Identifying TCP Congestion Control Algorithms Used on the Internet

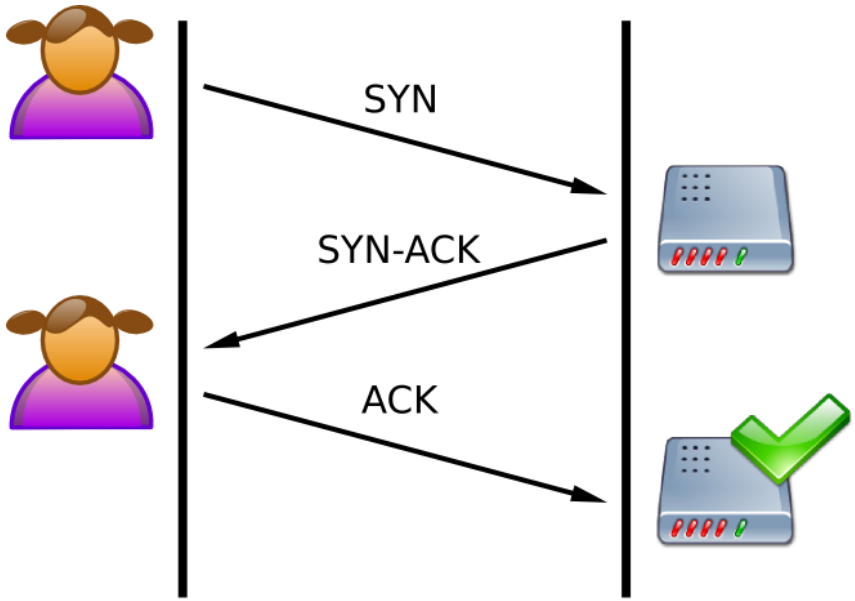
Anuj Sehgal
s.anuj@jacobs-university.de

Jürgen Schönwälder
j.schoenwaelder@jacobs-university.de

Outline

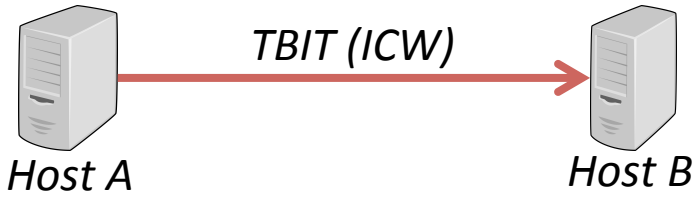
- Introduction
  - Background & motivation
  - TBIT
- Our approach
  - Detecting CWND
  - Identifying CCN
  - Crawler & TBIT Modifications
- Results

TCP on the Internet



- **Widely used on the Internet**
  - Performance tied to TCP
  - Congestion control (CCN) effects performance
- **"Newer" TCP CCN algorithms**
  - Cubic, BIC, etc.
- **What does the TCP CCN makeup on the Internet look like?**
  - Simulations – is it enough to use Reno?
  - TCP performance information useful to developers, ISPs, etc.

TBIT – TCP Behavior Inference Tool



- **Tool designed to probe and characterize TCP implementations**
  - Congestion control
  - Window size
  - End-to-end CCN
  - Selective acknowledgements
  - Explicit congestion notification
  - Time-wait duration
- **Does not require elevated privileges on servers**
  - Uses HTTP

1. Open a raw IP socket and setup a BSD Packet Filter (BPF) to capture all traffic to/from *B*.
2. Setup host firewall on *A* to prohibit packets from *B* reaching kernel.
3. Craft SYN packet with very large receiver window and desired MSS.
4. SYN/ACK received from *B* is forwarded to TBIT process by the BPF.
5. Host *A* requests "/" page from *B*. Host *B* sends data packets for "/".
6. TBIT does not ACK received packets. *B* sends packets fitting within ICW, times out and eventually retransmits.
7. Host *A* sends RST to *B* once retransmission is received.

ICW is the unique data bytes sent by Host B.

Important features of TBIT

- ✓ **The tests can be run against any webserver, as such does not require any special privileges.**
- ✓ **TBIT fabricates its own TCP packets.**
  - Can set custom options in SYN packet.
- ✓ **Traffic generated appears conformant to monitoring entities.**

- X **Cannot detect newer TCP CCN algorithms.**
  - Newer CCN based on Fast Retransmission (like NewReno).
  - All such CCN algorithms identified as NewReno.

Detecting CWND Size

- **No way of measuring Congestion Window (CWND) at server.**
- **We measure the Received CWND (RCWND)**
  - The number of TCP segments a client receives in one RTT.

1. **When a server starts sending data to client, TCP is in *slow start*.**
2. **An ACK causes CWND to change. So we must wait before sending ACKs to detect CWND** *(1s by default)***.**
   - Count packets in buffer, set RCWND as difference between the last buffer size and the newer buffer size.
3. **After CWND passes a threshold, it goes to *congestion avoidance*.**
   - *Slow start* is same in most TCP implementations.
   - *Congestion avoidance* identifies different versions.
4. **Packet loss is typically required to trigger congestion avoidance.**

Detecting CWND Size – During CA

1. **Startup TBIT as explained before, and request the "/" page.**
2. **Client starts receiving packets from remote host.**
3. **Wait for predefined period** (to count CWND – default 1s)**, before sending ACK for all captured packets.**
4. **No more ACKs are sent after packet 62, at this point CWND is typically 64 on the server** (RCWND = 32)**.**
5. **Send two duplicate ACKs for packet 62, forcing *congestion avoidance* on remote host.**
6. **Continue sending ACKs for packets received afterwards and store the measured RCWND.**
7. **Close connection after receiving FIN from remote host.**

Identifying CCN Algorithm

- **First *slow start* RCWNDs must be eliminated.**
  - While,

    $log(2) = log(RCWNDn) - log(RCWDNn-1)$

    is satisfied, slow start phase runs.

- **Curve fitting is then applied to the *congestion avoidance* data.**
  - Curve fitting always performed with a polynomial of fourth degree.
  - Dataset of at least five points during congestion avoidance is needed.
  - If number of RTTs after slow start < *5*, data is insufficient.
- **Linear, quadratic, cubic or higher order *congestion avoidance* classifications.**

Crawler & TBIT Modifications

- **Determine hosts suitable to scan with TBIT (avoid 301, 302 404 errors).**
  - Check if host is accessible at all.
  - Resolve any redirects.
  - Construct a BFS based list of sufficiently sized resources (*>6 kB*).
  - Pretend to be a well-know user agent (*Mozilla/5.0*).
- **Extend TBIT to support GET requests on arbitrary URLs (not just "/").**
  - Also modify user agent string to the one used by crawler.

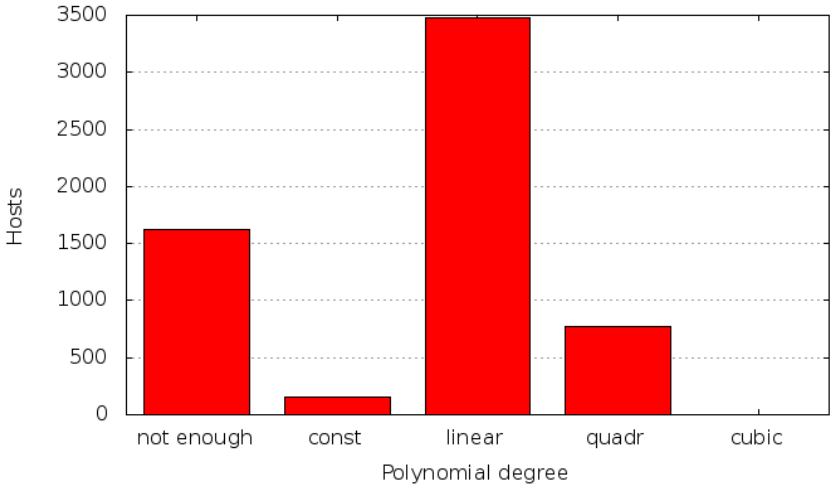
A crawl of 7000 hosts can be processed in under an hour by using 7 parallel processes.

Experimental Setup

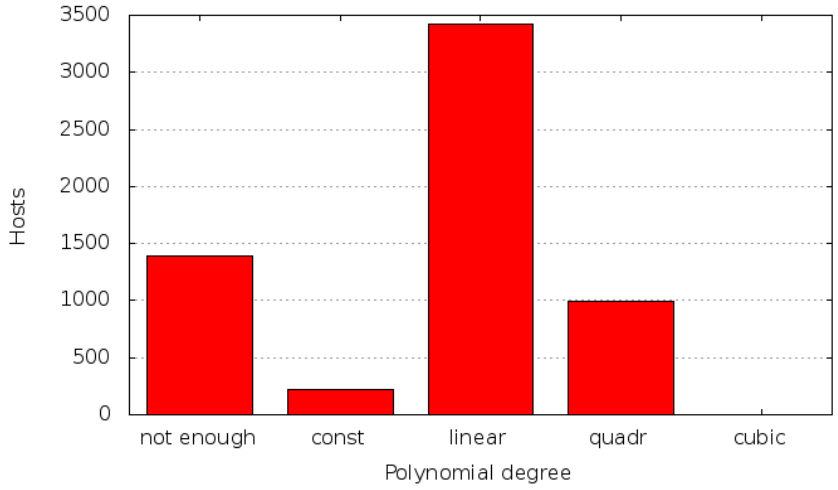
	MSS = 100	MSS = 200	MSS = 600
Hosts with at least one resource	4811	5173	4573
Hosts with at least 5 resources	4465	4801	4213
Selected hosts	**6028**	**6028**	**4861**

- **Crawl a list of top 7000 visited websites, as listed by Alexa.**
  - Minimum resource size of 12 kB for MSS = 100 and MSS = 200 bytes.
  - Minimum resources size of 40 kB for MSS = 600 bytes.
  - Maximum number of resources per host to explore set to 40.
  - Maximum crawl time of 3 mins per host.
- **Crawl performed twice to obtain largest possible dataset.**

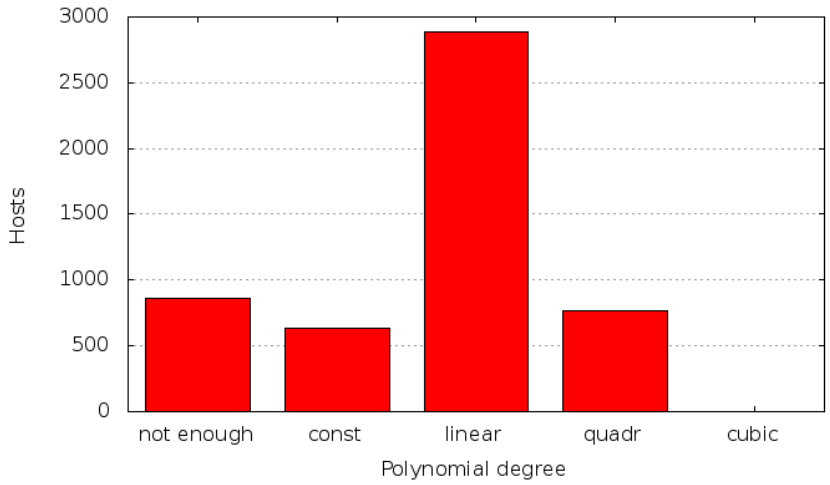
Observations



MSS = 100



MSS = 200



MSS = 600

Polynomial Class	MSS = 100	MSS = 200	MSS = 600
Constant	154 (3%)	218 (4%)	251 (5%)
Linear	3473 (58%)	3426 (57%)	2883 (59%)
Quadratic	774 (13%)	989 (16%)	769 (16%)
Cubic	2 (0.03%)	4 (0.06%)	0 (0%)
Not Enough Data	1625 (27%)	1391 (23%)	958 (20%)

Thank you for your attention!
Questions?