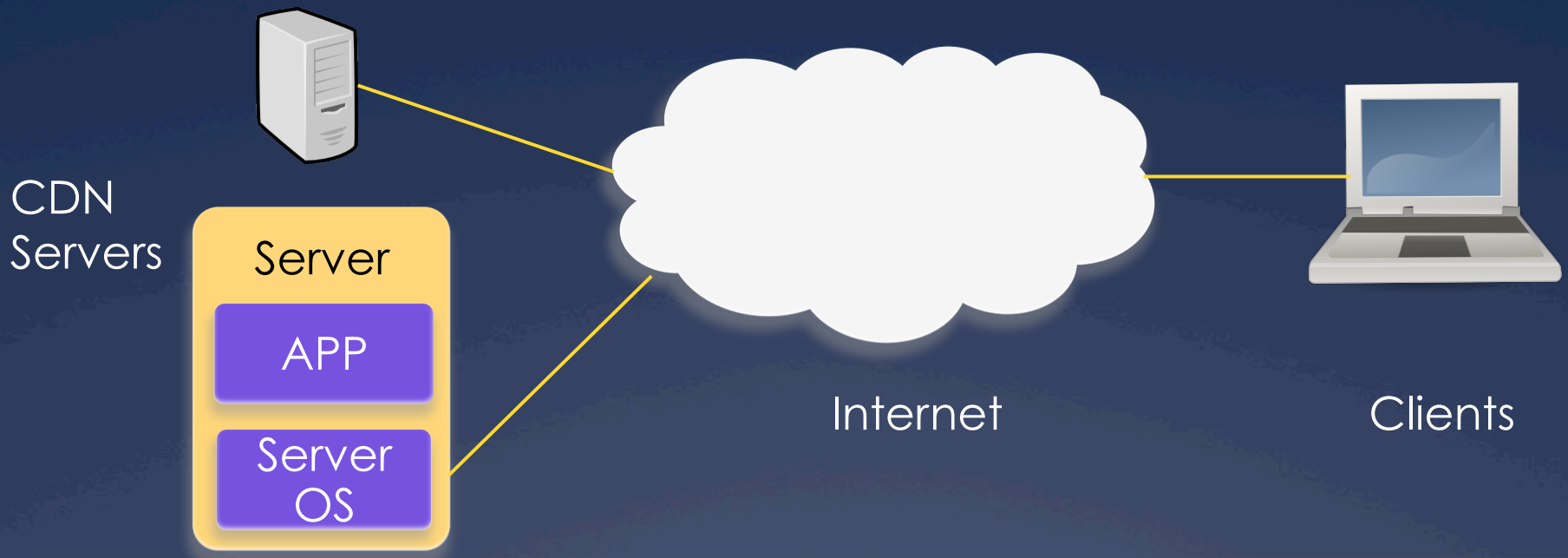
Identifying Performance Bottlenecks in CDNs through TCP-Level Monitoring

Peng Sun
Minlan Yu, Michael J. Freedman, Jennifer Rexford
Princeton University

August 19, 2011

Performance Bottlenecks



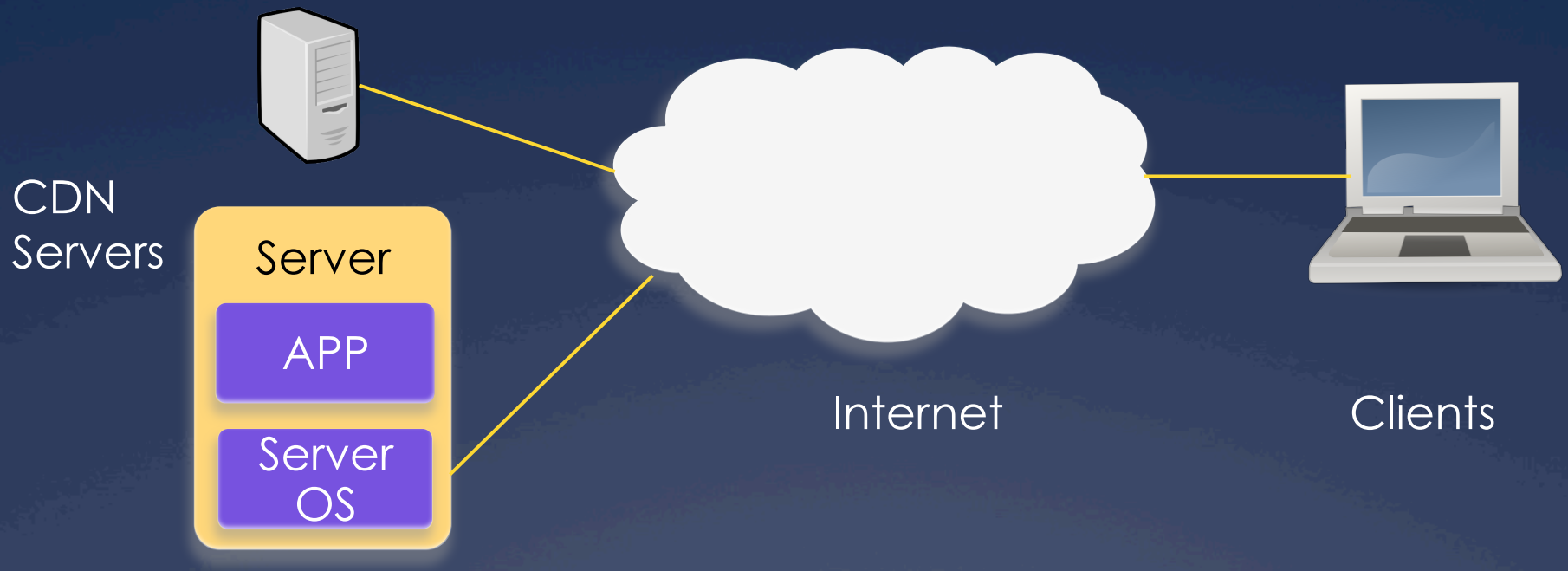
APP
Write too slowly

Internet
Network congestion

Client
Insufficient receive buffer

Server OS
Insufficient send buffer or Small initial congestion window

Reaction to Each Bottleneck



APP is bottleneck: Debug application

Server OS is bottleneck: Tune buffer size, or upgrade server

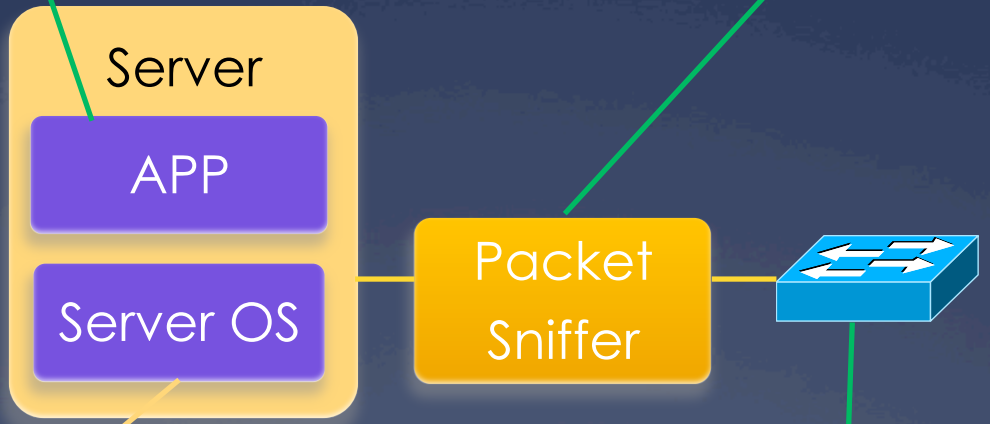
Internet is bottleneck: Circumvent the congested part of network

Client is bottleneck: Notify client to change

Previous Techniques Not Enough

Application logs:
No details of network activities

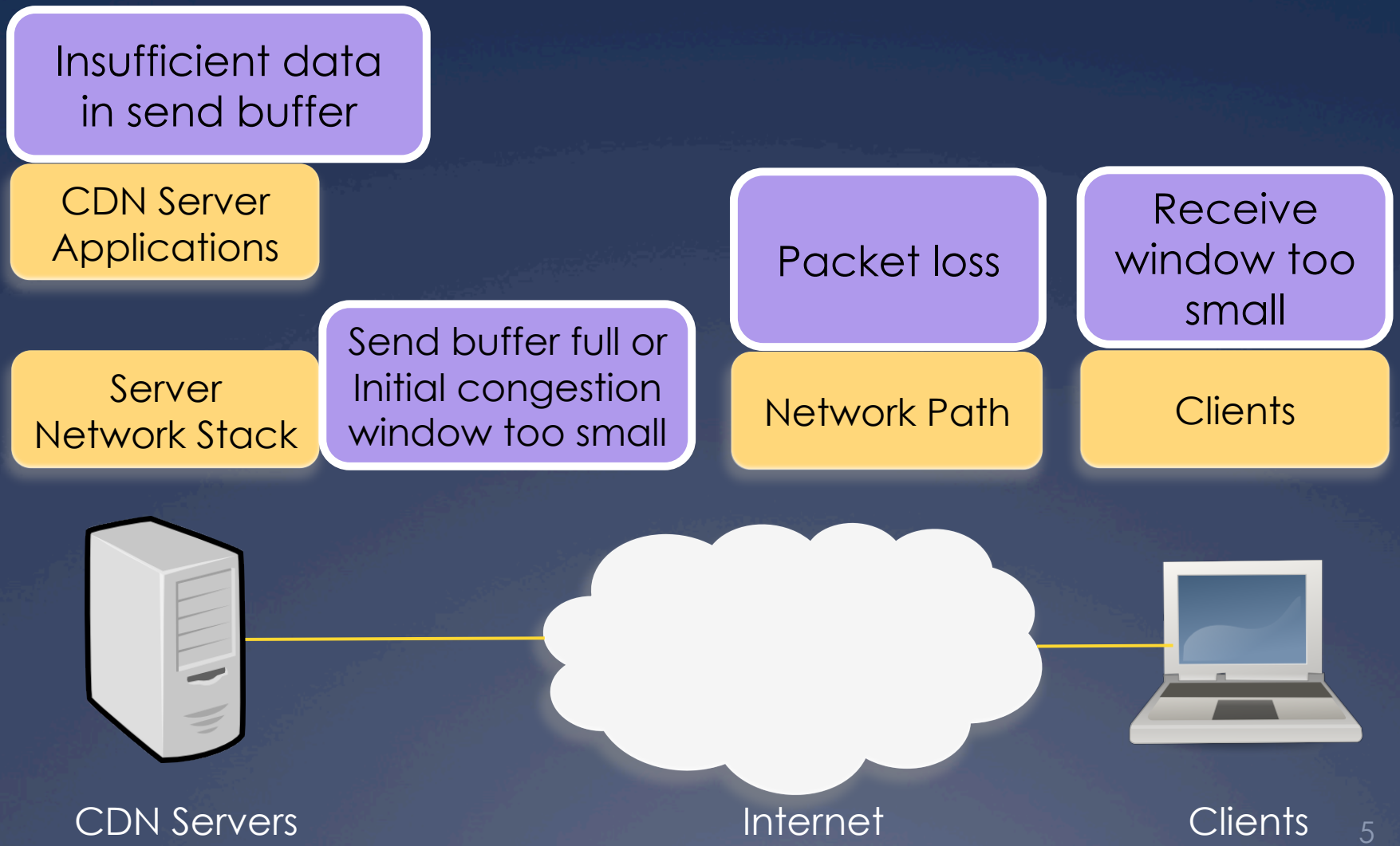
Packet sniffing:
Expensive to capture



Transport-layer stats:
Directly reveal perf. bottlenecks

Active probing:
Extra load on network

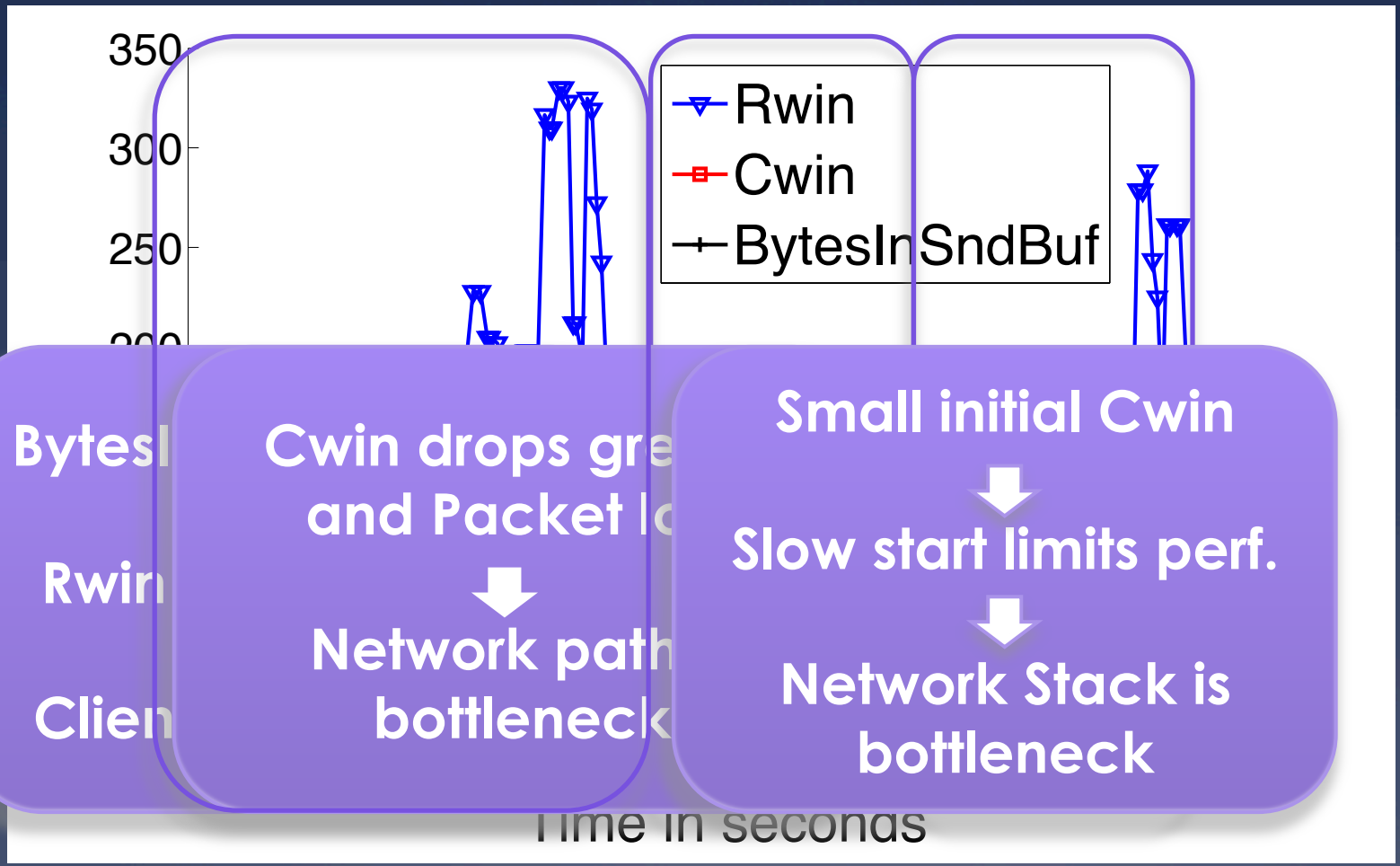
How TCP Stats Reveal Bottlenecks



Measurement Framework

- Collect TCP statistics
  - Web100 kernel patch
  - Extract useful TCP stats for analyzing perf.
- Analysis tool
  - Bottleneck classifier for individual connections
  - Cross-connection correlation at AS level
    - Map conn. to AS based on RouteView
    - Correlate bottlenecks to drive CDN decisions

How Bottleneck Classifier Works



CoralCDN Experiment

- CoralCDN serves 1 million clients per day
- Experiment Environment
  - Deployment: A Clemson PlanetLab node
  - Polling interval: 50 ms
  - Traces to Show: Feb 19th – 25th 2011
  - Total # of Conn.: 209K
  - After removing Cache-Miss Conn.: 137K (Total 2008 ASes)
- Log Space overhead
  - < 200MB per Coral server per day

What are Major Bottleneck for Individual Clients?

- We calculate the ***fraction of time*** that the connection is under each bottleneck in lifetime

Bottlenecks	% of Conn. With Bottleneck for >40% of Lifetime
Server Application	10.75%
Server Network Stack	18.72%
Network Path	3.94%
Clients	1.27%

Our suggestion:
Filter them out of decision making

AS-Level Correlation

- CDNs make decision at the AS level
  - e.g., change server selection for 1.1.1.0/24
- Explore at the AS level:
  - Filter out non-network bottlenecks
  - Whether network problems exist
  - Whether the problem is consistent

Filtering Out Non-Network Bottlenecks

- CDNs change server selection if clients have low throughput
- Non-network factors can limit throughput
- 236 out of 505 low-throughput ASes limited by non-network bottlenecks
- Filtering is helpful:
  - Don't worry about things CDNs cannot control
  - Produce more accurate estimates of perf.

Network Problem at AS Level

- CDN make decision at AS level
- Whether conn. in the same AS have common network problem
- For 7.1% of the ASes, half of conn. have >10% packet loss rate
- Network problems are significant at the AS level

Consistent Packet Loss of AS

- CDNs care about predictive value of measurement
- Analyze the variance of average packet loss rates
  - Each epoch (1 min) has nonzero average loss rate
  - Loss rate is consistent across epochs (standard deviation < mean)

Analysis Length	# of ASes with Consistent Packet Loss
One Week	377 / 2008
One Day (Feb 21st)	122 / 739
One Hour (Feb 21st 18:00~19:00)	19 / 121

Conclusion & Future Work

- Use TCP-level stats to detect performance bottlenecks
- Identify major bottlenecks for a production CDN
- Discuss how to improve CDN's operation with our tool
- Future Works
  - Automatic and real-time analysis combined into CDN operation
  - Detect the problematic AS on the path
  - Combine TCP-level stats with application logs to debug online services

Thanks!

Questions?