Hybrid Slow Start for High-Bandwidth and Long-Distance Networks

Sangtae Ha and Injong Rhee

PFLDnet 2008
Mar 7, 2008
Slow Start on a High BDP path

- Slow Start
  - Pros: Slow Start probes an available bandwidth very fast (exponentially)
  - Cons: # of packet drops can be well beyond BDP, so it is more problematic for high bandwidth and long distance network

400Mbps, 180ms one way delay, and 100% BDP buffer
Table of Contents

- History of research on improving Slow Start
  - Packet Pair based slow start
  - Modified slow start of Vegas
  - Limited Slow Start (Experimental RFC 3742)
  - Adaptive Start

- Hybrid Slow Start

- Experimental Evaluation

- Conclusion
Packet Pair based slow start

- Frequent over-estimation of bottleneck capacity
- Multiple flows can get the same answers. It can overshoot up to $N \times C$ (N: #flows, C: capacity)
Modified slow start of Vegas

- Temporary queue build-up leads to a premature termination of slow start
Limited Slow Start (Exp. RFC 3742)

- Algorithm during slow start
  - If \((cwnd \leq max_{ssthresh})\) \(cwnd +=\) MSS
  - Else \(K = \text{int}(cwnd/0.5*max_{ssthresh})\)
  - \(cwnd += \text{int}(\text{MSS}/K)\)

- RFC recommends \(max_{ssthresh}=100\) for most of cases. But this still suffers for a large BDP path

It took 10 seconds for probing
if (DUPACKS are received)
    switch to congestion avoidance phase;
else (ACK is received)
    if (ssthresh < (ERE*RTT_min)/seg_size)
        ssthresh=(ERE*RTT_min)/seg_size;/*reset ssthresh*/
    endif
    if (cwnd >= ssthresh) /*linear increase
        phase*/
        increase cwnd by 1/cwnd;
    else if cwnd < ssthresh) /*exponentially
        increase phase*/
        increase cwnd by 1;
    endif
endif

- Problem
  - ERE increases very slowly
What really happens during Slow Start?

Observation 1 - ACK train length

The ACK train length keeps increasing.

Packet losses!

50Mbps, 32ms RTT, and 100% BDP (133 pkts)
Detailed look on the ACK train (1)
Detailed look on ACK train length (2)

We can set ssthresh to the value of cwnd when we expect the cwnd overflood the pipe in the next round.
Hybrid Slow Start – ACK train length

Without knowing \((N-1)L\) and \(b(N)\), we can use \(\Lambda(N)\) to infer whether the packets in flight approaching the BDP of a path.

TCP sender doesn't need a high-resolution clock as the TCP sender needs only a rough estimation of ACK train.
Observation 2 – increase in packet delays

50Mbps, 166ms RTT, and 100% BDP (553pkts)
Testbed (Dummynet) Setup

The bottleneck bandwidth varied from 10Mbps to 400Mbps, RTT from 20ms to 160ms, and buffer sizes from 10% to 200% BDP.

Iperf – long lived flows
Surge – web traffic generator
Hybrid Slow Start with quick ACKs
(Linux 2.4 receivers)

Two TCP-SACK flows with Hybrid Slow Start.
400Mbps, 160ms RTT, and 100% BDP (5333 packets)
Hybrid Slow Start with quick and delayed ACKs (Linux 2.6 receivers)

Two TCP-SACK flows with Hybrid Slow Start.
400Mbps, 160ms RTT, and 100% BDP (5333 packets)
Hybrid Slow Start with delayed ACKs
(Windows and FreeBSD)

Two TCP-SACK flows with Hybrid Slow Start.
400Mbps, 160ms RTT, and 100% BDP (5333 packets)
More results with TCP-SACK

100M-1.0 BDP-250ms RTT - No BK

200M-1.0 BDP-50ms RTT - No BK
Apply Hybrid Slow Start to CUBIC

**Hybrid Slow Start**

- **cwnd (flow1)**
- **cwnd (flow2)**
- **ssthresh (flow1)**
- **ssthresh (flow2)**

**Throughput at Routers**

**Time (second)**

**Segments**

**Mpbs**

**Long-lived Background Flows**
**Mid-sized Background Flows**
**Total Forward Traffic**

**Slow Start**

**Hybrid Slow Start**
Testing under more diverse settings

- **Bandwidth:** 400Mbps
  - **RTT:** 160ms
  - **Buffer size:** 10% - 200% BDP

- **RTT:** 160ms
  - **Buffer size:** 100% BDP,
  - **Bandwidth:** 10Mbps - 400Mbps

- **Bandwidth:** 400Mbps
  - **Buffer size:** 100% BDP
  - **RTT:** 20ms - 160ms
Testing with Linux, FreeBSD, and Windows Receivers

Bandwidth: 400Mbps
RTT: 100ms
buffer size: 100% BDP
Conclusion and Future Work

- Using ACK train and delay information significantly improves the efficiency of Slow Start
- Hybrid Slow Start is a small plugin to an existing Slow Start and can be easily integrated with existing TCP congestion control algorithms
- More testing over real production networks and refinements for handling asymmetric link and congestion on the backward path are our future work
Q & A

More experimental results (including Internet2 results) will be available at http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/SlowStart

Thank you for your participation
Backup Slides
Algorithm 1: Hybrid Slow Start

Initialization:
\[ \text{lowsthresh} \leftarrow 16 \quad \text{nSampling} \leftarrow 8 \]

At the start of each RTT round:
begin
\[
\text{if} \! \text{found and cwnd} \leq \text{sthresh then}
\begin{align*}
&\text{// Save the start of an RTT round} \\
&\text{roundStart} \leftarrow \text{lastJiffies} \\
&\text{lastRTT} \leftarrow \text{curRTT} \\
&\text{curRTT} \leftarrow \infty \\
&\text{samplingCnt} \leftarrow \text{nSampling}
\end{align*}
\]
end

On each ACK:
begin
\[
\text{RTT} \leftarrow \text{usecs_to_jiffies(RTTms)} \\
\text{dMin} \leftarrow \text{min(dMin, RTT)}
\]
\[
\text{if} \! \text{found and cwnd} \leq \text{sthresh then}
\begin{align*}
&\text{// ACK is closely spaced, and the train length reaches to } T_{\text{forward}} \\
&\text{if Jiffies} - \text{lastJiffies} \leq \text{usecs_to_jiffies}(2) \\
&\text{then}
\end{align*}
\]
\[
\begin{align*}
&\text{lastJiffies} \leftarrow \text{Jiffies} \\
&\text{if Jiffies} - \text{roundStart} \geq \text{dMin}/2 \text{ then}
\end{align*}
\[
\begin{align*}
&\text{found} \leftarrow 1 \\
&\text{// Samples the delay}
&\text{if samplingCnt then}
\end{align*}
\]
\[
\begin{align*}
&\text{curRTT} \leftarrow \text{min(curRTT, RTT)} \\
&\text{samplingCnt} \leftarrow \text{samplingCnt} - 1 \\
&\text{et} \leftarrow \text{max}(2, \lfloor \text{lastRTT}/16 \rfloor) \\
&\text{// If the delay increase is over } \eta \\
&\text{if samplingCnt and curRTT} \geq \text{lastRTT} + \eta \text{ then}
\end{align*}
\[
\begin{align*}
&\text{found} \leftarrow 2 \\
&\text{if found and cwnd} \geq \text{lowsthresh then}
\end{align*}
\]
\[
\begin{align*}
&\text{sthresh} \leftarrow \text{cwnd}
\end{align*}
\]
end

Timeout:
begin
\[
\text{dMin} \leftarrow \infty \quad \text{found} \leftarrow 0
\]
end
We tested Hybrid Slow Start over the Internet2 path between NCSU (Linux 2.6.25-rc3) and NICT Japan (Linux 2.6.19) and found that the results are very promising.

Packet losses!