Internet Draft
Document: draft-van-beijnum-multi6-odt-00.txt
Expires: July 2004

I. van Beijnum
January 2004

# On Demand Tunneling For Multihoming

## 1 Mandatory Statements



## 2 Abstract

This document describes a protocol to negotiate and subsequently use tunnels on demand, and the use of such tunnels to achieve scalable multihoming. The tunnels may run over either IPv4 or IPv6 transport, and the payload may consist of any upper layer protocol that can be used with IPv4 or IPv6. Tunnels are defined end-to-end, which means that a single outer or "locator" address pair maps to a single inner or "identifier" address pair. In order to avoid unnecessary overhead, tunneled packets (in the absence of option headers) only carry an outer header, an upper layer header and payload data. There is no inner header containing the original addresses; this information is reconstituted from the state associated with the tunnel where required.

Additional mechanisms are used to provide basic security for the mapping between inner and outer addresses and to avoid unnecessary overhead and delays for short-lived sessions, and to facilitate rehoming in the event of a reachability problem.

This document is a first draft, and as such doesn't contain a full protocol specification as higher-level changes are expected.

## 3 Rationale

In the IETF site multihoming in IPv6 (multi6) working group there have been many discussions over the past years about different approaches to scalable multihoming. One very promising approach is to separate the identifier and locator functions that have been traditionally overloaded in the IP address. An ideal locator/identifier separation solution wouldn't introduce any new state and subsequently not require any negotiation or other interaction between the hosts involved. However, it looks like this type of solution would enable a malicious host to use the multihoming mechanisms to perform redirection attacks. Rather than trying to keep stateless mechanisms and repair vulnerabilities with additional end-to-end interactions, the on demand tunneling approach adopts end-to-end negotiation to perform all discovery functions. The only exception is discovery of additional addresses before the ODT negotiation is started: these addresses are discovered through the DNS.

## 4 Identifier Limitations

It would be very easy to make the address family used as the outer or locator address type and the address family used as the inner or identifier address type completely independent. However, this has the unfortunate effect that anyone can use any identifier, as the implicit return routability check that exists in IP today doesn't apply anymore. And since current upper layer protocols aren't capable of using non-IPv4 and non-IPv6 address families anyway, this specification only allows the use of routable and reachable IPv4 and IPv6 addresses as inner addresses. Subsequent specifications may lift this limitation if they supply mechanisms to bind identifiers to their users in a reasonably secure manner.

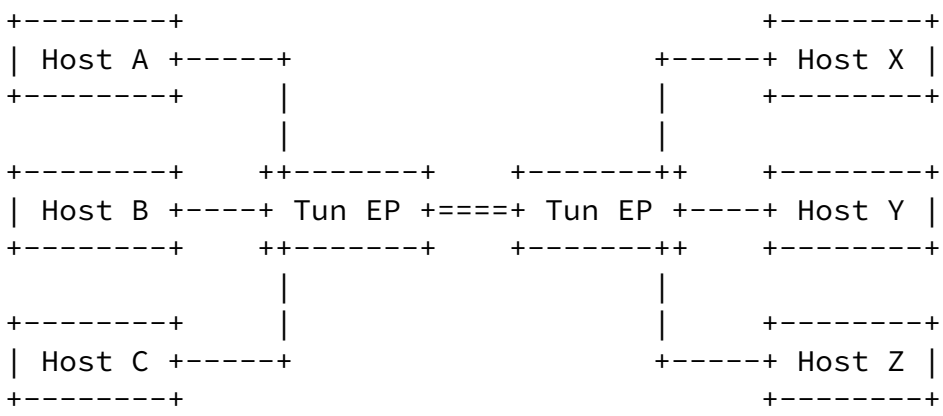
## 5 End-To-End Tunnels

Traditionally, tunnels are used to provide only a part of the full path packets must travel between two endpoints that are

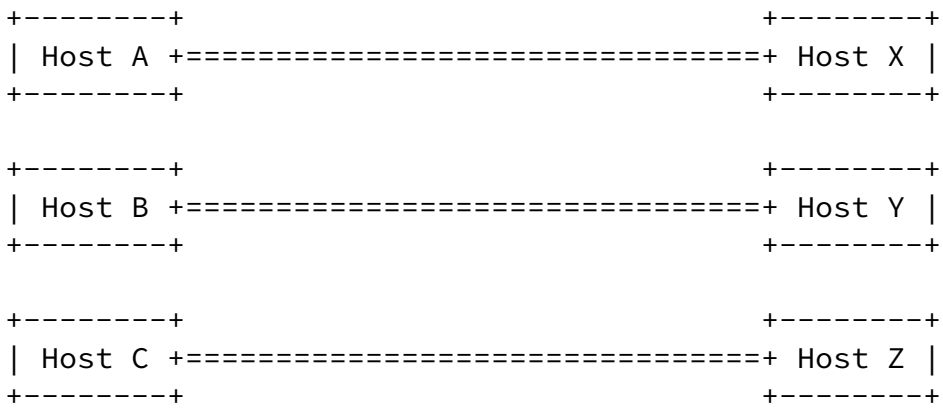
communicating. This means that almost always, a tunnel carries packets to and from many endpoints, making it necessary to explicitly carry addressing information in an inner header:








Since packet encapsulation and decapsulation for end-to-end tunnels happens on the endpoints themselves, the inner addresses are always the same so there is no need to explicitly carry them in headers:




## 6 Session and Context Start

With ODT in use, upper layer sessions are started as before. For instance, an application takes an FQDN and resolves it to a set of addresses, and then proceeds to open a TCP session towards the first

address that was returned by the resolver. The TCP three-way handshake executes as usual, with one change: in the absence of previously established state, the ODT multihoming mechanism adds an ODT header containing authentication information to the first packet transmitted in each direction; or ODT packets are exchanged as independent packets shortly after the first regular non-ODT packets are sent.

The ODT authentication information consists of a secret or semi-secret key that each side announces to its correspondent. However, IPsec may be applied to the packets containing the initial ODT interaction, possibly just for these packets and not for the packets carrying the actual communication. If IPsec isn't used anyone capable of intercepting the ODT packets can recover the key and subsequently redirect the data to themselves.





ODT tunnels can have different scopes:

- address only: addresses are remapped without regard for protocol or port numbers
- address and protocol: addresses are remapped if the protocol number matches
- address, protocol, service: addresses are remapped if they match the specified service
- address, protocol, port numbers: only a single session is remapped

"Address only" is mandatory; all other scopes are optional. Optional scope capabilities are announced in the initial ODT packets that are exchanged.

## 7 Unreachable Initial Address

As the ODT mechanism can only repair failures that occur after an initial exchange, an additional mechanism is necessary to establish communication when the initial address is unreachable. Preferably, applications or upper layer protocols should obtain a full list of addresses for a correspondent and try them all. To address the case where applications only try one address, systems implementing ODT MAY recover additional addresses from the resolver library or by doing a reverse and then a forward lookup on the available address, and then redirect a session setup attempt to such additional

addresses. However, as this breaks application expectations, this behavior MUST be both user and application controllable.

## 8 Tunnel Creation

On demand tunnels are created either immediately following the initial handshake or after it has become apparent that communication is long-lived and protection against failures is therefore prudent. It is even possible to start tunnel negotiation after the original address pair has stopped functioning if additional addresses are known through out-of-band means such as the DNS.

Tunnels are negotiated by the following sequence of exchanges:

1. Host A announces its addresses to host B

The addresses may be of different address families. Each address is accompanied by preference information. In order to not unnecessarily trigger NAT incompatibility, a "current source address" address family is used to refer to the source address in the IP packet, which may have been rewritten.

2. Host B sends challenge packets to addresses advertised by A





This is done to determine whether the advertised addresses are reachable and belong to host A. B may limit the number of challenge packets to a subset of the addresses advertised by A. The source address or addresses used by B must be known to A.

3. Host A replies to the challenge packets from host B

4. Host B sets up mapping state that allow it to remap A's original address to and from the additional addresses

5. Host B announces to host A which of A's addresses it is prepared to tunnel to/from

6. Both A and B monitor the communication and perform tunneling when regular reachability is impaired

7. When either side wishes to remove the tunnel state, it announces this to the other side at least 60 seconds in advance.

In most cases, the above steps will happen in both directions.

## 9 Session Collisions

If host X has two addresses published in the DNS, say X1 and X2, it is possible that another host A sets up connections to both of these addresses. If a tunnel is then created with X1 as the inner address and X2 as the outer address, X has no way to know whether a packet addressed to X2 was sent to X1 but tunneled to X2, or sent to X2 untunneled.

This problem can be avoided by either setting aside dedicated outer addresses, or by using a smaller scope. The former is advised for situations where there is no lack of address space (such as when IPv6 is used), the latter when address consumption is an issue. Note that it is always possible to reject a certain mapping or session setup request when this would create the potential for collisions.

## 10 The Protocol

When hosts implementing On Demand Tunneling need to exchange information pertaining to the protocol, they do this by encoding messages in an ODT extension header. Under some circumstances the header may be part of an IPv6 protocol chain and be followed by upper layer protocol data, under other circumstances the ODT header may constitute the entire payload of an IP packet.

When part of a protocol chain, the ODT header immediately follows a routing header, preceding a fragmentation header, any destination





options, an IPsec header and upper layer data. This is necessary as the protocol may be implemented in a middlebox.

When IPsec encryption or authentication of the ODT header is desired, the header must be be transmitted in an IP packet that doesn't contain any upper layer protocol. In this case, the IPsec header precedes the ODT header. [TBD: if implemented in a middlebox, the addresses involved in the IPsec negotiation don't match the inner addresses on one or both ends, handle this how?]

Included in all ODT headers/packets are:

- source inner address
- destination inner address
- if applicable: protocol, local and/or remote port numbers
- local context identifier
- remote context identifier
- a monotonically increasing sequence number
- a hash of the entire header and key previously learned from the other sid

The following ODT packet types are defined:

- key init
- error
- address announcement
- address/reachability challenge
- address/reachability response
- teardown

## 11 Tunneling Operation

Outgoing packets are matched against the established tunnel state to see if they're eligible for tunneling. When a packet potentially matches multiple tunnels, the one with the most specific scope is assumed to match. Whether the packet is forwarded as-is or tunneled depends on the reachability state of the untunneled inner or identifier address. As long as this address is reachable, no tunneling is performed.

If outgoing tunneled packets are TCP or UDP, and in the case of IPv4, when the UDP packet contains a checksum, the checksum is (re)calculated using the outer addresses in the pseudo header to create a packet with a valid TCP or UDP checksum. Implementations may use a simple incremental update on the checksum rather than perform a full checksum calculation over the entire segment. (This recalculation of the checksum makes it possible to use non-ODT aware checksum offload implementations.)





For incoming packets, the tunnel mappings are consulted to see whether the packet was tunneled. If so, the checksum is calculated using the outer addresses but apart from that the packet is processed as if it were sent using the inner addresses. If the segment is handed off to unchanged upper layer protocols, including

IPsec using authentication, the TCP or UDP checksum is reconstituted to match the inner addresses. However, in no event shall an invalid checksum using the outer addresses be overwritten with a valid one using the inner addresses.

## 12 API Changes

A mechanism to indicate that redirecting sessions isn't desired is necessary for systems implementing session creation redirection.

Upper layer protocols that wish to implement the use of more than a single source address and a single destination address may want to take advantage of the ODT address negotiation mechanisms. To be able to do this, additional API methods to enable sending and receiving using "raw" addresses and for binding additional addresses to a session are necessary.

## 13 Further Study: TLS

It would be useful to take advantage of the keying information negotiated through TLS in the same way that IPsec can be used to confidentially exchange ODT keys. Further study is required here.

## 14 ODTP in Middleboxes

If adding an ODTP header to the packet makes the packet too large to be transmitted to the next hop router or to its destination, a middlebox must either return a "packet too big" message, perform fragmentation itself or send the ODT header in a self-contained packet. However, the latter may lead to problems when the packet containing just an ODT header is treated differently from packets containing upper layer segments.

## 15 Non-ODTP Middlebox Considerations

ODTP interferes with the operation of certain middleboxes by introducing unpredictable routing of packets belonging to a single upper layer session and by rewriting source and/or destination addresses. If a middlebox isn't situated such that any and all packets pass it, the middlebox will very likely be unsuccessful in monitoring or manipulating an entire upper layer session. But even if the middlebox gets to see all packets, it may be fatally confused by a change in addresses.

## 16 Security Considerations

ODT can make applications use addresses that aren't visible to those applications. This can break assumptions on which security for certain applications is based.

When the initial ODT key exchange is intercepted, for instance when a wireless link is used without employing IPsec, attackers can redirect the traffic flow to arbitrary addresses under their control.

## 17 IANA Considerations

First of all, an IP protocol number is required for ODT. Second, a new AFI or SAFI number to indicate "source address in this packet" and finally a new registry containing ODT packet types and parameters.

## 18 Document and Author Information

This document expires July, 2004. The latest version will always be available at http://www.muada.com/drafts/. Comments are welcome at:

Iljitsch van Beijnum
Karel Roosstraat 95
2571 BG  Den Haag
Netherlands

Email: iljitsch@muada.com