# Bulk Transfer Capacity Estimation in IPv6 Networks – “ReturnFlow6”

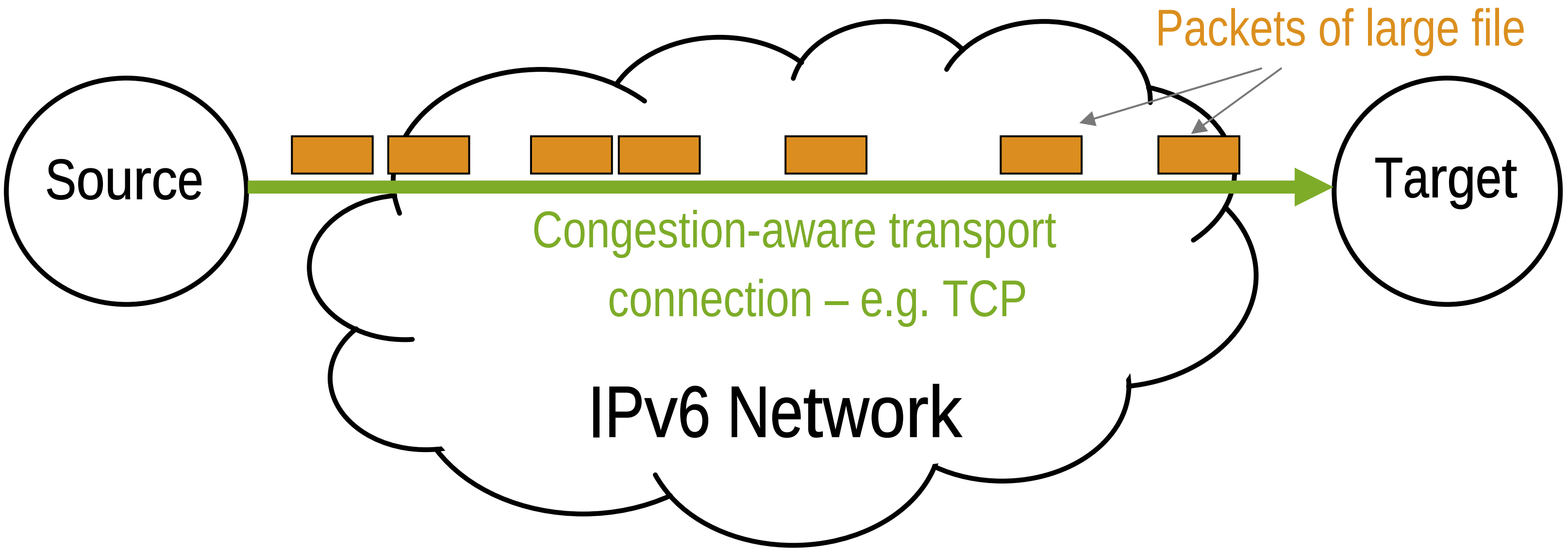
M-Power Project, Agilent Laboratories (Scotland)

Francisco Garcia, Robert Gardner

3rd August 2006

# Measurement of Bulk Transfer Capacity

- **Bulk Transport Capacity (BTC) is a measure of a network's ability to transfer significant quantities of data with a single congestion-aware transport connection → most commonly via TCP.**



# Measurement of Bulk Transfer Capacity

- **Network Operators & Service Providers are extremely interested in BTC measurements as they provide an indication of the quality of service (QoS) experienced by users over their networks.**

- **An estimate of BTC in an IPv6 network might be obtained by measuring the time taken to transfer a large file along a network path.**

- **However, this requires the instrumentation of the TCP receiver at the far-end of the network path.**

- **Also, for IPv6, it is not straightforward to simply port existing IPv4-based tools, because of stipulations in the IPv6 'standard'.**

# Measurement of Bulk Transfer Capacity

- **A Framework for Defining Empirical Bulk Transfer Capacity Metrics is defined in RFC 3148**

$$\textbf{BTC} = \frac{\text{data sent}}{\text{elapsed time}}$$

**where data sent represents data bits transferred not including header bits**

- **Bulk transfer capacity vs. available bandwidth**
  - Available Bandwidth → maximum attainable throughput
  - BTC → Attempt to achieve fair utilisation via congestion control

- **It is difficult to standardise BTC metrics:**
  - There are many congestion control algorithms permitted by IETF standards, thus different implementations will yield non-comparable measures. Thus BTC metric must be much more tightly specified than the typical IETF protocol.
  - Evidence that most TCP implementations exhibit non-linear performance over some portion of their operating region such that incremental improvements to a path (such as raising the link data rate) results in lower overall TCP throughput.

# Measurement of Bulk Transfer Capacity

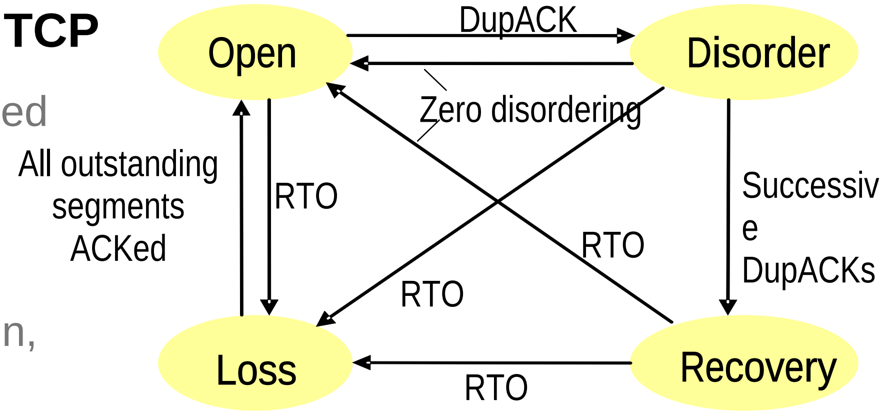
- **Congestion Control Algorithms**
  - Nearly all congestion control in the Internet is TCP-based and nearly all TCP implementations use congestion control algorithms published by Jacobson ("Congestion Avoidance and Control“) at SIGCOMM '88, refined in RFC2581.
  - TCP utilises a Sliding Window-based form of flow control, using algorithms that dynamically change the size of the window and retransmit ‘lost’ packets based on feedback in the form of acknowledgement packets.
  - Correspondingly, BTC methods are mostly based upon TCP implementations.

# Measurement of Bulk Transfer Capacity

- **Congestion Control Algorithms (…continued)**
  - TCP (and therefore BTC) implements an ACK clock and five standard congestion control algorithms: (1) Congestion Avoidance, (2) Retransmission Timeouts, (3) Slow-start, (4) Fast Retransmit and (5) Fast Recovery.
    - All BTC methodologies MUST implement Slow-start, Congestion Avoidance & Retransmission Timeouts.
    - All BTC methodologies SHOULD implement Fast Retransmit and Fast Recovery.
  - RFC2581 provides insufficient detail in many areas and thus the algorithm used in a particular BTC methodology MUST be defined – see RFC3148 for more details.
- **Ancillary Metrics**
  - Ancillary metrics can provide additional information about the network and the behavior of the implemented congestion control algorithms in response to the behavior of the network path. It is RECOMMENDED that these metrics be built into BTC methodologies.
    - Congestion Avoidance Capacity
    - Lost Transmission Opportunities
    - Losing an Entire Window
    - Heroic Clock Preservation
    - False Timeouts
    - False Timeouts
    - Ancillary Metrics Relating to Flow Based Path Properties
    - Ancillary Metrics as Calibration Checks
    - Ancillary Metrics Relating to the Need for Advanced TCP Features
    - Validate Reverse Path Load

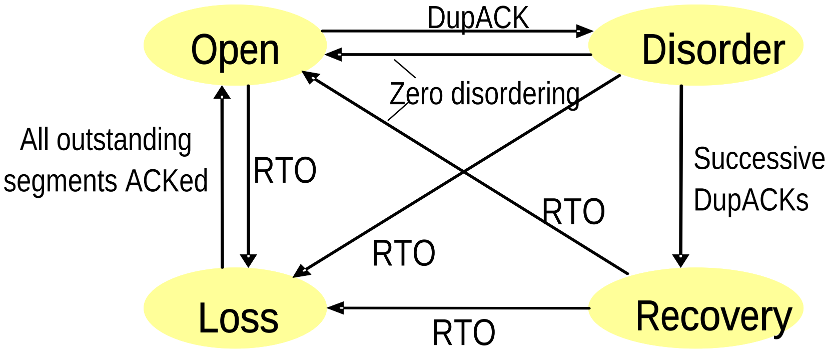
# Measurement of Bulk Transfer Capacity

- **Implementation of an algorithm to mimic TCP**
  - Sending packets to a target → ACK is returned
  - Round-trip time
- **Outline of the Algorithm**
  - TCP can be modelled using four states: Open, Disorder, Recovery and Loss.



  - Movement between states depends on how packets are treated by the network.
  - Packet transmissions if segments ‘in-flight’ < size of the congestion window {*cwnd*}
- **Open State:**
  - Each ACK expands *cwnd* by one packet in slow-start mode.
  - Once *cwnd* reaches the slow start threshold {*ssthresh*}, → congestion avoidance
  - With no packet loss, transfer rate would reach an equilibrium dependent on the lower of the total link bandwidth or the maximum window size divided by RTT.
  - Packet losses occur because of congestion and Disorder state is entered.

# Measurement of Bulk Transfer Capacity



- **Disorder state:**
  - Each ACK triggers transmission of new packet.
  - Return to the Open state if the disordering resolves itself or to the Recovery state if not.
- **Recovery:**
  - The Recovery state emulates the Fast Retransmit and Fast Recovery algorithms.
  - The first missing segments are retransmitted and *cwnd* and *ssthresh* are reduced.
  - Every 2nd ACK continues to deflate *cwnd* by one segment until it reaches *ssthresh*.
  - First missing then new segments continue to be transmitted until all outstanding ACKs have successfully been received, then the Open state is re-entered.
- **The Loss:**
  - Loss state is entered at any time on the expiry of the retransmission timeout (RTO).
  - All unacknowledged segments are marked as lost, and *cwnd* is set to one segment and slow-start is invoked.
  - The system exits to Open state after all outstanding segments have been ACKed.

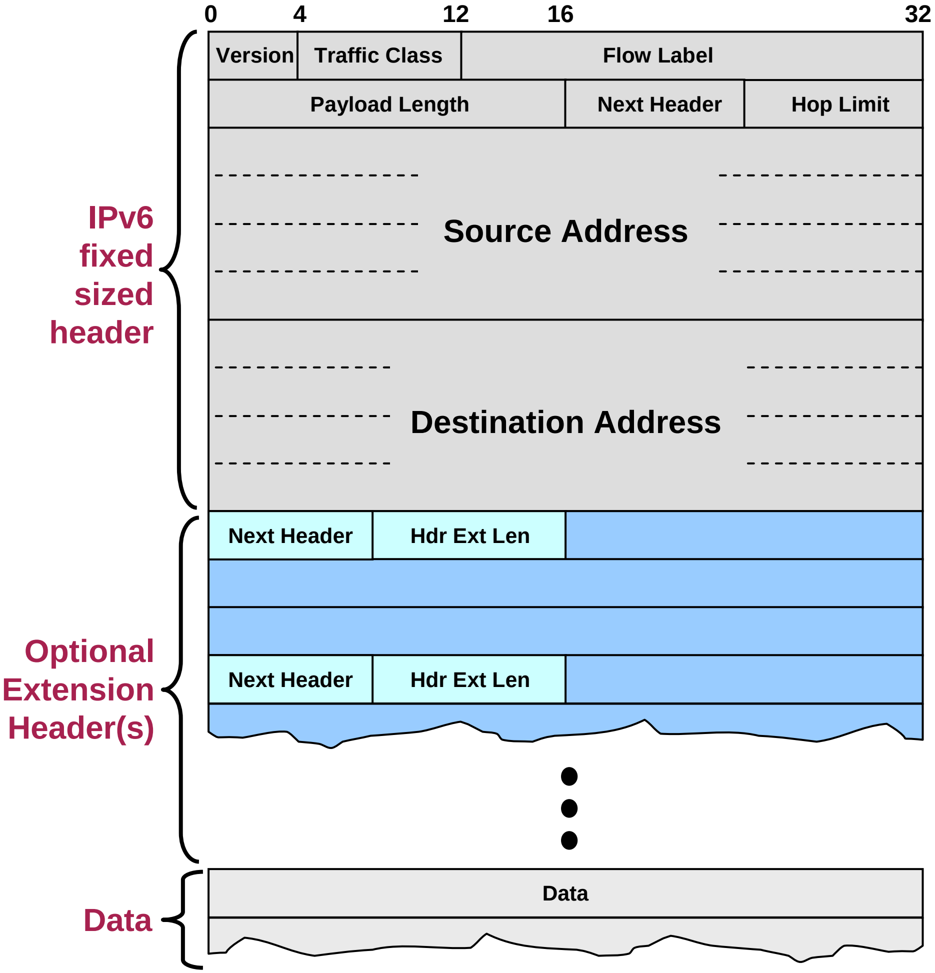
# Measurement of Bulk Transfer Capacity

## QUESTION:

**How do we emulate TCP packet transmission and the return of acknowledgements in IPv6 without instrumenting the target node?**

# IPv6 - The Extendible Protocol

## IPv6 Header Format



- **IPv6 Extension Headers (EHs)**
  - Optional information encoded in separate headers between IPv6 header and upper-layer header.
  - Packet may carry zero or more EHs.
  - So far there are only a handful of standardised EHs:
    - Hop-by-Hop Options.
    - Routing (Type 0).
    - Fragment.
    - Destination Options.
    - Authentication.
    - Encapsulating Security Payload.
  - With one exception, EHs are only processed at the destination(s).
  - The exception is the Hop-by-Hop Options, processed at every node.

# Routing Header

- **The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination.**
  - Identified by a Next Header value of 43 in the immediately preceding header

## Generic Routing header

| 1 octet | 1 octet | 1 octet | 1 octet |
|---|---|---|---|
| **Next Header** | **Hdr Ext Len** | **Routing Type** | **Segments Left** |
| **Routing type-specific data** | | | |

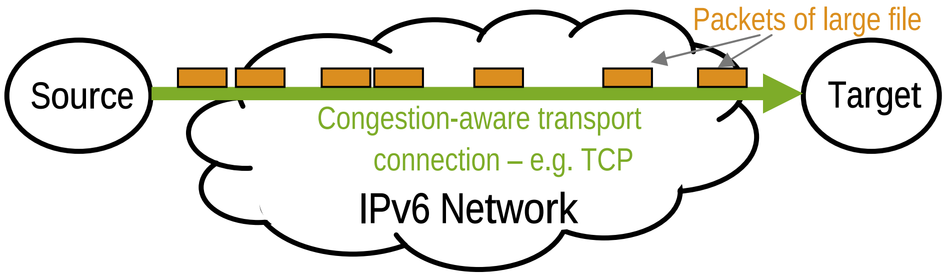
- A Routing header is not examined or processed until it reaches the node identified in the Destination Address field of the IPv6 header.
- The Destination Address is then swapped with one from the Routing header and so on…

## Type 0 Routing header

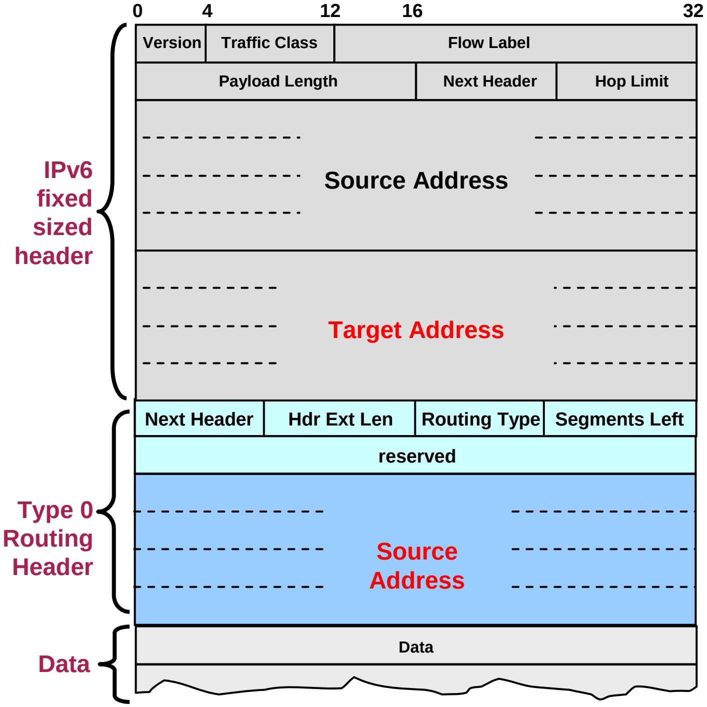
| 1 octet | 1 octet | 1 octet | 1 octet |
|---|---|---|---|
| **Next Header** | **Hdr Ext Len** | **Routing Type** | **Segments Left** |
| **reserved** | | | |
| **Address 1** | | | |
| | | | |
| **Address N** | | | |

# Routing Header

- **By creating a packet whose Routing Header causes it to return to the source, we can emulate the acknowledgement mechanism of the TCP implementation**



**IPv6 Header with Routing Header**



0 4 12 16 32

IPv6 fixed sized header

Version | Traffic Class | Flow Label

Payload Length | Next Header | Hop Limit

Source Address

Target Address

Type 0 Routing Header

Next Header | Hdr Ext Len | Routing Type | Segments Left

reserved

Source Address

Data

Data

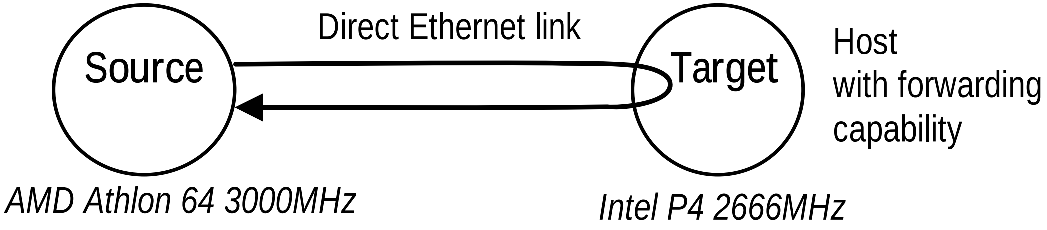
# ReturnFlow6

- **ReturnFlow6 is a single-point BTC measurement application for use in IPv6 networks.**
- **ReturnFlow6 measures the BTC of a path out to one or more a remote nodes in the network.**
- **ReturnFlow6 uses IPv6 Routing Headers to specify the measurement path.**
- **ReturnFlow6 implements a RFC3148-compliant BTC measurement algorithm (an emulation based on the Linux implementation of Transmission Control Protocol) in order to make estimates of the Bulk Transfer Capacity of a network path.**
    - Uses an emulation closely based on the Linux implementation of Transmission Control Protocol and implements Slow-start, Congestion Avoidance, Fast Retransmit, Fast Recovery and Retransmission Timeouts.
    - The path must form a round trip and it is not possible to distinguish between the BTC for ‘outward’ / ‘intermediate’ / ‘return’ paths

# ReturnFlow vs. Treno vs. wget

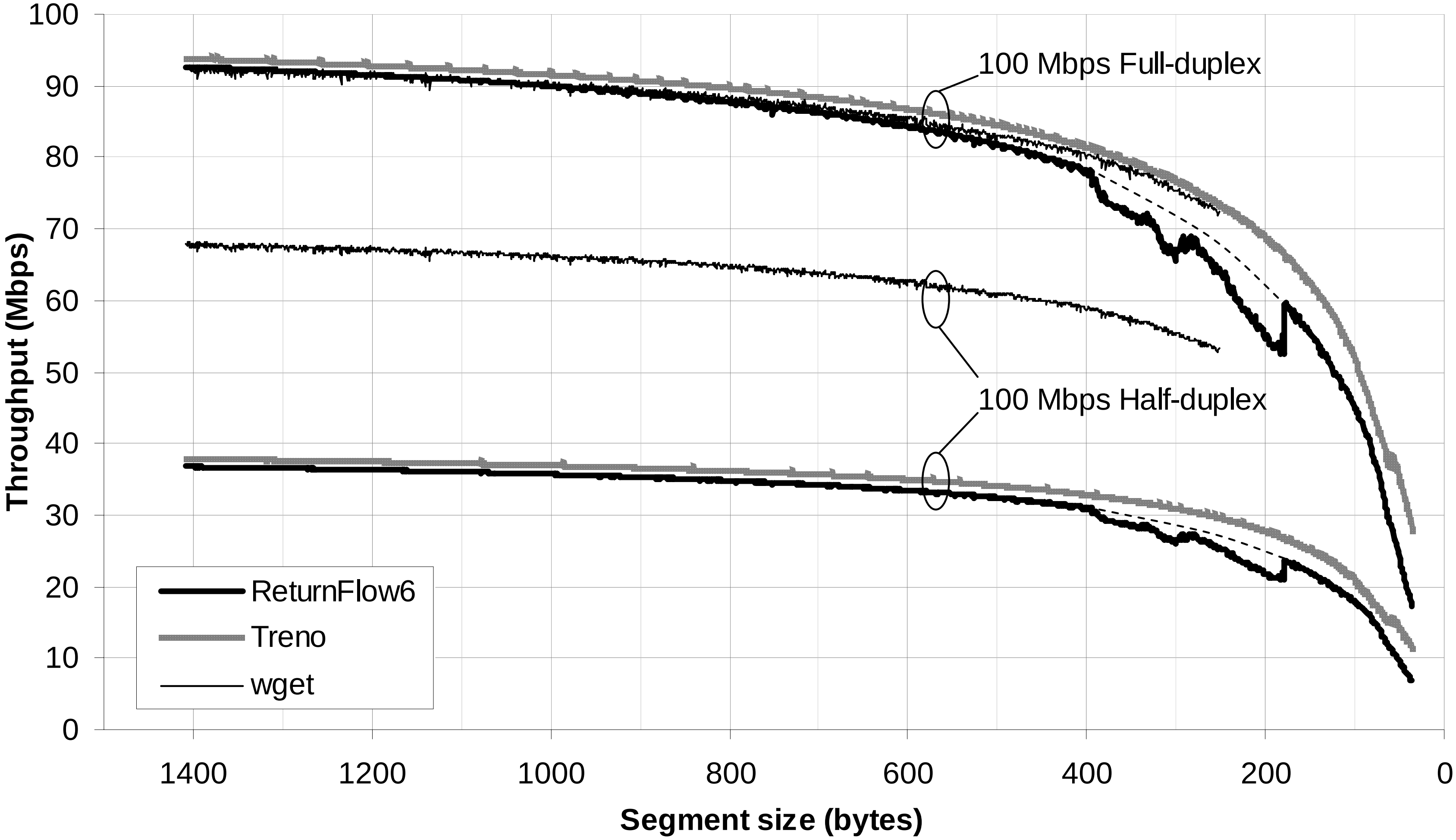
- ***Treno* (Traceroute RENO)**
  - Treno is a network testing tool designed to test network performance under load similar to that of TCP.
  - In one mode, Treno uses the same technique as traceroute to probe the network. By sending out UDP packets with low TTL, hosts and routers along the path to the final destination will send back ICMP TTL Exceeded messages which have similar characteristics to TCP ACK packets.
  - Treno also has an ICMP mode, which uses ICMP ECHO Requests instead of low TTL UDP packets. In this mode, you only get information about the final destination; however, the same sized packets are sent in both directions, giving you some information about the return path.
- **wget**
  - GNU wget is a free software package for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols.
  - It is a non-interactive command-line tool, so it may easily be called from scripts, cron jobs, terminals without X-Windows support, etc.

# Experimental results: Point-to-point link



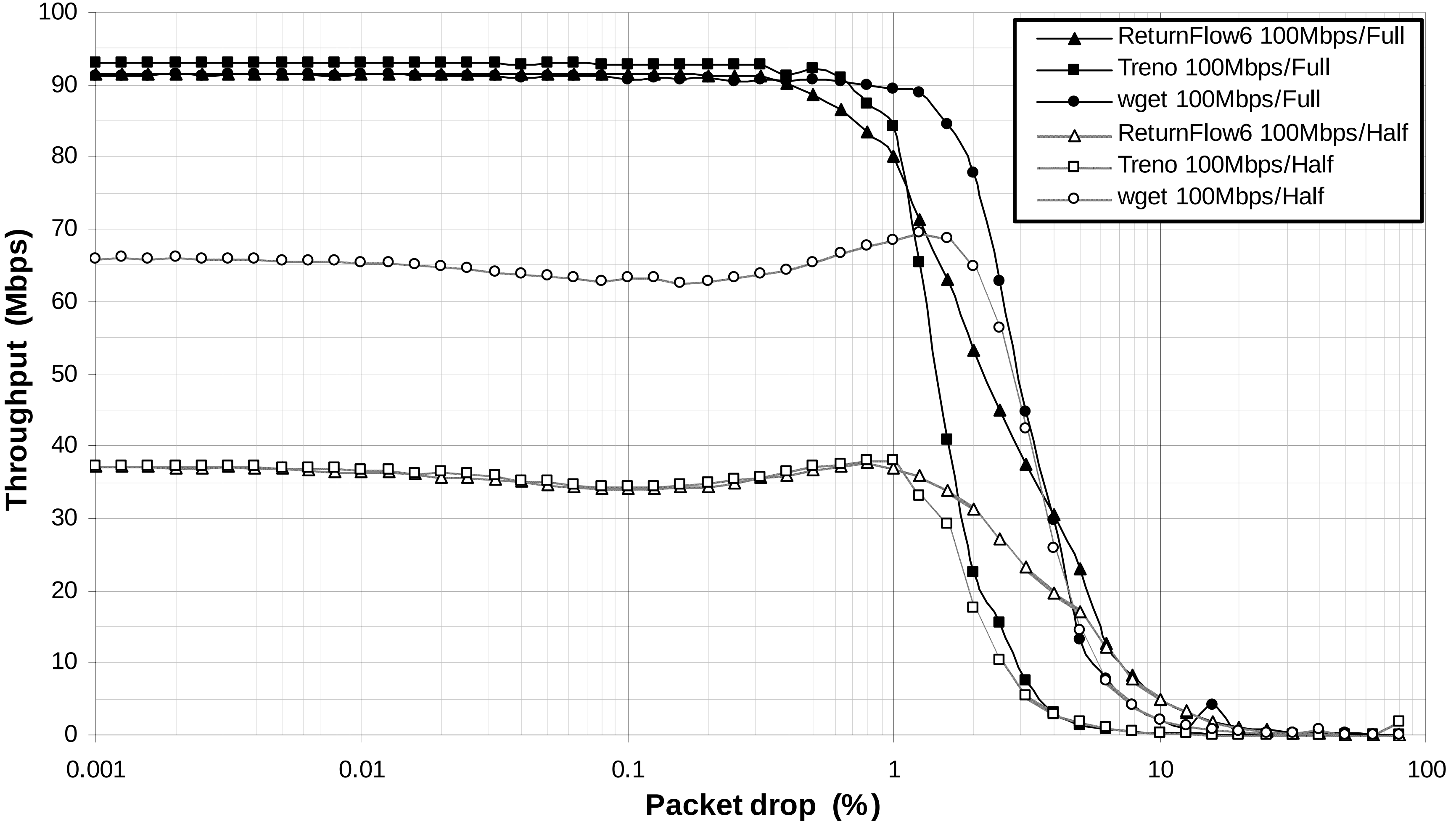
- **Unimpeded bulk data transfers for *ReturnFlow6*, *Treno* and *wget* on a single Ethernet link between two Linux hosts using 100Mbps Full Duplex and 100Mbps Half Duplex modes.**
- **Tests of approximate duration 30 seconds were carried out for increasing segment sizes of 36-1408 bytes.**
- **Each test was carried out three times.**
- **Note that whereas there is direct control over the segment sizes in *ReturnFlow6* and *Treno*, the segment size for *wget* transfers was controlled by altering the link MTU [via the Linux '*ifconfig*' utility]**
- **It was not possible to force *wget* segment sizes of less than 252 bytes.**

# Experimental results: Single Point-2-Point Link



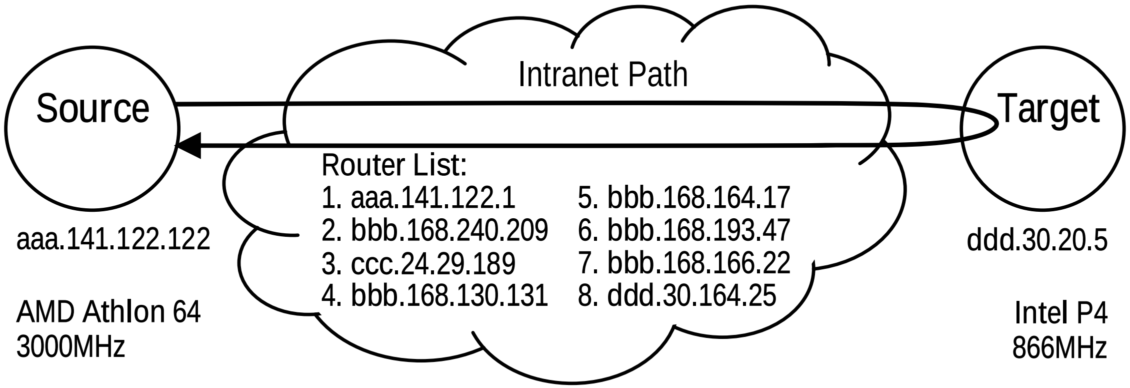
- Figure 5: Transfers over P2P link at 100Mbps full- and half-duplex

# Experimental results:
# Single P2P Link with added artificial loss



- Figure 6: 1208-byte segs, 100Mbps full/half duplex, varying packet drop rates

# Experimental results: Transatlantic Intranet Path

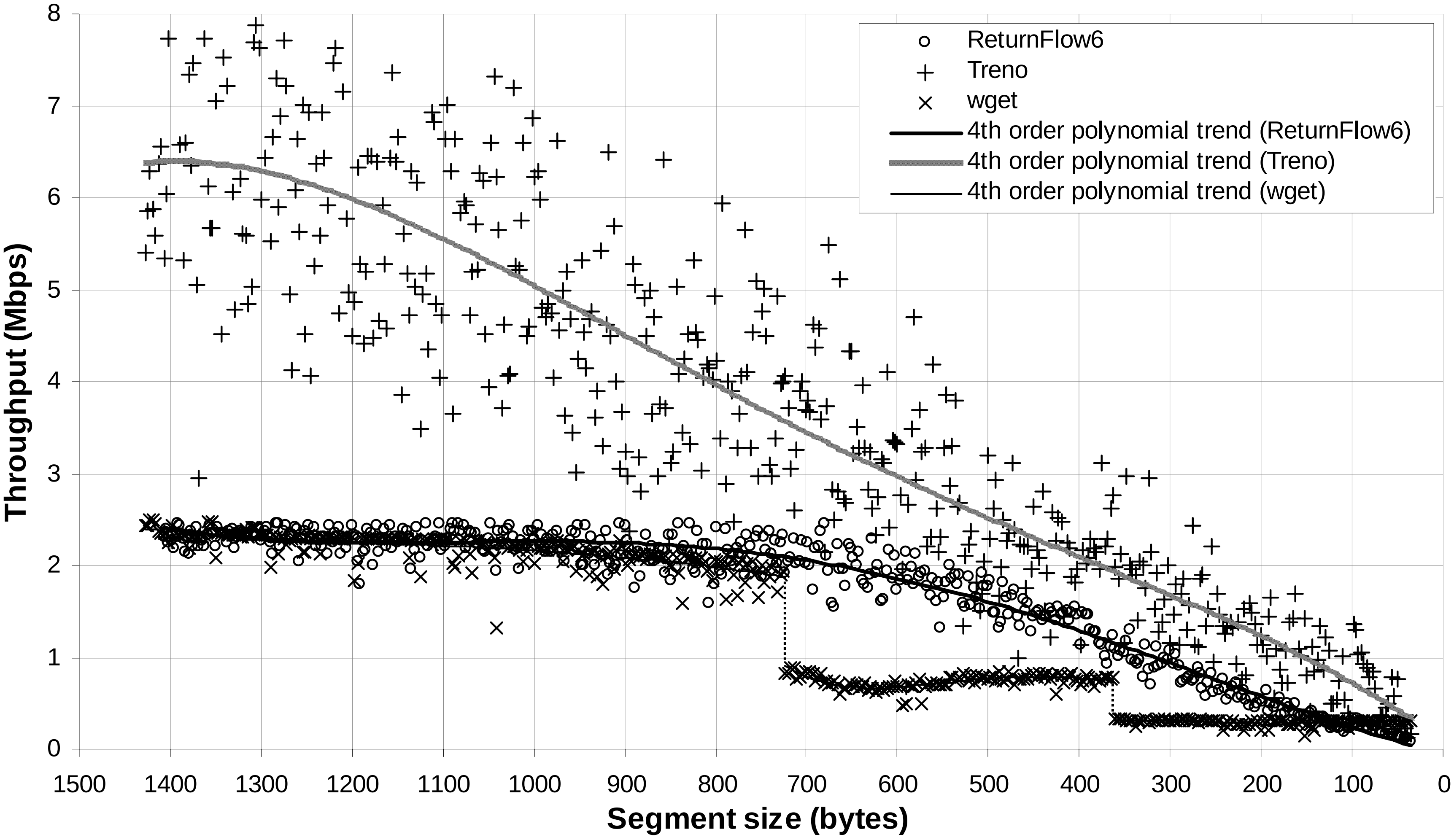


Router List:

| | |
|---|---|
| 1. aaa.141.122.1 | 5. bbb.168.164.17 |
| 2. bbb.168.240.209 | 6. bbb.168.193.47 |
| 3. ccc.24.29.189 | 7. bbb.168.166.22 |
| 4. bbb.168.130.131 | 8. ddd.30.164.25 |

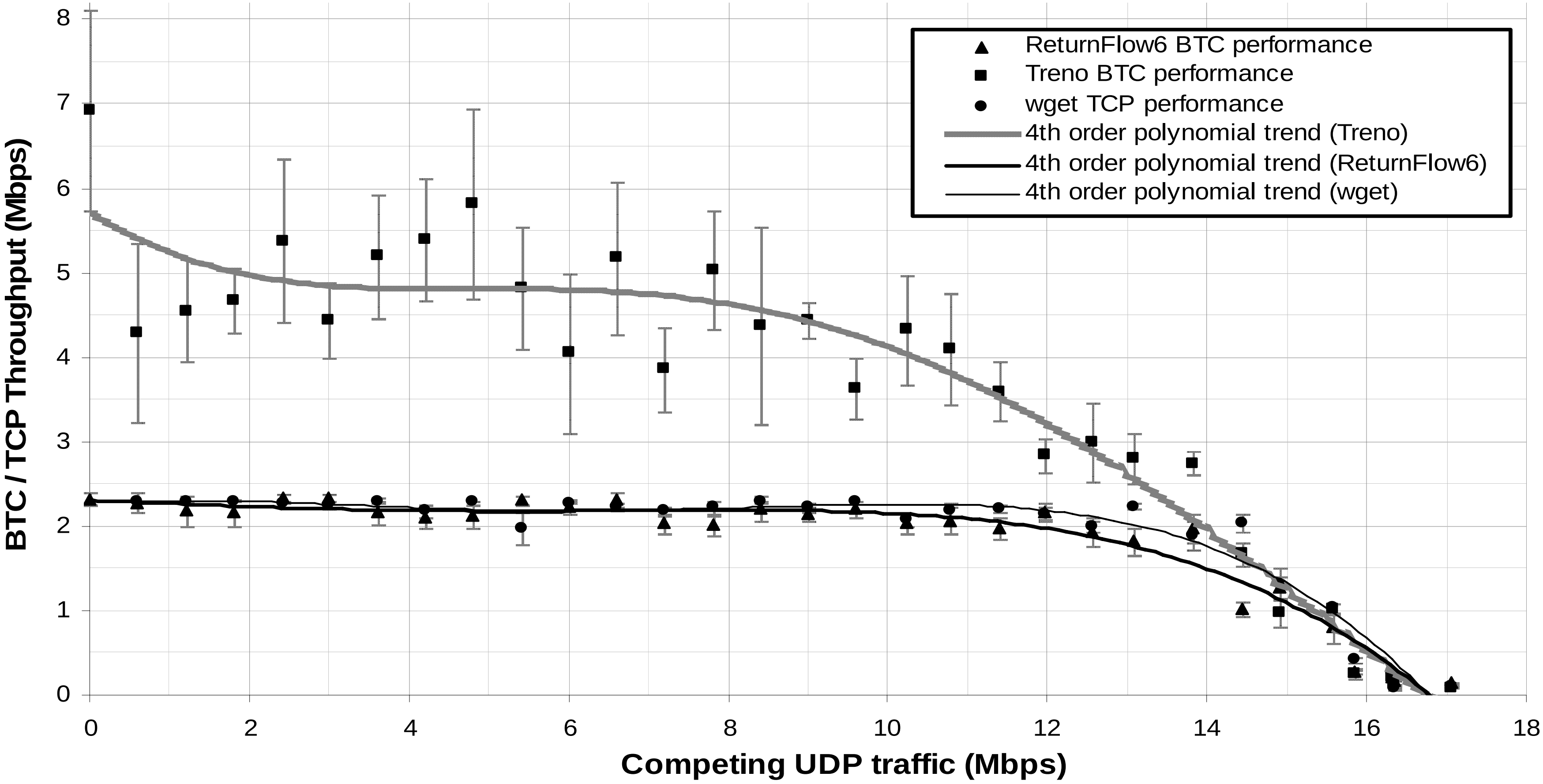
- **Next, the BTC tools were used on a ‘live’ Intranet from the UK to the USA.**
- **Competition for network resources with other traffic from across the company.**
- **The network path consisted of eight IPv4-based routers → 9 hops.**
- **An IPv6-in-IPv4 tunnel was used to convey *ReturnFlow6* packets since the Intranet was not a native IPv6 network.**

# Experimental results: Transatlantic Intranet Path

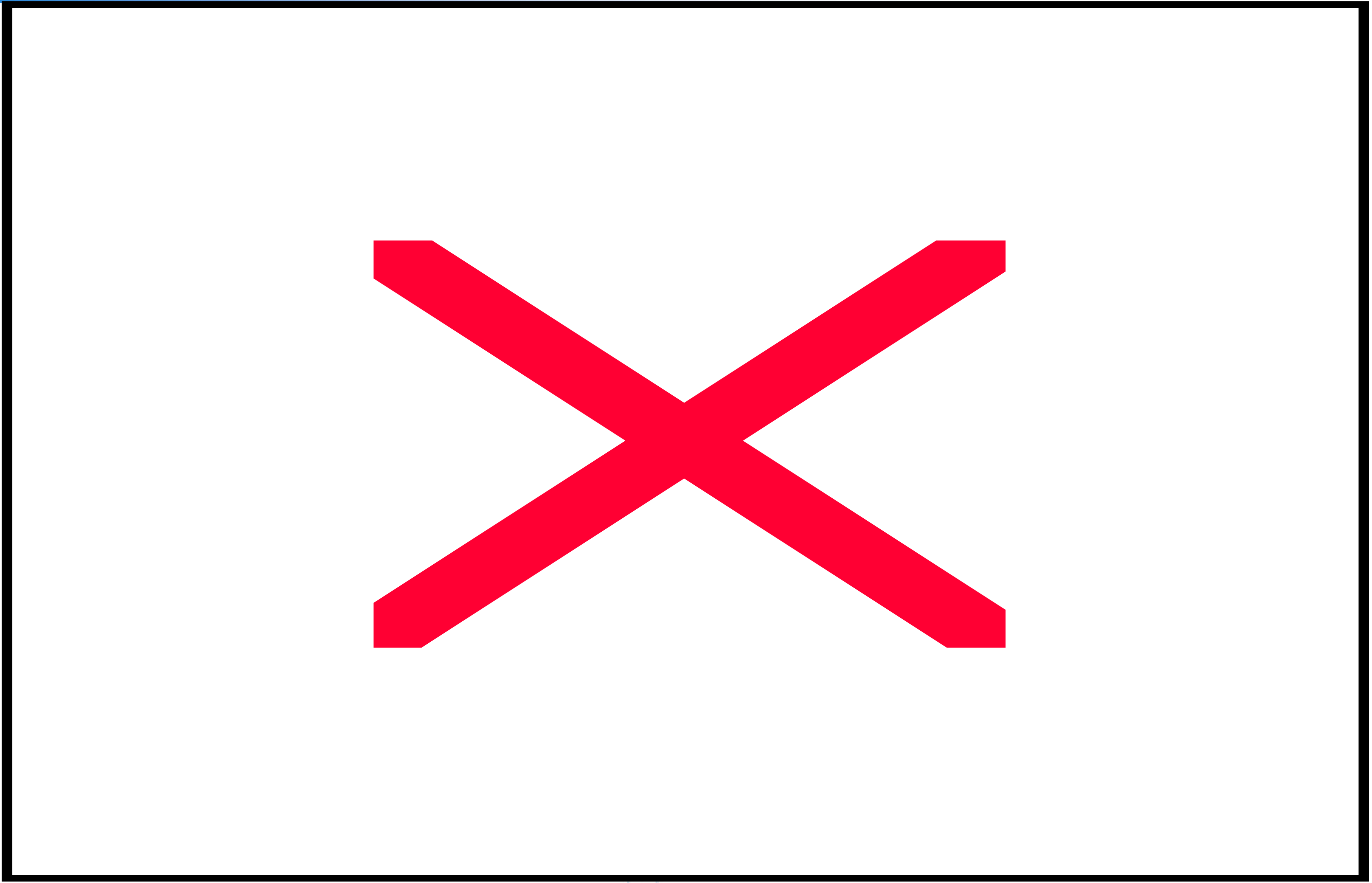


# Experimental results:

## Transatlantic Intranet Path with competing traffic



# Experimental results: Single 100Mbps Full Duplex Link to Cisco 2621 Router



# Results

- **Treno (ICMP-version) and ReturnFlow6 have a closely matching performance when no ICMP rate-limiting is done by network.**
- **ReturnFlow6 has superior performance as ICMP rate-limiting does not interfere with its operation. Treno measurements are adversely affected.**
- **When no rate-limiting is in operation, Treno (UDP) will more closely match real TCP performance because returning packets are closer to TCP ACK size. ReturnFlow6 will measure a more conservative throughput under such asymmetric conditions.**
- **As an extension of ReturnFlow6, it may be appropriate to produce an IPv6 version of Treno (TRENOv6) and conduct performance comparisons. However IPv6 networks also perform ICMPv6 rate-limiting.**
  - An operational strategy might be to perform a TRENOv6 measurement followed by a ReturnFlow6 measurement and use the greater value as the estimated BTC.

# The End

M-Power

## Thank you for your attention.

## Any Questions?