Internet2 E2E piPEs Project Update: Reaching the First/Last Mile (FLM)

Richard Carlson
Internet2
RIPE 48
May 5, 2004

rcarlson@internet2.edu

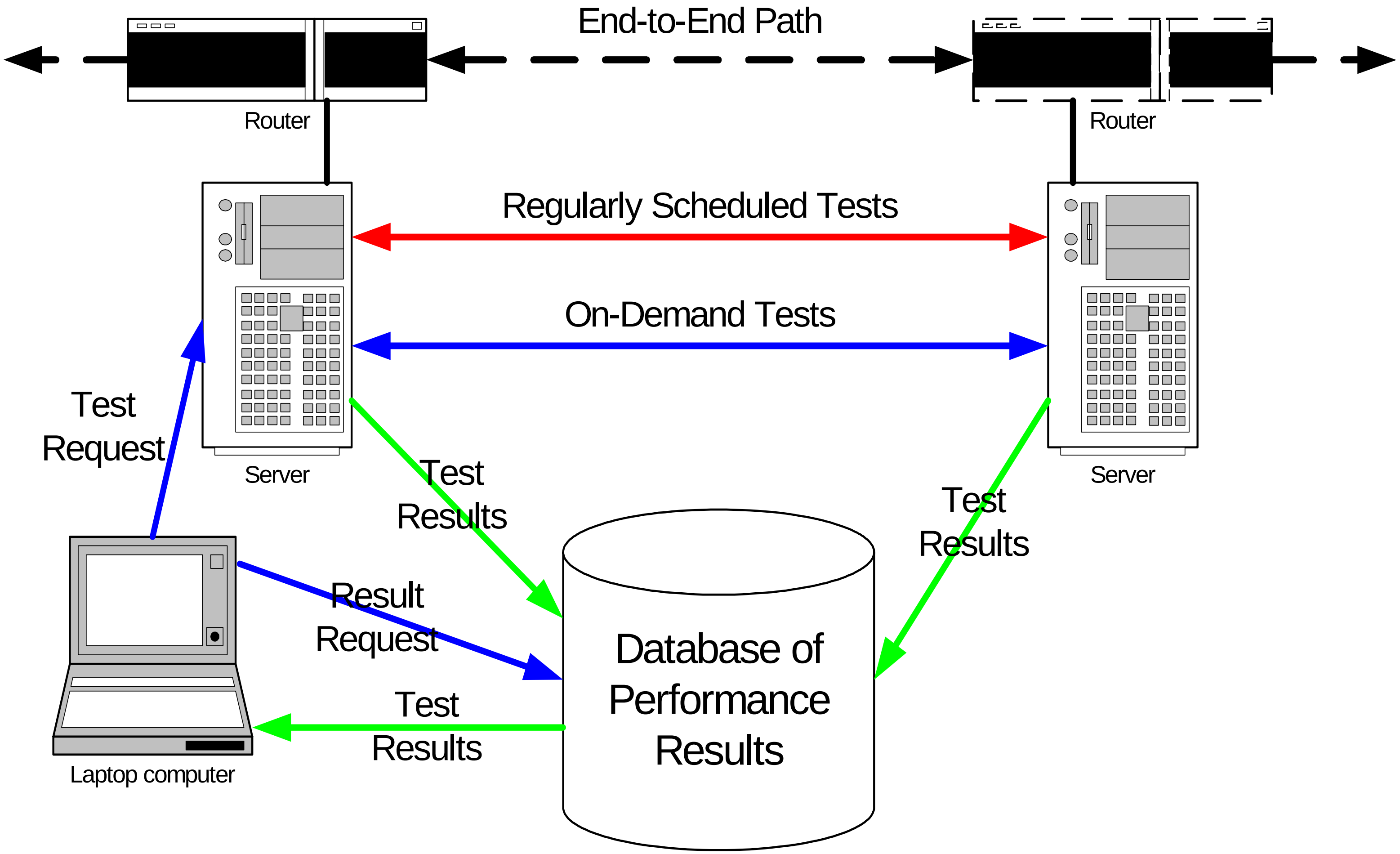
Internet2 E2E piPEs

- Project: End-to-End Performance Initiative Performance Environment System (E2E piPEs)
- Approach: Collaborative project combining the best work of many organizations, including DANTE/GEANT, EGEE, GGF NMWG, NLANR/DAST, UCL, Georgia Tech, etc.

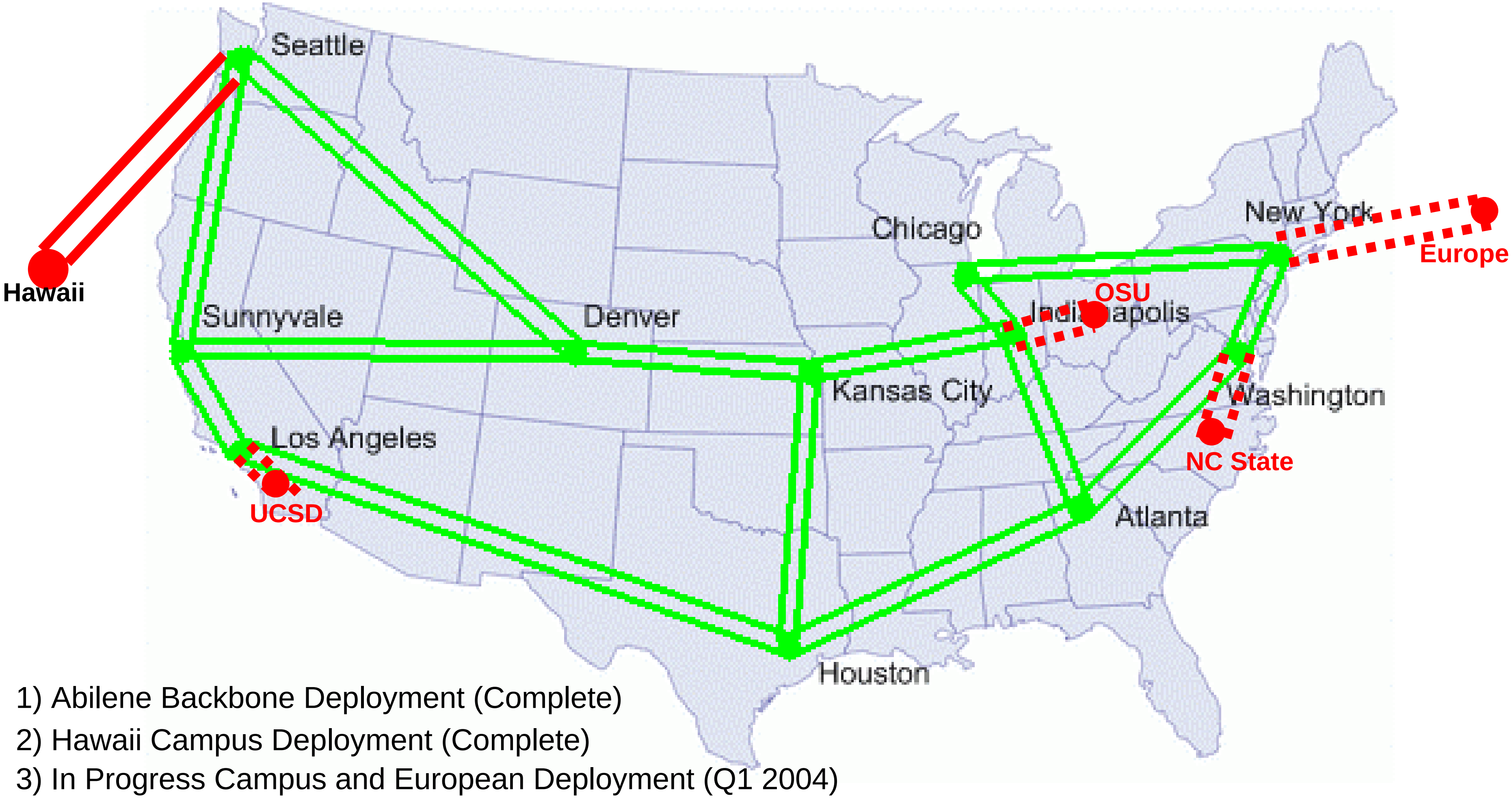
Internet2 E2E piPEs Goals

- Enable end-users & network operators to:
  - Determine E2E performance capabilities
  - Locate E2E problems
  - Contact the right person to get an E2E problem resolved
  - Enable remote initiation of partial path performance tests
- Interoperable with other performance measurement frameworks
  - Make partial path performance data publicly available
  - GGF standard schema for request/response messages
  - Standard security mechanisms for AAA

Measurement Infrastructure Components



piPEs Deployment

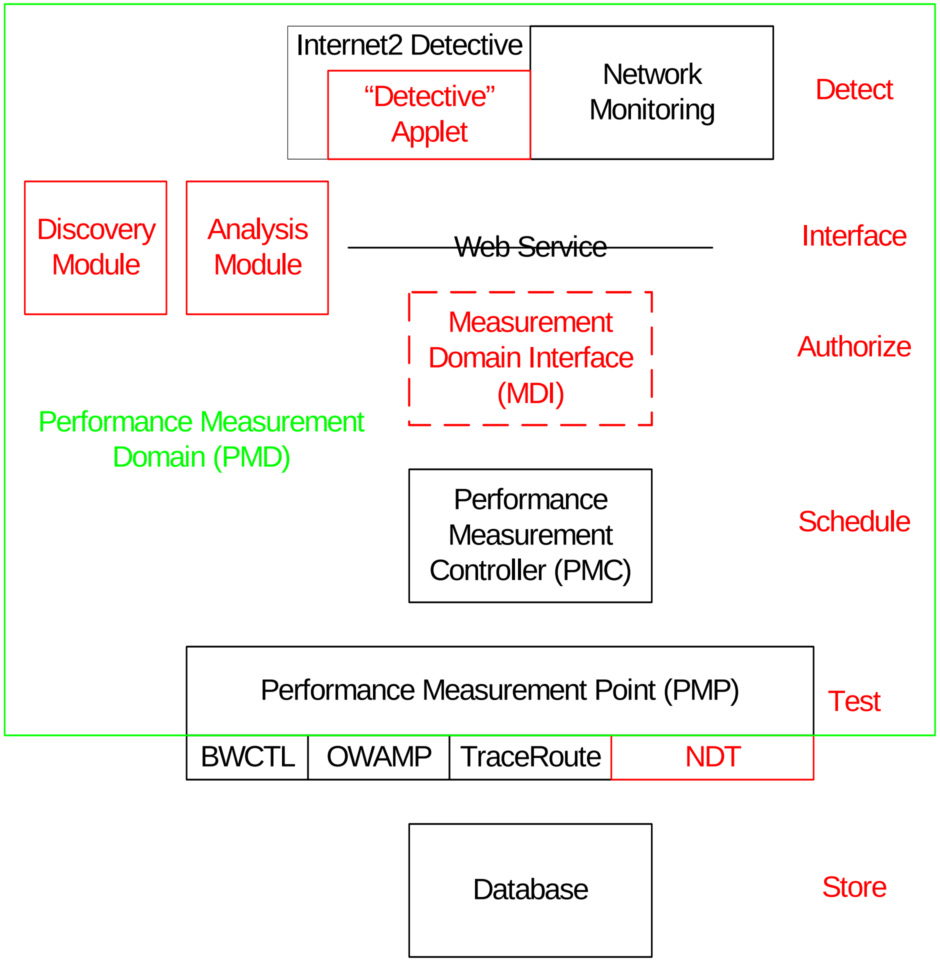


1) Abilene Backbone Deployment (Complete)

2) Hawaii Campus Deployment (Complete)

3) In Progress Campus and European Deployment (Q1 2004)

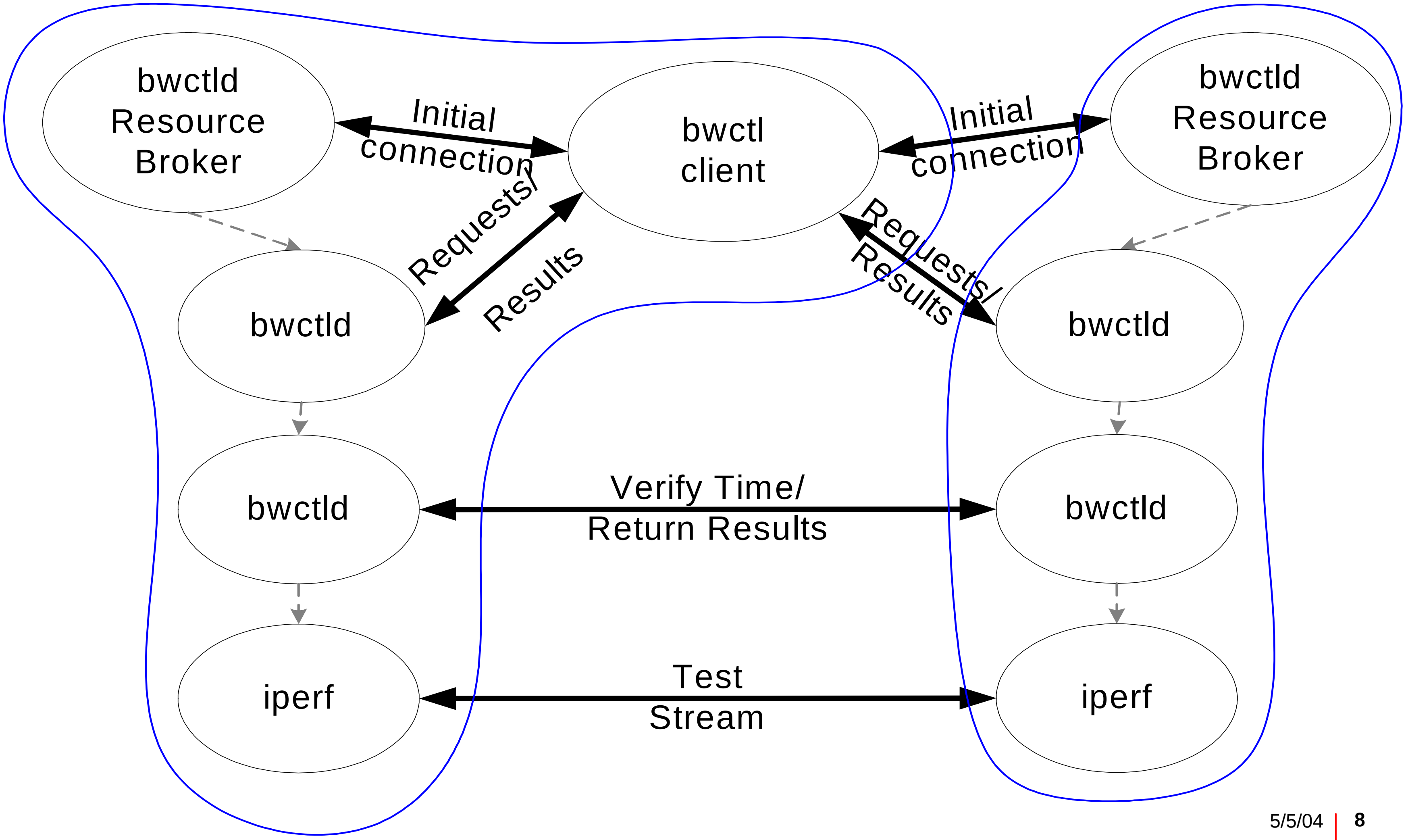
Measurement Software Components



BWCTL Design Goals

- Bandwidth Control Server
- Wrapper for Dast Iperf tool
- Performs scheduled tests between 11 peers
- Supports on-demand tests between peer nodes

Architecture



Specific difficulties

UDP

- Iperf doesn’t always send at requested rate
- Iperf sender hangs (likely Linux/iperf interaction – could be due to signal handling of the bwctl level)
- End of session is difficult to detect, which is problematic for a “scheduled” timeslot
- Iperf sometimes takes large amounts of time to finish

Specific difficulties

TCP

- Large pipe to small pipe
  - Launch a large window
  - Test waits until completion
  - Terminate test to remain within schedule
  - □⇒ Sets of incomplete tests to interpret
- Full mesh presents difficulties for window size selection (and other path specific characteristics)
  - bwctl uses the peer to peer server connection to deduce a “reasonable” window
  - If at all possible path specific parameters need to be dynamically configured

Future Possibilities

- Server-less client side for end hosts
- Closer integration with test engine (iperf API?)
  - Better error detection
  - Better timing control (begin and end of test is currently a problem)
- 3-party tests (client not on one of the endpoints)
- Open source development

Availability

- Beta version currently available

  www.internet2.edu/bwctl/

Mail lists:

- bwctl-users
- bwctl-announce

https://mail.internet2.edu/wws/lists/engineering

OWAMP Design Goals

- One-Way-Active-Measurement-Protocol
  - Possible due to growing availability of good time sources
  - Wide deployment of “open” servers would allow measurement of one-way delay to become as commonplace as measurement of RTT using ICMP tools such as ping.
  - Current Draft: draft-ietf-ippm-owdp-07.txt
    - Shalunov,Teitelbaum,Karp,Boote,Zekauskas
    - RFC just released
  - Sample implementation under development

    Alpha code currently available

Abilene OWAMP deployment

- 2 overlapping full meshes (IPv4 & IPv6)
  - 11 measurement nodes = 220 ongoing tests
  - UDP singletons
  - Rate: 10 packets/second*
  - Packetsize: (32 byte payload)*
  - Results are continuously streamed back to “Measurement Portal” for long-term archive and data dissemination (Near real-time)

OWAMP Errors

- Preliminary Findings:
  - Min error estimates look to be in the 55-60 usec range.
  - Serialization Delay: ~5usec x 2
  - Get Timestamp: ~15usec x 2
  - Additional error is:
    - Time from userland “send” to 1st byte hits the wire
    - Time from kernel has packet to userland “recv” returns
    - Potentially recv process data processing before calling “recv”

OWAMP implementation status

- Sample implementation
- http://e2epi.internet2.edu/owamp/

- Alpha Release ver 1.6c:
  - No “policy”
  - No authentication/encryption
  - Tested on FreeBSD & Linux

NDT Design Goals

- Develop “single shot” diagnostic tool that doesn’t use historical data
- Measure performance to users desktop
- Combine numerous Web100 variables to analyze connection
- Develop network signatures for ‘typical’ network problems
- Provide a single entry point into measurement domain

Web100 Project

- Joint PSC/NCAR project funded by NSF
- ‘First step’ to gather TCP data
  - counters, timers, events, retransmissions
- Requires patched Linux kernel
  - RPM release also available
- Preliminary auto-tuning functions to improve application performance on a per-flow basis

Web Based Performance tool

- Operates on Any client with a Java enabled Web browser
- Web100 enhanced server
- What it can do
  - Positively state if Sender, Receiver, or Network is operating properly
  - Provide accurate application tuning info
  - Suggest changes to improve performance

Web base Performance tool

- What it can’t do
  - Tell you where in the network the problem is
  - Tell you how other servers perform
  - Tell you how other clients will perform

Configuration Signatures

- Duplex Mismatch Detection
  - Good results in Campus environment
- Faulty Hardware/Link
  - Few reports, needs more work
- TCP buffer size and BW*Delay product reported
  - Window scaling should work now

Performance Signatures

- Bottleneck Link Type
  - Uses packet dispersion techniques
- Link Duplex setting
  - Needs more work
- Normal Congestion
  - Needs more work

Current Deployment

- Public Servers (6)
  - Argonne National Laboratory – Argonne IL
  - Swiss Education and Research Network (SWITCH)
  - University of Michigan – Flint, MI
  - University of California - Santa Cruz, CA
  - Stanford University – Palo Alto, CA
  - Thomas Jefferson National Accelerator Facility – Newport, VA

  - StarLight (REN only) – Chicago, IL
  - Abilene Federation being deployed

Different HW same Network port

- 10 Mbps NIC
  - Throughput 6.8/6.7 Mbs send/receive
  - RTT 20 ms
  - Retransmission/Timeouts 25/3
- 100 Mbps NIC
  - Throughput 84.6/86.5 Mbs send/receive
  - RTT 10 ms
  - Retransmission/Timeouts 0/0

Effect of Faulty HW & Congestion

100 Mbps FD

Ave Rtt	%loss	loss/sec	Speed	
5.41	0.00	0.03	94.09	Good
14.82	0.00	0.10	33.61	Congestion
1.38	0.78	15.11	22.50	Bad NIC
6.16	0.00	0.03	82.66	Bad reverse
10 Mbps				
72.80	0.01	0.03	6.99	Good
8.84	0.75	4.65	7.15	Bad NIC

Link Detection Algorithm

- Uses Packet-Pair timing
  - Small Libpcap program captures data
  - Timing taken for each transmit/receive pair
  - Results quantized into unique bins
  - Statistical analysis on resulting bin counts

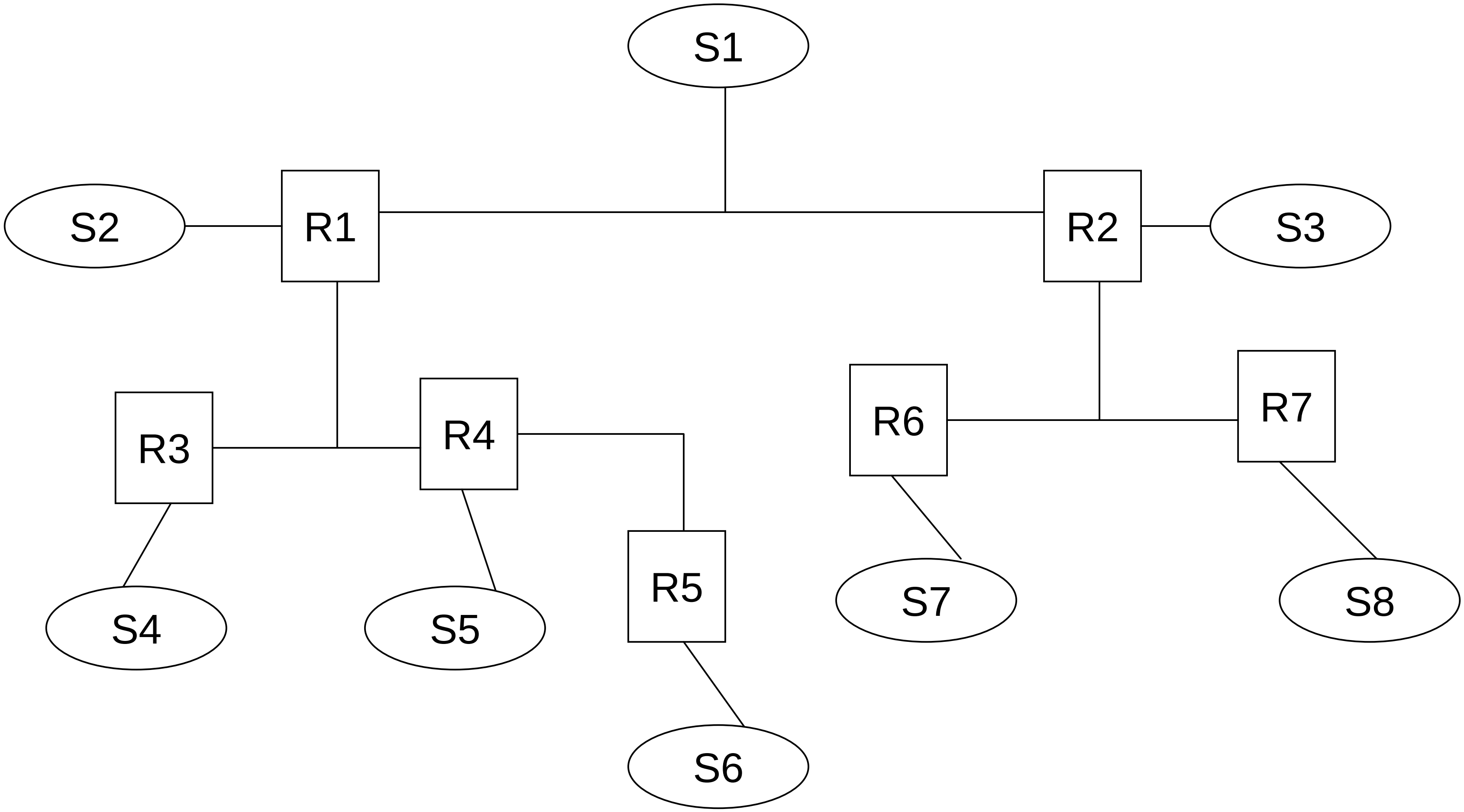
NDT in Federated Mode
Basic Assumptions

- A collection of testing servers form a measurement domain
- Test requests come from ‘outside’ the measurement domain
- Users can test to the ingress or egress point of the measurement domain
- User doesn’t know which server is at ingress or egress

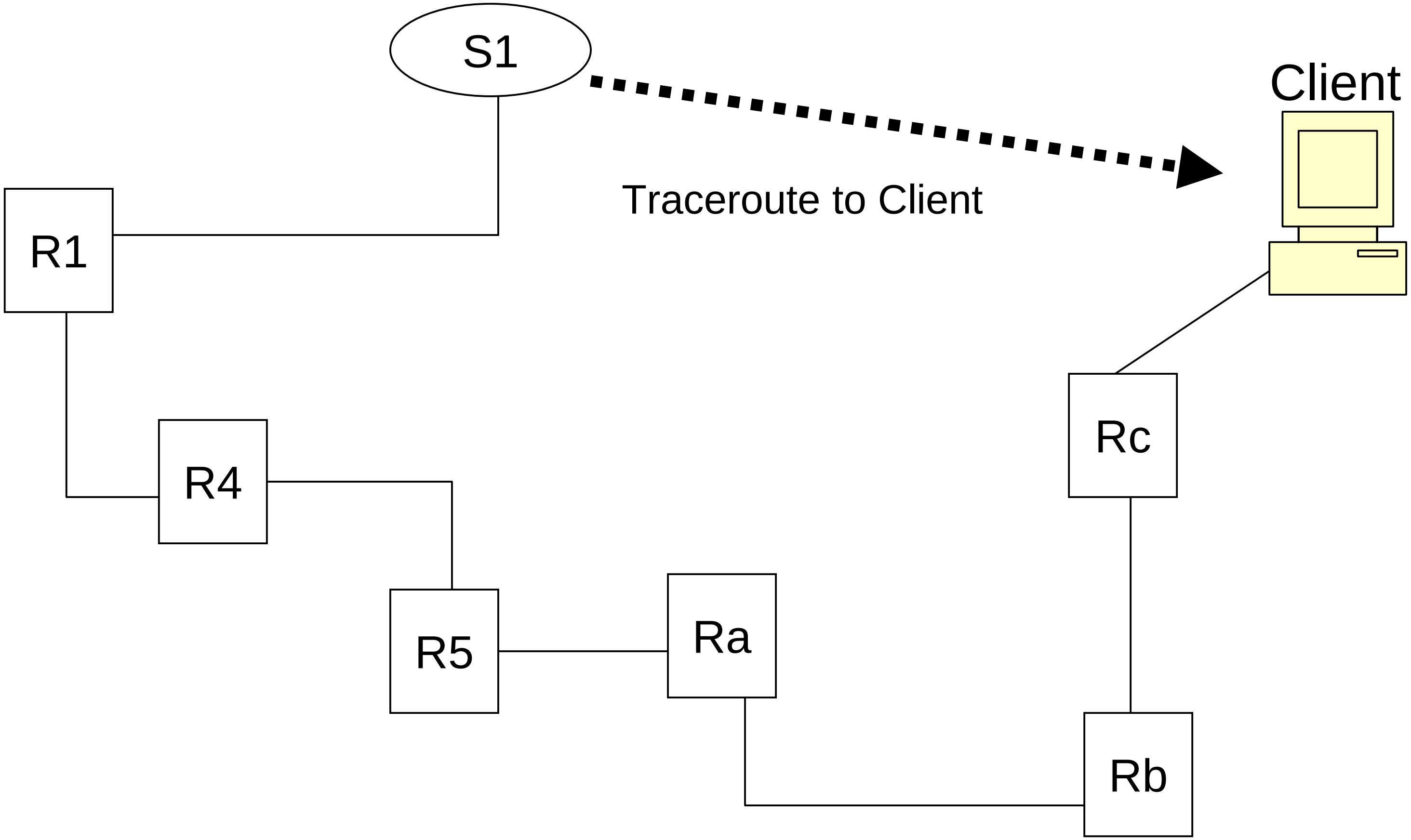
piPEs Design Choice

- All information for ingress testing is supplied in the initial connection request
  - Packet goes to testing engine, indicating intent
  - Packet contains client IP address
  - Most problems are at/near the host
- Default initial test to ingress node
  - Destination information can be supplied for egress testing
  - Testing service will support both, but will prefer ingress point testing

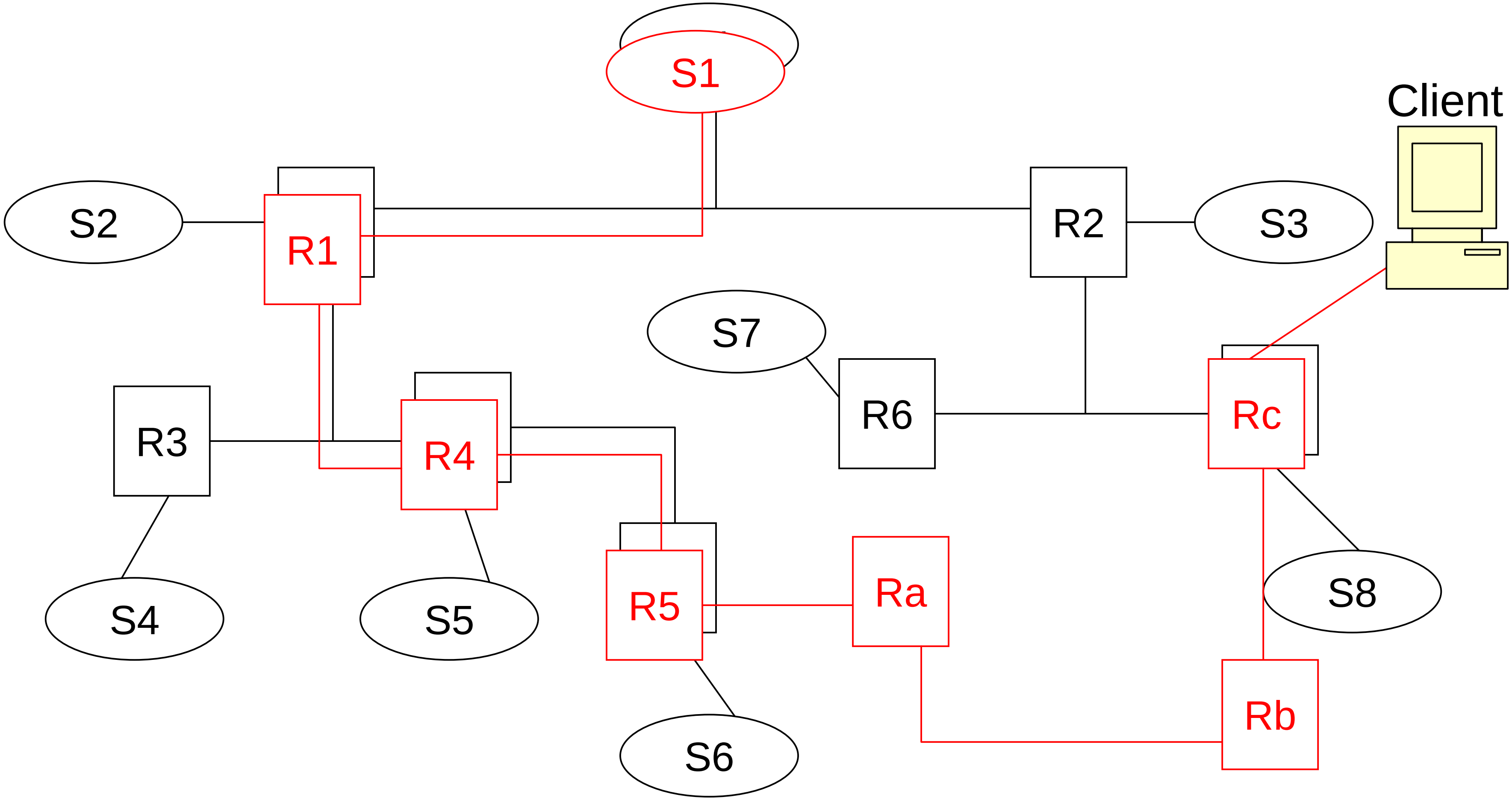
Sample Traceroute Tree map



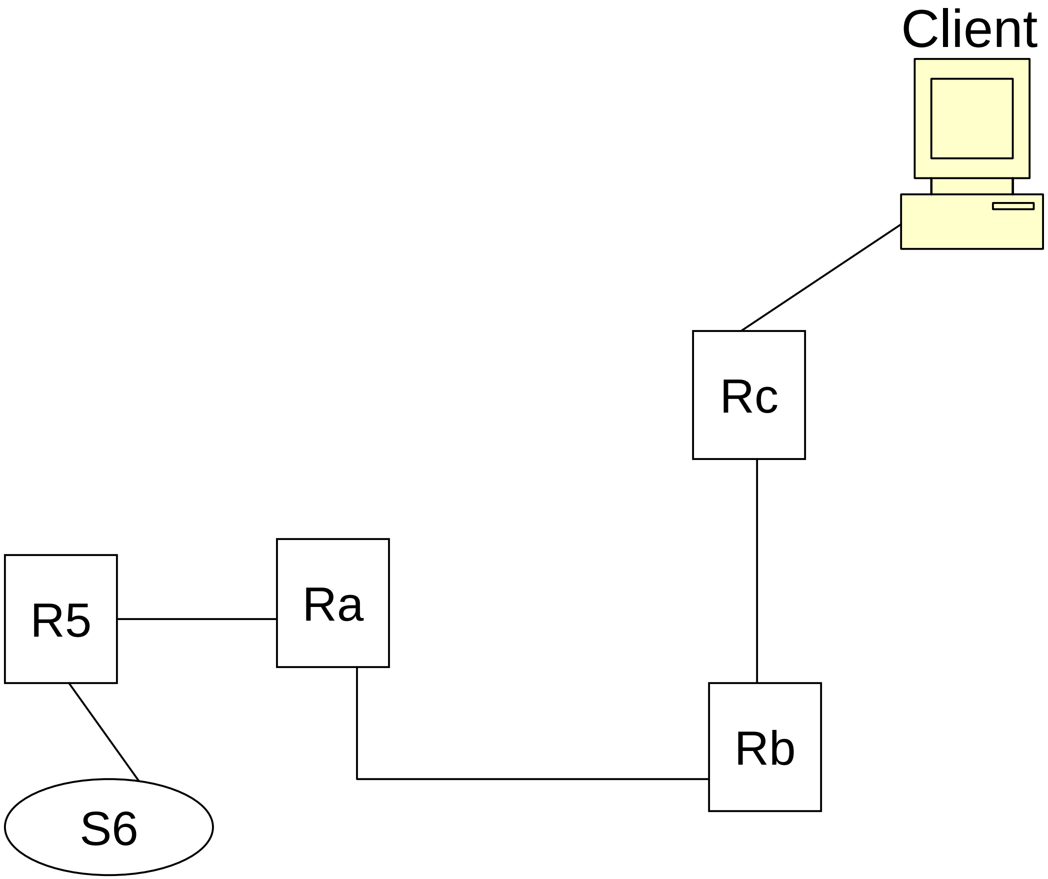
Sample Traceroute from S1 to Client



Sample Traceroute Tree map



Client re-directed to S6 for test



piPEs Approach – Picking the Another server

- Allow client to find egress server
  - Allow performance testing
  - Client provides destination name/address
  - Ingress server will use traceroute map to find egress server and re-direct client
- Allow client to manually select any server in the cloud

Initial Deployment

- Start deployment of NDT federation in Abilene core
  - Requires Web100 enhanced Linux server
  - Generic name “ndt-city”
    - http://ndt-seattle.abilene.ucaid.edu:7123
  - Do we want/need a single central name?

Additional info

- StarLight server now operational
  - http://ndt.sl.startap.net
  - No access from commodity Internet
- Command Line version of client code under development (web100clt)
  - Compiles and runs under Linux, FreeBSD, and Windows (cygwin)

Demo

http://miranda.ctd.anl.gov:7123

http://lg.net.switch.ch/network/performance/web100/tcpbw100.html

Abilene Measurement Domain

- Part of the Abilene Observatory:

  http://abilene.internet2.edu/observatory

- Regularly scheduled OWAMP (1-way latency) and BWCTL (Iperf wrapper) Tests
- Web pages displaying:
  - Latest results http://abilene.internet2.edu/ami/bwctl_status.cgi/TCP/now
    “Weathermap”
    http://abilene.internet2.edu/ami/bwctl_status_map.cgi/TCP/now
  - Worst 10 Performing Links
    http://abilene.internet2.edu/ami/bwctl_worst_case.cgi/TCP/now
- Data available via web service:

  http://abilene.internet2.edu/ami/webservices.html

- End of formal presentation

Feedback

- Are we on the right track? (As conceptualized, would our individual and joint goals meet the needs of the DataTag community?)
- What's missing?
- What is of particular importance?

Data Collection / Correlation

- Collection Today:
  - Iperf (Throughput)
  - OWAMP (1-Way Latency, Loss)
  - SNMP Data
  - Anonymized Netflow Data
  - Per Sender, Per Receiver, Per Node Pair
  - IPv4 and IPv6
- Collection in the Future
  - NTP (Data)
  - Traceroute
  - BGP Data
  - First Mile Analysis
- Correlation Today:
  - “Worst 10” Throughputs
  - “Worst 10” Latencies
- Correlation in the Future:
  - 99th Percentile Throughput over Time
  - Throughput/Loss for all E2E paths using a specific link
  - Commonalities among first mile analyzers
  - Sum of Partial Paths vs. Whole Path

Data Analysis

- Analysis Today:
  - Throughput over Time
  - Latency over Time
  - Loss over Time
  - Worrisome Tests? (Any bad apples in “Worst Ten”?)
  - “Not the Network” (If “Worst Ten” is good enough)

- Analysis in the Future:
  - Latency vs. Loss
  - How good is the network?
  - Do common first mile problems exist?
  - Does a link have problems that only manifest in the long-haul?
  - Is the network delivering the performance required by a funded project?

Data Discovery / Interoperability

- Discovery in the Future:
  - Where are the measurement nodes corresponding to a specific node?
  - Where are the test results for a specific partial path?

- Interoperability in the Future:
  - Can I have a test within or to another measurement framework?
  - Can I have a measurement result from within or to another measurement framework?

Problem Statement

- Users want to verify available bandwidth from their site to another.

Methodology

- Verify available bandwidth from each endpoint to points in the middle to determine problem area.

Typical road blocks

- Need software on all test systems
- Need permissions on all systems involved (usually full accounts*)
- Need to coordinate testing with others *
- Need to run software on both sides with specified test parameters *

(* bwctl was designed to help with these)

Functionality

Bwctl client application makes requests to both endpoints of a test

- Communication can be “open”, “authenticated”, or “encrypted”
- Requests include a request for a time slot as well as a full parameterization of the test
- Current client is limited in that one of the endpoints must be the localhost, but the protocol is designed to support 3 parties
- Same “basic” command line options as iperf (some options limited or not implemented.)

Functionality

bwctld on each test host

- Accepts requests for “iperf” tests including time slot and parameters for test
- Responds with a tentative reservation or a denied message
- Reservations by a client must be confirmed with a “start session” message
- Resource “Broker”
- Runs tests
- Both “sides” of test get results

NDT Benefits

- End-user based view of network
- Can be used to identify performance bottlenecks (could be host problem)
- Provides some ‘hard evidence’ to users and network administrators to reduce finger pointing
- Doesn’t rely on historical data

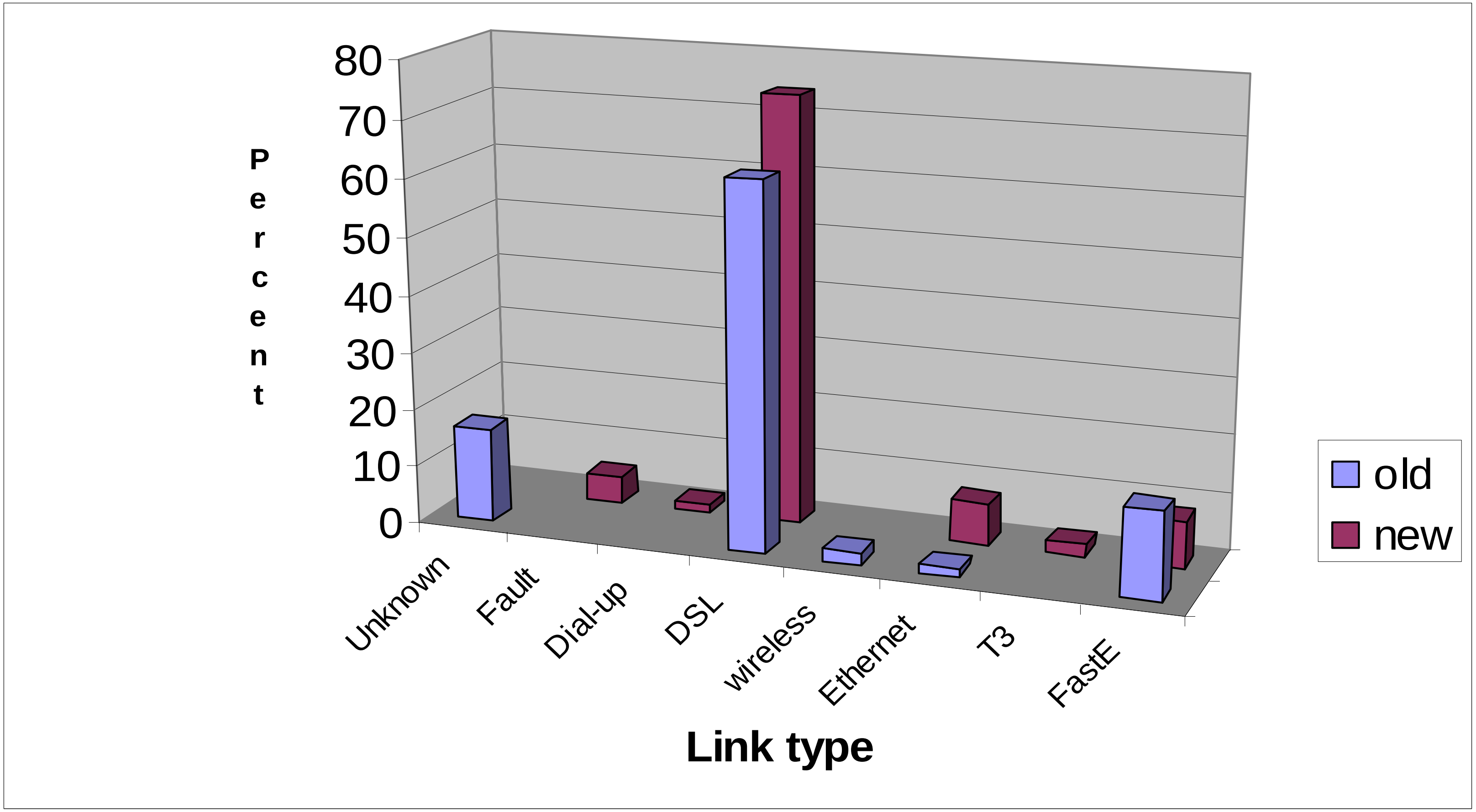
Availability

- Tools available via anonymous ftp from: achilles.ctd.anl.gov/pub/web100 directory
  - Current version is ndt-3.0.9.tar.gz
  - Contains source code and executables
- Email discussion list <ndt@anl.gov>
  - Majordomo list <majordomo@achilles.ctd.anl.gov>
  - subscribe ndt

Results and Observations

- Changing desktop effects performance
- Faulty Hardware identification
- New Link Detection algorithm & preliminary results
- Mathis et.al formula fails
- Usage statistics
- Demo

Comparison between old and new link type detection



Ingress point Benefits

- Closer to client (user’s desktop)
- Shorter network path, fewer links to analyze
- Reduces test traffic over network core
- Better for finding configuration problems with client host/network

Egress point Benefits

- Closer to destination
- Approximates the path an application will use
- Better for finding E2E performance problems

Prototype implementations

- Modified NDT web server to:
  - Perform server discovery process
  - Dynamically generates re-direct page
- Modified NDT testing engine to interoperate with other piPEs testing functions (BWCTL, OWAMP)
  - Schedules multiple requests in FIFO manner
  - Will interact with meta-scheduler

Disclosure/Disclaimer

- This work was supported (in part) by the Office of Science, U.S. Department of Energy under Contract W-31-109-ENG-38
- Packet-Pair work was supported by the Cisco University Research Program Work-for-Others Contract P-03008

Sample Comparison between map and path from S1 to client

- Traceroute to client
  - S1
  - R1
  - R4
  - R5
  - Ra
  - Rb
  - Rc
  - Client
- Traceroute Map
  - S1
  - R1
  - R4
  - R5
  - S6

Obtaining the test results

- Runs 10 sec test from Client to Server
  - no diagnostic data collected
- Runs 10 sec test from Server to Client
  - Web100 diagnostic data collected at end of test
- Prints out summary status message
  - Link speed and duplex
  - Informational or Warning messages

Analyzing the test results

- Statistics button
  - Send and Receive throughput achieved
  - Details for 5 configuration tests (link type, duplex mode, congestion, excessive errors, duplex mismatch condition)
  - Throughput limits section (%S-R-N limited, RTT, %loss, %out-of-order)
  - ‘Tweakable’ settings (TCP modifications to improve performance)

Analyzing the test results

- More Details button
  - Individual TCP counters collected by Web100
  - Conditional test parameters
  - Throughput analysis section including theoretical limits, bandwidth*delay products, loss rate, and buffer sizes

Analyzing the test results

- Report Problem button
  - Invokes local email client <mailto:>
  - Automatically inserts collected data into body of email
  - Provides “comment” section for user feedback
- Server logs all counter variables used for condition tests

Iperf as the “tester”

- Well known – widely used
- Level of integration
  - Iperf server initialization (port number allocation)
  - Iperf error conditions
  - End of session detection

(iperf designed to do diagnostics, we are using it to benchmark)

OWAMP Implementation

- Basically:
  - NTP system call interface
  - Multiple processes for recv/send loops
  - Written as an API to allow one-off implementations

Mathis et.al Formula fails

- Estimate = (K * MSS) / (RTT * sqrt(loss))
  - old-loss = (Retrans - FastRetran) / (DataPktsOut - AckPktsOut)
  - new-loss = CongestionSignals / PktsOut

- Estimate < Measured (K = 1)
  - old-loss 91/443 (20.54%)
  - new-loss 35/443 (7.90%)
  - old agrees with new 26/35 (74.29%)

Server Discovery

- User contacts any server in the piPEs federation
- Server runs discovery process to find ingress server
- Client re-directed to ingress server