 2001: Cisco Press.

[17] Cisco Systems Inc, Cisco Networking Academy Program CCNA 1 and 2 Companion Guide, Third Edition. 2003, Indianapolis, USA: Cisco Press.

[18] Morrissey, P. Demystifying Cisco Access Control Lists. [cited 2006; Available from: http://www.networkcomputing.com/907/907ws1.html.

[19] Davies, P.T., Securing and Controlling Cisco Routers. 2002, Boca Raton, FL: Auerbach Publications.

[20] Morrissey, P. Implementing Access-Control Lists: Access Control. 2004 [cited 2006; Available from: http://nwc.securitypipeline.com/shared/article/printablePipelineArticle.jhtml;jsessionid=QXWHTY2HVS2NYQSNDBGCKH0CJUMEKJVN?articleId=18400169.

[21] Grice, M., CCNP Guide to Advanced Routing, ed. S. Soloman and A. Valsangiacomo. 2001, Canada: Course Technology, Thomson Learning.

[22] Morrissey, P. The Cost of Security on Cisco Routers. 1999 [cited 2004; Available from: http://www.networkcomputing.com/1004/1004ws22.html.

[23] Veal, D., et al. A Framework for a Network Performance Model Based Upon B-Nodes. in 9th Annual Conference on Innovation and Technology in Computer Science Education (iTCSE). 2004. Leeds, UK.

[24] Cisco Systems Inc. Configuration Fundamentals Configuration Guide- Part 3 Interface Configuration. 2005 [cited 2005; Available from: http://www.cisco.com.

[25] Perkins, C. RFC 2003 IP Encapsulation within IP. 1996 [cited 2006; Available from: http://www.rfc-editor.org.

[26] DeMarco, T., Structured Analysis and System Specifications. 1979, London: Prentice/Hall.

[27] Karn, P. The KA9Q NOS TCP/IP Package. 2002 [cited 2006; Available from: http://www.ka9q.net/code/ka9qnos/.

[28] Karn, P. NET User Reference Manual (NOS Version). 1991 [cited 2006; Available from: ftp://ftp.simtel.net/pub/simtelnet/msdos/tcpip/.

[29] Lantz, B.A., Hamming it Up on Linux, in Linux Journal. 1995.

[30] Cisco Systems Inc. Cisco IOS Software Releases 12.3 T IPSec Virtual Tunnel Interface. 2005 [cited 2006; Available from: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123t/123t_14/gtipsctm.pdf.

[31] Information Sciences Institute. RFC 791 Internet Protocol. 1981 [cited; Available from: http://www.rfc-editor.org.

[32] Farinacci, D., et al. RFC 2784 Generic Routing Encapsulation (GRE). 2000 [cited; Available from: http://www.rfc-editor.org.
[33] Cisco Systems Inc. MPLS FAQ for beginners. 2005 [cited 2006; Available from: http://www.cisco.com/en/US/tech/tk436/tk428/technologies_q_and_a_item09186a00800949e5.shtml#qa11.
[34] Cisco Systems Inc. IP Fragmentation and PMTUD. 2005 [cited 2006; Available from: http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml.
[35] Wikipedia. Generic Routing Encapsulation. 2006 [cited 2006; Available from: http://en.wikipedia.org/wiki/Generic_Routing_Encapsulation.
[36] Cisco Systems Inc. GRE Tunnel Keepalives. 2005 [cited 2006; Available from: http://www.cisco.com/en/US/tech/tk827/tk369/technologies_tech_note09186a008048cffc.shtml.
[37] Carpenter, B. and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. 2001 [cited; Available from: http://www.rfc-editor.org.
[38] Cisco Systems Inc. Implementing Tunneling for IPv6. 2005 [cited 2005; Available from: http://www.cisco.com.
[39] Smith, C. IPv6 in Campus Networks. 2003 [cited 2005; Available from: http://www.cenic.org/events/cenic2004/pres/csmith.pdf.
[40] Cisco Systems Inc. Interconnecting IPv6 Domains Using Tunnels. 2003 [cited 2005; Available from: http://www.cisco.com.
[41] Nordmark, E. and R. Gilligan. RFC 4213 Basic Transition Mechanisms for IPv6 Hosts and Routers. 2005 [cited; Available from: http://www.rfc-editor.org.
[42] Cisco Systems Inc. Cisco- Tunneling IPv6 through and IPv4 Network (Document ID: 25156). 2005 [cited 2005; Available from: http://www.cisco.com.
[43] Cisco Systems Inc. IPv6: Providing IPv6 Services over and IPv4 Backbone Using Tunnels. 2003 [cited 2005; Available from: http://www.cisco.com.

**A/Prof S. P. Maj** has been highly successful in linking applied research with curriculum development. In 2000 he was nominated ECU University Research Leader of the Year award He was awarded an ECU Vice-Chancellor's Excellence in Teaching Award in 2002, and again in 2009. He received a National Carrick Citation in 2006 for "the development of world class curriculum and the design and implementation of associated world-class network teaching laboratories". He is the only Australian judge for the annual IEEE International Student Competition and was the first Australian reviewer for the American National Science Foundation (NSF) Courses, Curriculum and Laboratory Improvement (CCLI) program.

**Dr. David Veal** is a Senior Lecturer at Edith Cowan University. He is the manager of Cisco Network Academy Program at Edith Cowan University and be a unit coordinator of all Cisco network technology units. His research interests are in Graphical User Interface for the visually handicapped and also computer network modeling.