

Enriched Classification and Dynamic Tunneling as Elementary Internet Mechanisms

Gísli Hjálmtýsson and Björn Brynjúlfsson
Networking Systems and Services Laboratory
Department of Computer Science
Reykjavík University, Reykjavík, Iceland
{gisli,bjorninn}@ru.is

Abstract—In this paper we make the case that mechanisms for enriched classification and dynamic tunneling with a common elementary protocol should be widely deployed on the Internet. While such mechanisms are in common use and widely available their use remains ad-hoc, static and “underground.” Our elementary control protocol defines an interface that provides for consistent and interoperable implementations of the two mechanisms. We contend that rather than inviting the continued proprietary ad-hoc deployment, these mechanisms would bring substantial additional value to the Internet without significantly increasing complexity.

Index terms — Protocol design, Tunneling, Classification

I. INTRODUCTION

One of the strengths of the Internet is scalability in its forwarding, caused by aggregation of addressing. In the Internet architecture packet classification at input ports maps packets to outgoing ports based on longest prefix matching of the destination address. This allows routers to be oblivious of individual flows and indeed aggregate routing (and forwarding) entries for multiple flows and multiple hosts into one. Consequently, in segments of the network where addressing reflects the physical connectivity (as seen from a particular router) very few entries are sufficient for forwarding regardless of number of active hosts or flows.

Enriched classification is common and valuable. In spite of the clarity and scalability of this architecture attraction various factors have driven enriched classification functions into routers. Boundary gateways typically perform some form of fire-walling function, whether using Cisco style access control lists, or employing a full blown connection tracking firewall. Similarly, IntServ reservation based services, even coarser grained differentiated services, and traffic engineering in general call for richer packet classification. IP multicast requires per flow classification and forwarding based on source and destination. Continuous IP services in rapidly mobile environments call for enriched forwarding support either at the IP layer or by IP aware lower layer facilities.

Ignoring the reality and the need for forwarding based on enriched classification has resulted in a range of approaches being employed in ad-hoc manner at various points of the Internet. In fact, mechanisms for enriched forwarding are widely available, and on boxes at all levels of performance. However, some of these are vendor specific, others protocol or service specific, and in general cannot be activated in a coherent manner over multiple routers.

IP layer tunnels are common and valuable. IP-in-IP tunnels constitute another key invention by allowing a virtual Internet to be implemented over the Internet itself. A great benefit of IP-in-IP tunnels is to support incremental deployment. Indeed, tunneling is widely deployed for this purpose, in particular for test-beds such as the M-bone, the 6-bone and the A-bone, where the virtual connectivity of tunnels give the illusion of universal deployment. Tunneling is also commonly used with IPsec [1] to establish virtual private networks (VPN’s) over the open Internet. In addition, IP network operators and service providers employ various lower level mechanisms to effectively implement tunnels across multiple hops, using for example ATM virtual circuits, MPLS LSP, optical channels, MAC layer manipulations [2] and transport layer header swapping (e.g. UDP tunnels).

Protocols like mobile IP assume dynamic tunneling from the local agent at a router in the home domain to a remote agent at router in the remote domain. Newer protocols employ application level tunneling to overlay topologies over the Internet.

We contend that tunneling should be viewed as an IP layer abstraction, even when implemented using lower layer mechanism, and should be offered as a service on all (or most) routers. A basic abstract tunnel can be easily described, simply as an association between the tunnel end-points. By adding attributes, such as QoS reservations, security objects, or explicit routes for example, the basic abstraction can be specialized (sub-typed) to realize the various incarnations of tunneling in current use.

While the IP-in-IP encapsulation is defined in [3], no protocol exists to dynamically establish IP level tunnels. Although MPLS could be generalized to signal IP-in-IP tunnel requests, we contend that a more elementary protocol is sufficient and preferable. In fact, rather than

incorporating the functionality of traffic engineering, recovery, and more into a single protocol, we propose that it would be preferable to have multiple protocols each focusing on a single task, albeit employing same general purpose elementary mechanism.

In this paper we describe an elementary control protocol, and how this protocol can be used to construct dynamic tunnels, and to employ enriched classification mechanism to map a set of packets onto these tunnels. We claim that these mechanisms are needed and already commercially deployed. By standardizing the protocol interface to these basic mechanisms a number of applications and services would benefit substantially. We describe examples of how these elementary mechanisms can be used to realize multicast, mobility, traffic engineering, and more, that we have experimented with over the open Internet and in simulations.

The rest of the paper is organized as follows. After discussing related work on tunneling in section II we define the two mechanisms in section III. Section IV describes the elementary (yet extensible through subtyping) tunnel management protocol. In section V we give five specific examples of protocols and network services that would benefit significantly from our mechanisms. We then conclude in section VI.

II. RELATED WORK

RSVP [4] allows for receiver initiated resource reservation. An elementary request consists of a “flow spec”, used to set parameters in a node’s packet scheduler, and a “filter spec” is used to trigger filtering in a node’s classifier [5]. Our filters are similar to those of RSVP. In contrast to the complexity of RSVP, however, our tunneling protocol is designed to be used by other protocols as a general building block.

L2TP [6] is a protocol to establish dynamically PPP tunnels for Layer 2 circuits across packet-oriented data networks. The base L2TP protocol utilizes two types of messages, control message for dynamic creation, maintenance and tear-down of L2TP tunnels/sessions; and data message used to encapsulate PPP frames being carried over the tunnel. L2TP relies on Internet Protocol security (IPsec) for encryption services. In contrast to the layer 2 support, our protocol is aimed at facilitating the use of IP layer tunnels to establish forwarding paths, for the benefit of IP layer protocols and applications.

MPLS implements a virtual circuit service infrastructure incorporating a wide range of connection oriented functionality for traffic engineering, restoration and more. However, prior attempts to run IP over connection oriented

infrastructures (ATM) were not successful, and exhibited overburdening complexity. We fear that MPLS is headed the same way. Perhaps more importantly, we believe that having tunneling and enriched classification as elementary IP layer functions would benefit IP protocols and applications. In contrast MPLS provides virtual circuit capabilities beneath the IP layer and thus is inaccessible from higher protocols.

Beyond the examples presented in this paper, a number of other protocols and applications, such as various overlays [7], [8], IPv6 Tunnel Brokers [9] and mobile IP [10], would benefit from having tunneling mechanisms, as these protocols end up essentially replicating the functionality of the mechanisms and protocol presented herein.

III. THE MECHANISMS

The two mechanisms discussed in this section – enriched classification and tunneling – are well known and commonly used. The purpose of this section is to establish a reference definition for the subsequent discussion, of the mechanisms and of the information needed to be exchanged to manipulate these mechanisms.

A. Enriched classification

Many modern routers implement enriched forwarding, for example based on the five tuple, protocol id, source address, destination address, and source and destination ports.

Abstractly, a classifier maps packets into equivalence classes based on attributes of the packet. Commonly these attributes include the IP header and the transport level header. The equivalence relation is defined by a filter, F , so that two packets, p_i and p_j , are equivalent, iff $F(p_i) = F(p_j)$. We define the equivalence classes, in terms of a byte mask, M , and a value V , such that $F(p) = (p \& M = V)$, where the $\&$ is the bitwise AND operator. Multiple filters may map to the same equivalence class, i.e., an equivalence class is defined as $E = \{ p \mid F_0(p) \text{ or } F_1(p) \dots F_n(p) \}$. Each equivalence class has associated with it state, in particular the forwarding state. The forwarding state consists of a set of flow descriptors, each descriptor identifying a internal flow name, and a set of output device(s). Additional attributes may be assigned to an equivalence class, including queuing, bandwidth reservations and more.

In practice flow classification based on exact matches is relatively simple as those can be implemented by hashing. Range matching, e.g., based on prefix matching on source and destination, is however significantly more expensive [11].

B. Dynamic tunneling

We define an elementary tunnel to be an association between the two endpoints, the tunnel entry and the tunnel exit, plus an associated set of filters defining the set of packets to be routed to the tunnel at the tunnel entry. We consider all other tunnel attributes to specialize this basic type of tunnel. By this definition a tunnel constructs a unidirectional forwarding path from entry to exit. Taking an object-oriented type inheritance view, we consider all other types of tunnels to be subtypes of this base type. Using very late object binding we can therefore implement a general purpose interface and protocol, that still support subtype specific tunnel attributes.

We separate the abstraction of a tunnel from its implementation and specific attributes potentially associated with a tunnel. Examples of specific types of tunnels can include QoS attributes, route pinning, and security associations. A tunnel could abstract out physical implementations such as using optical light-paths.

Although tunnels are typically implemented as bidirectional interfaces, we define our abstraction as a one directional forwarding path. This simplifies dynamic tunnel construction and management, while providing a more valuable abstraction as a building block for protocols and applications, as discussed below. Bidirectional tunnels are constructed from two unidirectional tunnels.

One common objection to IP-in-IP tunnels is that tunnels impede performance. This need not be the case. In our implementation we have separated the processing entities (and object types) that implement tunnel entry and tunnel exit. Tunnel exit is trivial, as the router need only advance the IP-header pointer to the inner header and then restart the IP classification and processing. To improve tunnel entry performance, we have implemented in our prototype a scheme where for each packet the router allocates extra headroom enough for one extra outer header. If the packet is tunneled this avoids reallocating a new buffer and copying the data.

More elaborate tunneling may require more processing and incur additional overheads. For example, when using secure tunnels, the ESP header must also be processed at the tunnel boundary before the security processing (dispatch and encryption/decryption) takes place.

IV. THE TUNNEL MANAGEMENT PROTOCOL

The simple tunnel management protocol is designed to support the minimal functionality to establish the association between the tunnel endpoints, associate a filter to the tunnel at entry, and to invoke a subtype specific

create method at both ends of the tunnel. The protocol is a building block to be used by other protocols and services, and is void of all other semantics. The protocol supports tunnel creation only from the tunnel exit (i.e. the receiver) to the sender. The tunnel is assumed to become active instantly.

The protocol consists of two types of control messages and their associated semantics which are implemented at routers. The two messages defined are CREATE, used to create a tunnel, and REMOVE expressing that the tunnel is no longer desired. The protocol is designed to exchange the minimum amount of information required to successfully construct tunnels. The protocol is limited only to this task. All other concerns are exterior to the protocol and are addressed by service protocols or applications.

The protocol permits additional tunnel type specific data to be carried in the control messages allowing the exchange of type specific data. An option field in the protocol message header allows designation of the additional information and serves as subtype identifier to support a typical factory pattern. The additional data does not change the semantics of the two protocol messages and thus does not alter the complexity of the protocol.

A trivial tunnel, one between immediate neighbors, having no attributes other than the associated filter, is essentially a forwarding association and can be realized without any encapsulation taking place.

All state of the protocol is soft. As a consequence only the CREATE message and its correct interpretation is necessary for protocol correctness. The REMOVE message is used to accelerate resource reclaim, which is important for some protocols and applications. The control messages are sent as IP packets with a new protocol number and the Router Alert option, and carry the common protocol header. All control messages are sent best effort.

A. Soft state

The soft protocol state on routers is only valid for a certain period of time. Thus every system periodically sends CREATE messages to refresh the corresponding soft state, frequently enough to avoid unwanted loss of state due to random losses of CREATE messages. The primary benefit of soft state is simplification of exception handling. Specifically the use of soft state affords us to send the REMOVE message unreliably as the allocated forwarding state is eventually removed after timer expiration. To reduce the message overhead of maintaining the soft state we let the refresh interval grow exponentially (to a certain limit) as the forwarding state becomes older, as suggested in [12].

B. CREATE

CREATE messages are used to create tunnels and to refresh the forwarding state timer. The protocol header common to all messages is shown in Figure 1. It contains the following fields:

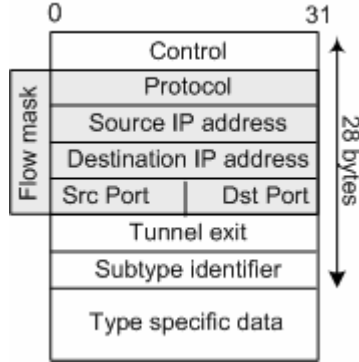


Figure 1 The protocol header for the tunneling management protocol

- **Control:** Containing control fields, such as protocol version etc, and an 8 bit message type set to the value CREATE or REMOVE (detailed content of the other 24 bits TBD via consensus).
- **Flow mask (F):** consisting of the five-tuple: protocol identifier, source address (S), destination address (D), source port, and destination port.
- **Tunnel-Exit-end:** The IP address of the end of the requested channel. This is either the router terminating the tunnel or the receiving end system.
- **Subtype identifier:** If present, indicated by a value not equal to null, the protocol message header is followed by type specific data.

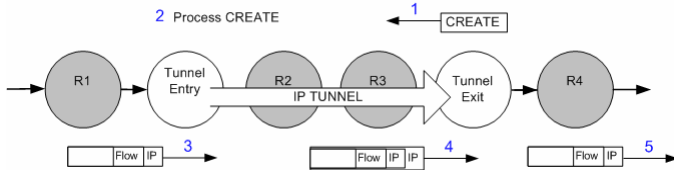


Figure 2 A dynamic tunnel being created and used with the protocol

An end system or a router interested in creating a tunnel sends a CREATE message towards the tunnel entry point, step 1 in Figure 2. The CREATE carries the flow mask, defining the flow which maps to the tunnel, the tunnel exit point (the CREATE sender stamps its own IP address here)

and, if appropriate, a subtype identifier. Upon receiving the CREATE message, step 2, the entry point of the tunnel sets up a classifier entry matching the flow and maps this to the tunnel, applying subtype processing as needed. Step 3, 4 and 5 show how a packet matching a flow traverses the tunnel.

Periodically - on expiration of the soft state refresh timer - each router/end system terminating a tunnel refreshes the upstream state, by resending a CREATE message for each channel associated with the expired timer.

The tunnel entry may refuse a tunnel request, for various reasons, including policy, security, or lack of resources. In this case it replies with a protocol unreachable ICMP message.

C. REMOVE

The only other message of the protocol is a REMOVE message, sent from a downstream node expressing that the association and the channel should be removed. The message has the same format as the CREATE message, with the message type set to REMOVE. While not necessary for the correctness of the protocol (due to the soft state) the REMOVE adds minimal complexity and expedites resource release.

V. PROTOCOLS AND APPLICATIONS

In this section we discuss few examples of how the elementary mechanisms and the tunnel management protocol can be applied as a building block to implement an array of protocols, and valuable network services.

A. Multicast

We have implemented and experimented with a Self-configuring Lightweight Internet Multicast (SLIM) [13], a single source multicast that incorporates dynamic tunneling to self-configure over the open Internet. Although our protocol specification declares protocol specific messages, JOIN and LEAVE protocol could easily be realized using the CREATE and REMOVE respectively, with an appropriate SLIM subtype. In the case of SLIM the filter specifies source and destination address as the multicast source and the channel identifier respectively, and nullifies the ports.

An end-system sends a CREATE message towards the source. The closest SLIM router intercepts the message (due to router alert, using the subtype as dispatch ID) and constructs a tunnel to the client and sets up the forwarding state, and suppresses the message. Each tunnel exit in the multicast distribution tree maintains the soft-state with

periodic refreshes. An end-system sends a REMOVE when the (last) application closes the channel. An intermediate router issues a REMOVE when the last outgoing branch either leaves or expires.

We have defined two mappings that constitute further subtyping. These subtypes have proven valuable in reaching end-systems not directly connected to a multicast router and allowing multicast traffic to flow through symmetric firewalls and NATs [14].

The elementary mechanisms described in this paper allow for simple implementation of multicast, support incremental deployment and organic growth of the multicast service.

B. Rapid mobility in wireless networks

As the network edge becomes increasingly wireless, and network use increasingly mobile, there is a need to incorporate support for mobility in the IP infrastructure. Whereas mobile IP may be suitable for personal mobility (relocation) it is inadequate for providing services where mobility is rapid and persistent.

Using the elementary mechanisms above, we are now experimenting with device initiated vertical handoffs for highly mobile environments. To initiate a handoff, the receiver sends a CREATE message towards the sender to the target base station (new access point). A mobility management daemon on routers (comparable to a multicast daemon) intercepts the CREATE request at the closest common ancestor of the two access points (old and new) and terminates the CREATE request. Instead it creates a new tunnel on the same flow mask, effectively triggering a multicast on both branches during the handoff. When the handoff is completed, the device sends a REMOVE to the old station terminating the multiple deliveries. Upon the next refresh of the new tunnel state, the management daemon at the branch point removes the tunnel entry at the intermediate point, reverting back to a single end-to-end tunnel.

C. Traffic engineering

A common use of ATM circuits and MPLS LSP's is for traffic engineering, either to load balance onto multiple paths or to explicitly route some class of packets between two routers along a path not selected by destination based routing. This is one of the primary reasons for current provider interest in MPLS.

In most cases, however, specifying every hop in a path is not necessary to achieve the traffic engineering goals. Most commonly, selecting a particular gateway, or a particular intermediate router is sufficient. For example, a

management system may decide to balance load from NY to LA among the three main paths in a backbone network, selecting between Chicago, Saint Luis or Huston as intermediate points may be sufficient to effectively select the routing path and balance the load. Constructing three tunnels from NY to each of these cities and load balancing among those tunnels thus achieves the task and significantly lower complexity (and cost) than by employing a full blown connection oriented infrastructure below the IP network.

D. Restoration-Failure hiding and recovery

Tunneling provides a powerful abstraction to allow for fast recovery of broken links. Using approaches similar to those used for connection-oriented fast restoration, a restoration tunnel can be pre-provisioned and associated with an exception so that when a failure occurs (e.g. link failure) the router immediately switches to the tunnel as an alternate path. When unplanned failures occur, some sub-optimality is unavoidable. Thus the restoration tunnel only needs to bridge the time it takes for the routing protocols to converge (after failure). A restoration tunnel thus only needs to satisfy two criteria: a) that it is forwarded on a different device than it is meant to protect, and b) the traffic routed into the tunnel upon a failure does not bounce right back. In prior work we have experimented with such a scheme for IP layer restoration in optical networks, resulting in restoration within a millisecond [15, 16].

E. Secure tunnels

Our mechanisms are sufficient for securing traffic between two points by allowing for the creation of secure tunnels. IPsec tunnels are constructed by sub-typing the basic tunnel, adding the security attributes. While the protocol is unchanged, the objects implementing the tunnel exit and entry are clearly more complex than for the elementary tunnel.

An end system/router interested in creating an IPsec tunnel sends a CREATE message to a tunnel entry point, setting the subtype identifier to IPsec. The type specific data trailing the protocol header includes attributes needed for the tunnel entry to establish a Security Association (SA) and start Internet Key Exchange. After establishing a valid SA the corresponding tunnel and flow state is setup. When the secure tunnel is no longer needed the end system sends a REMOVE message to the secure tunnel entry point.

We are currently prototyping these mechanisms on the Pronto [17] router.

VI. SUMMARY

We propose that enriched classification and dynamic tunneling should be viewed and deployed as elementary mechanisms on routers throughout the Internet. We have shown several examples of protocols and network services that would benefit substantially. As these mechanisms are already in widespread use, in an ad-hoc manner, we feel that the complexity of the Internet would be reduced by adopting these as building blocks.

ACKNOWLEDGMENTS

The authors would like to thank Ólafur Ragnar Helgasson for constructive criticism on this paper.

REFERENCES

- [1] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
- [2] G. Goldszmidt and G. Hunt, "Scaling Internet Services by Dynamic Allocation of Connections," in Proceedings of the 6th IFIP/IEEE Integrated Management, Boston MA, May 1999.
- [3] W. Simpson, "IP in IP Tunneling", RFC 1853, October 1995.
- [4] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol," IEEE Network Magazine, Vol. 7, No. 9 (Sept. 1993), pp. 8-18.
- [5] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.
- [6] W. Townsley, A. Valencia, A. Rubens, G. Pall G. Zorn, B. Palter, "Layer Two Tunneling Protocol L2TP", RFC 2661, August 1999.
- [7] B. Zhao, Y. Duan, L. Huang, A. Joseph, J. Kubiatowicz, "Brocade: Landmark Routing on Overlay Networks," First International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, March 2002.
- [8] Yatin Chawathe, "Scattercast: an adaptable broadcast distribution framework," ACM Multimedia Systems Journal 9(1), 104-118, 2003.
- [9] A. Durand, P. Fasano, I. Guardini, D. Lento, "IPv6 Tunnel Broker", RFC 3053, January 2001.
- [10] C. Perkins, "IP Mobility Support for IPv4", RFC 3344, August 2002.
- [11] T.V. Lakshman, and D. Stiliadis, "High-Speed Policy Based Packet Forwarding using Efficient Multi-Dimensional Range Matching," Proceedings of ACM SIGCOMM'98
- [12] H. Holbrook, S. Singhal and D. Cheriton, "Log-Based Receiver-reliable Multicast for Distributed Interactive Simulation," In Proceedings of ACM SIGCOMM 1995, pages 328-341, Cambridge, Massachusetts, August 1995.
- [13] G. Hjálmtýsson, B. Brynjúlfsson and Ó. R. Helgason, "Self-configuring Lightweight Internet Multicast - Protocol specification," IEEE SMC, 2004
- [14] Gísli Hjálmtýsson, Björn Brynjúlfsson and Ólafur Ragnar Helgason, "Overcoming last-hop/first-hop problems in IP multicast," in proceedings of NGC 2003, September 2003.
- [15] Albert Greenberg, Gísli Hjálmtýsson and Jennifer Yates, "Smart Routers - Simple Optics. A Network Architecture for IP over WDM," in the proceedings of the OFC 2000, Baltimore, March 2000.
- [16] Gísli Hjálmtýsson, Panagiotis Sebos, Graham Smith, and Jennifer Yates, "Simple IP Restoration for IP/GbE/10GbE optical networks," Postdeadline paper PD-36, OFC 2000, Baltimore, MD, March 2000.
- [17] G. Hjálmtýsson, H. Sverrisson, B. Brynjúlfsson and Ó. R. Helgason, "Dynamic packet processors - A new abstraction for router extensibility," in proceedings of OPENARCH-2003, San Francisco, April 2003.