Passive Network Analysis Using Libtrace

Shane Alcock

Introductions

- Shane Alcock
  - WAND member since 2003
  - Libtrace developer since 2005
- Tony McGregor
  - Associate Professor at University of Waikato
  - Former head of NLANR Active Measurement Project
  - Keynote speaker at PDCAT

Outline

- Introduction and Basics
- The Libtrace Tools
- Simple Libtrace Programming
- Advanced Topics

Part One

- Introduction to Passive Measurement
- Programming Issues
- Introducing Libtrace
- A Brief History Lesson
- Acquiring and Installing Libtrace
- Libtrace Basics: URIs
- Libtrace Basics: BPF filters

Passive Measurement

- Use existing network traffic to analyse network behaviour
  - No artificial “measurement” traffic
- Can be divided into two principal steps
  - Capture – reading data off the network
  - Analysis – applying metrics to the data
- We're going to focus on measurement at the packet level

Packet Capture

- Hardware
  - Endace DAG cards
- Software
  - PCAP (tcpdump)
- Kernel
  - Linux native

Packet Capture

- A header is prepended to each captured packet
  - Timestamps
  - Packet length
- Header structure differs for each capture format
  - Timestamp format can be different too

Packet Capture

- Example: ERF header used by DAG

<table>
<tr><td colspan="3">Timestamp</td></tr>
<tr><td colspan="3">Timestamp</td></tr>
<tr><td>Frame Type</td><td>Flags</td><td>Record Length</td></tr>
<tr><td colspan="2">Loss Counter</td><td>Wire Length</td></tr>
</table>

- Example: PCAP header

<table>
<tr><td colspan="2">Timestamp</td></tr>
<tr><td colspan="2">Timestamp</td></tr>
<tr><td>Capture Length</td><td>Wire Length</td></tr>
</table>

Packet Traces

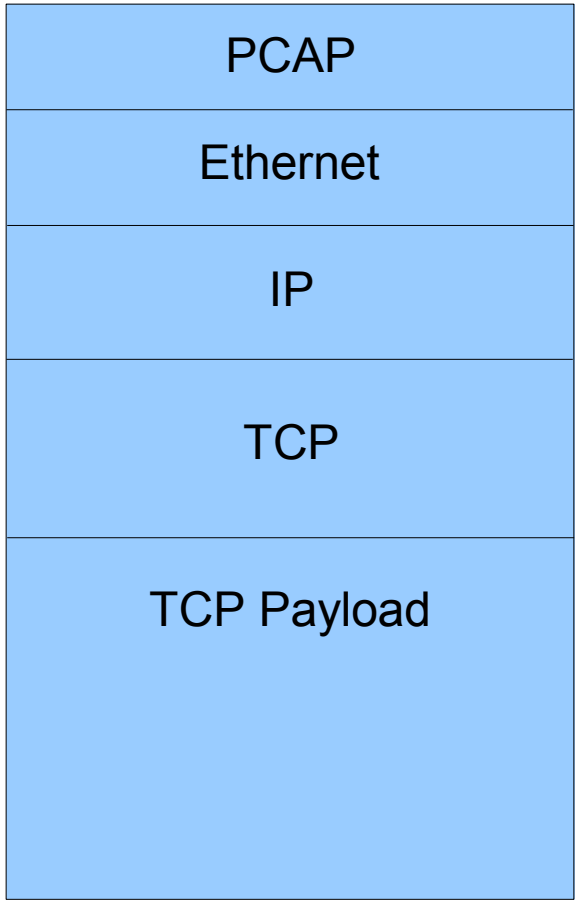
- Captured packets can be written to disk to create a trace
  - Packets are in chronological order
  - Capture format header is retained on each packet
- Analysis is repeatable
  - Errors in analysis technique can be corrected
  - Interesting behaviour can be investigated further
- Collaboration with other researchers
  - WITS - http://www.wand.net.nz/wits/
  - Datcat - http://www.datcat.org/
  - CRAWDAD - http://crawdad.cs.dartmouth.edu/

Packet Traces

- Full payload capture
  - All of the packet is retained
  - Simple to implement
  - Investigating application behaviour is easier
- Disadvantages
  - Privacy concerns due to capturing user data
  - Trace files are extremely large

Packet Traces

- Example – blue area represents the captured data

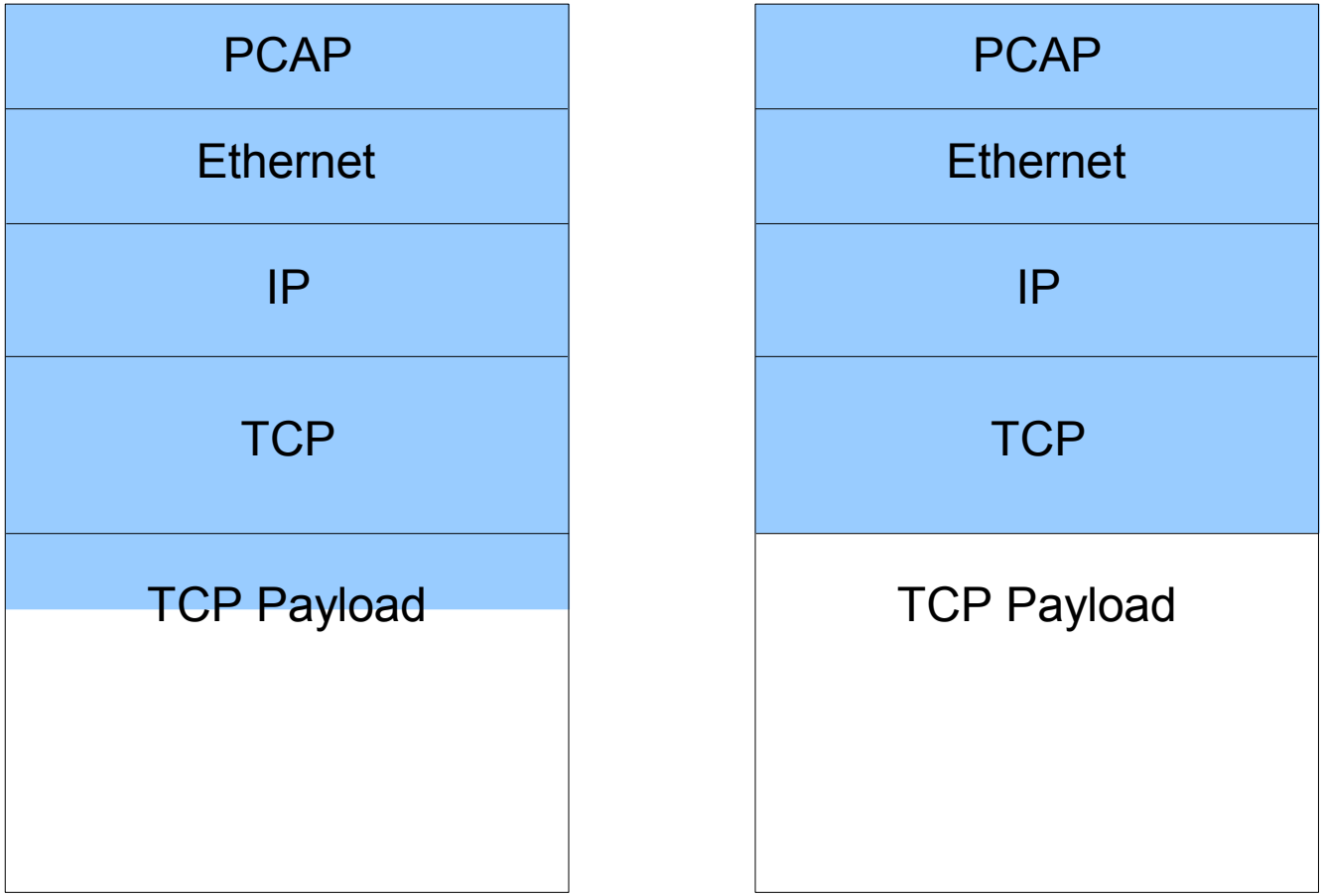


Packet Traces

- Header capture
  - Captured packets are truncated (snapped) to remove user payload
    - Fixed length snapping vs header-based snapping
  - Traces require less space
  - Most pertinent information is in the headers

Packet Traces

- Example – fixed length (left) vs header snapping (right)



Passive Analysis

- Simple examples
  - Counting packets or bytes
  - Examining TCP/IP headers
- Advanced ideas
  - TCP object extraction
  - Application analysis, e.g. HTTP
  - Visualisation

Passive Analysis

- Real-time
  - Capture process reads straight off a network interface
  - Performance is critical
  - Most practical applications are real-time
    - e.g. anomaly detection, visualisation

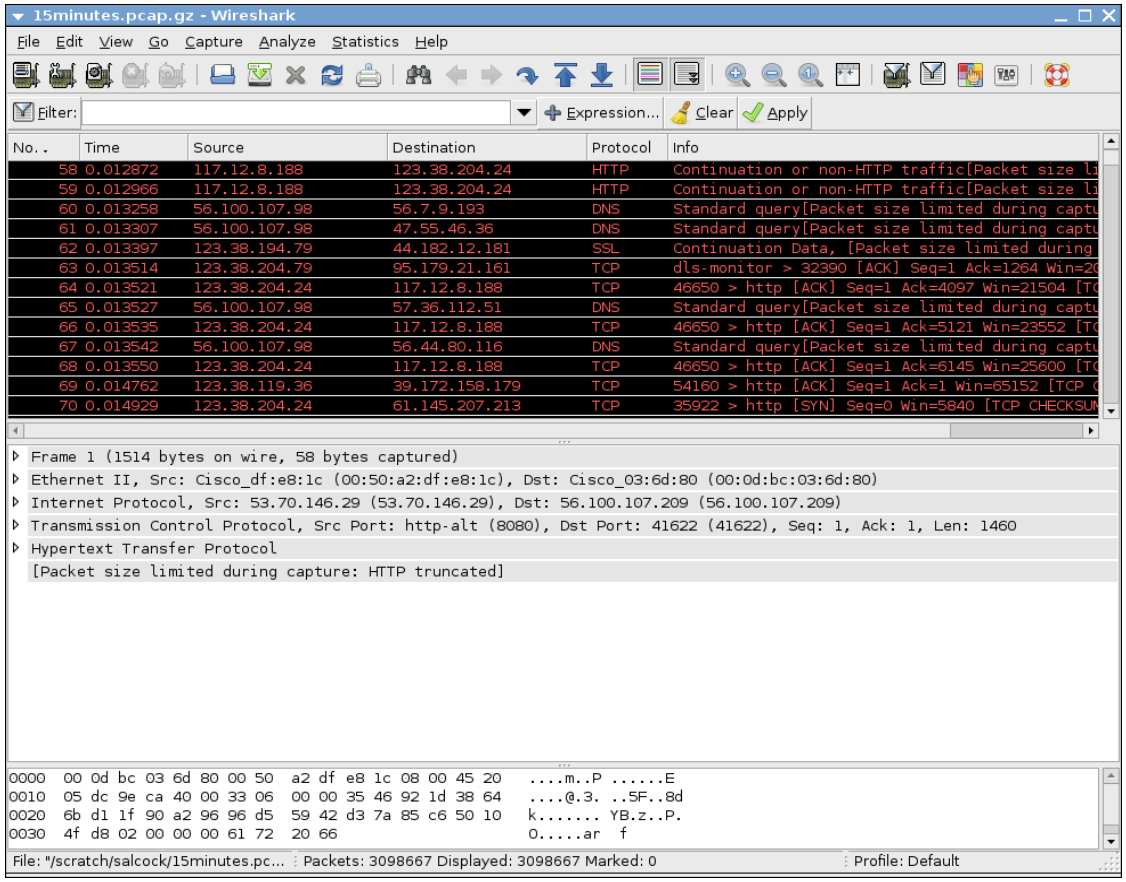
- Off-line
  - Replace the capture step with reading from a trace file
  - Best for resource-intensive analysis
  - Many research applications can be done solely off-line

Passive Analysis

- Use existing tools
  - Examples: wireshark, tcptrace
  - Designed to perform a specific set of tasks
- Develop new analysis tools
  - Particularly common for research applications

Passive Analysis

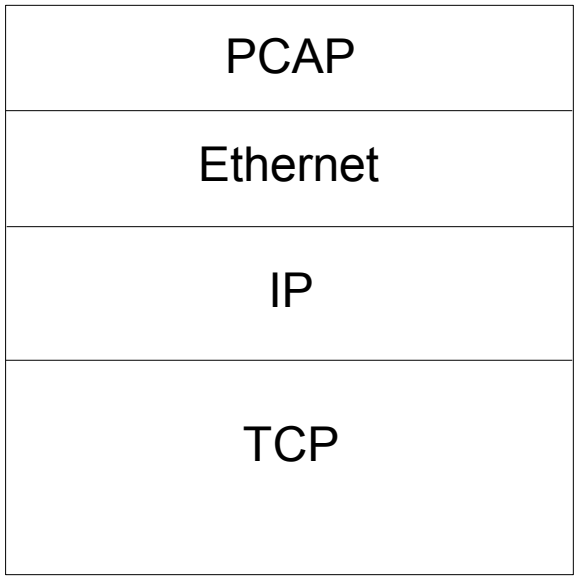
- Example of an existing tool: wireshark



Development Issues

- Aim is to count packets using TCP port 80
  - Should be easy, right?

- Standard TCP/IP packet captured using PCAP from an Ethernet link



Development Issues

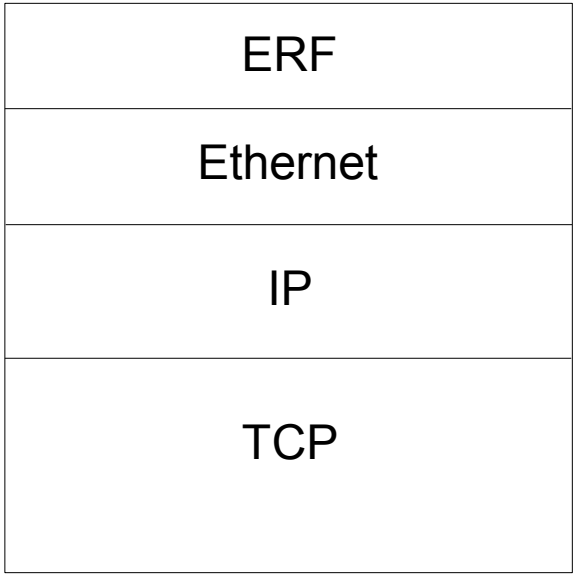
- The general case is simple
  - Step through the preceding headers to reach the TCP header
    - Be careful of the variable length IP header!
  - Check the port numbers inside the TCP header
  - Increment counter if necessary
  - Move onto next packet
    - PCAP header will tell us how far we need to skip ahead

Development Issues

- What about the special cases?
  - The packet isn't a TCP packet, e.g. UDP or ICMP
  - The packet isn't an IP packet, e.g. ARP
  - The packet was truncated before the TCP header
  - The packet was truncated part-way through the TCP header
  - The packet was fragmented
    - TCP header could be in a different fragment
  - Note that this is not a comprehensive list!

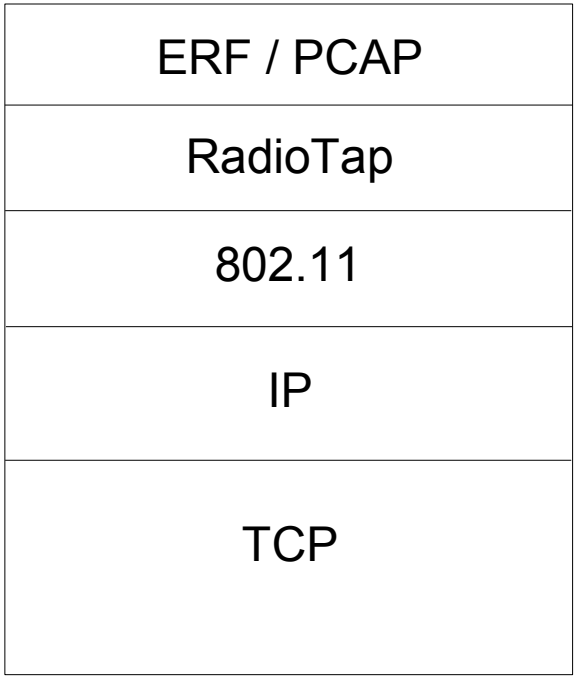
Development Issues

- Try our analysis on another trace set
  - What if the traces use the ERF format instead of PCAP?
  - Update program to support new capture format



Development Issues

- Applying our analysis to a trace from a wireless link
  - Need to add code to detect and skip over 802.11 headers
  - Still need to keep our old code for Ethernet as well

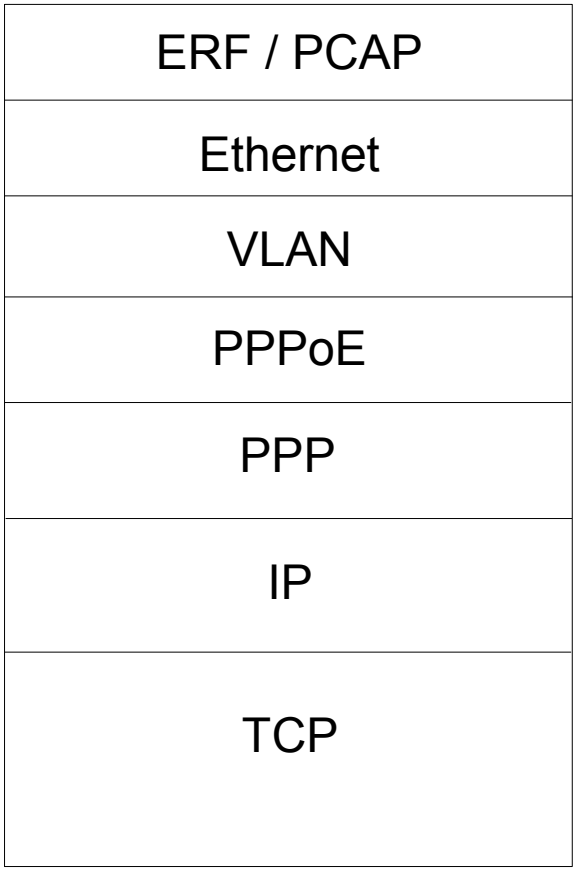


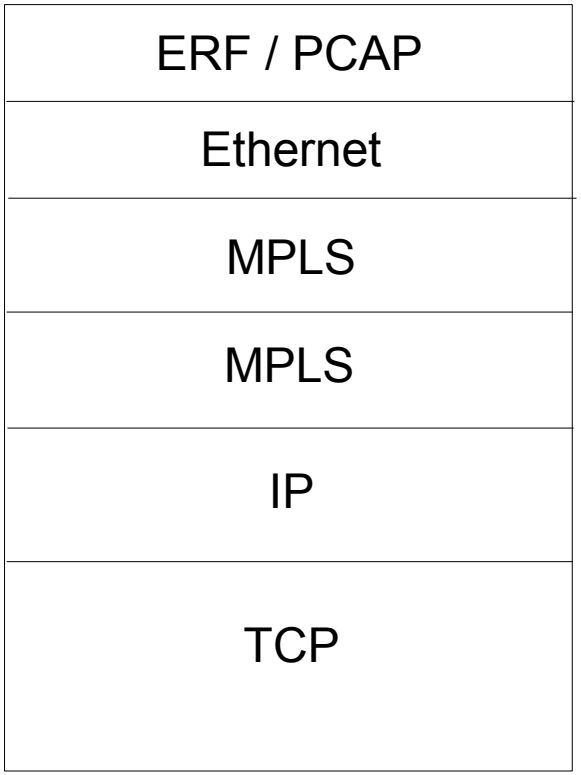
Development Issues

- Wireless introduces an entirely new set of problems
  - 802.11 header varies in length
  - RadioTap header is not always present
    - Might be an entirely different header altogether, e.g. Prism
    - Might be no header at all before the 802.11 header
  - Frame corruption
  - Fragmentation can also occur at the 802.11 level
  - Once again, not a comprehensive list

Development Issues

- Other link layer protocols





Development Issues

- What about running our analysis on a live capture?
  - Live capture APIs add an extra level of complexity
  - Buffer management
  - Code needs to be efficient

Summary

- Developing a portable analysis tool is very difficult
  - Subtle differences between each format header
  - Link layer encapsulation is a nightmare
  - Live capture formats are particularly difficult to code
  - Huge variety of special cases and banana skins

- Wouldn't it be nice if someone...
  - did all the tricky programming for us
  - wrapped it in a nice API that abstracted away all the nasty details
  - gave it all away for free
  - was willing to show you how to use it

Introducing Libtrace

- Packet capture and analysis library
- Developed by WAND (University of Waikato)
  - Written in C, but we have added Ruby bindings
- Design aimed to resolve all these issues
  - Make passive analysis simple and reduce code replication
- Supports reading and writing of trace files

Libtrace Features

- Capture format agnostic
  - The same libtrace program works on any supported capture format
  - No difference between live and off-line capture formats
  - Developmental advantages
    - Analysis programs can be tested off-line before running live

Libtrace Features

- Protocol details are dealt with internally
  - Direct access to each protocol layer
    - e.g. `trace_get_tcp()` jumps straight to the TCP header
  - Handles a variety of link layer protocols including ...
    - Ethernet
    - 802.11 wireless
    - VLAN
    - MPLS

Libtrace Features

- Bundled with a suite of tools to perform common tasks
  - Dumping packet contents
  - Trace splitting and filtering
  - Converting between trace formats
  - Statistical reports

History

- Began as a library for live analysis from Waikato capture
  - Reading ERF packets received via TCP
- Extended to read from trace files
  - Using zlib to read compressed trace files
- Writing analysis code was involving lots of copy / paste
  - Add commonly used functions to the library
- Added support for PCAP using libpcap

History

- Libtrace 2 (2004)
  - Began as a major API clean-up
- Many features were added throughout its lifetime
  - Support for writing traces
  - Libtrace tools
  - Support for many new protocols and link types
  - Support for new capture formats, e.g. Auckland trace formats
  - Easy conversion between capture formats
  - Packet dumping library
  - Internal error system, e.g. trace_perror()

History

- Libtrace 3 (2006)
  - Zero-copy for live capture formats
    - Significant performance improvements for live capture
  - Native pcap file support
  - Added configuration system for captures
    - Snap length, BPF filters, etc.
  - Decode IPv6 headers
  - Comprehensive wireless support
  - Even better error handling
  - Cleaned up API (again!)
  - Polished libtrace tools

Present Day

- Current libtrace version is 3.0.5
  - Available from http://research.wand.net.nz/software/libtrace.php
- Open source
  - GPL license
- Operating Systems
  - Linux
  - FreeBSD
  - MacOS
  - Windows is unsupported, but we have built DLLs in the past

Installing Libtrace

- Requirements
  - automake-1.9 or later
  - libpcap-0.8 or later
  - flex and bison
- Strongly recommended
  - zlib-dev
- Required for DAG capture
  - DAG libraries (provided with DAG hardware)

Installing Libtrace

```
./configure
make
make install
```

- Installs to /usr/local/lib by default
  - Append `--prefix=DIR` option to `./configure` to change

- `./configure --help` for a full list of options

Libtrace URIs

- URIs are used to tell libtrace programs the format and location of a trace or live capture

- All URIs resemble:

  <format>:<location>

Supported Capture Formats

Format	Base URI	Example	Write
DAG	dag	dag:/dev/dag0	No
ERF	erf	erf:/trace/example.erf.gz	Yes
PCAP interface	pcapint	pcapint:eth0	Yes
PCAP file	pcapfile	pcapfile:/trace/example.pcap.gz	Yes
Native Linux	int	int:eth0	Yes
Native BSD	bpf	bpf:eth0	No
TSH	tsh	tsh:/trace/example.tsh.gz	No
RT protocol	rt	rt:localhost:4500	No
Legacy ATM	legacyatm	legacyatm:/trace/example.atm.gz	No
Legacy Ethernet	legacyeth	legacyeth:/trace/example.eth.gz	No
Legacy PoS	legacypos	legacypos:/trace/example.pos.gz	No
ATM Cell Header	atmhdr	atmhdr:/trace/example.atmhdr.gz	No
FR+	fr+	fr+:/trace/example.fr.gz	No

BPF Filters

- Libtrace supports using BPF expressions to filter traffic
  - Commonly used in PCAP (tcpdump)

- Examples
  - “tcp port 80”
  - “src host 192.168.2.1 and src host 192.168.2.2”
  - “less 100”
  - `man tcpdump` for more details

WAND Network Research Group
Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

www.crc.net.nz
www.wand.net.nz
www.waikato.ac.nz

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato