Dependent Link Padding Algorithms for Low Latency Anonymity Systems

Wei Wang, Mehul Motani and Vikram srinivasan
Department of Electrical & Computer Engineering
National University of Singapore, Singapore

CCS '08: *Proceedings of the 15th ACM conference on Computer and communications security*

Motivation

- To protect anonymous communication
  - allow users to communicate with each other while not revealing the identity of the sender and the receiver to a third party.
- Low latency anonymity systems are vulnerable to traffic analysis attacks
  - Discern user identity by using message content, length and correlations of packet transmission times, etc.

- This paper proposes a scheme to provably defeat all packet matching attacks against low-latency anonymity systems while providing a strict delay bound, by introducing the minimal amount of cover traffic.

Background

- One way to thwart such an attack is to use dummy traffic
  - Manipulating the sending time of outgoing packets
  - A dummy packet will be sent by the server when there is no packet to send at the scheduled time
- Effectiveness of dummy traffic
  - Link ability from a suspect input to any suspect output to be:
    - Minimized?
    - Randomized?
    - Equalized?
- Cost: genuine traffic vs. dummy traffic

Independent Link Padding Schemes

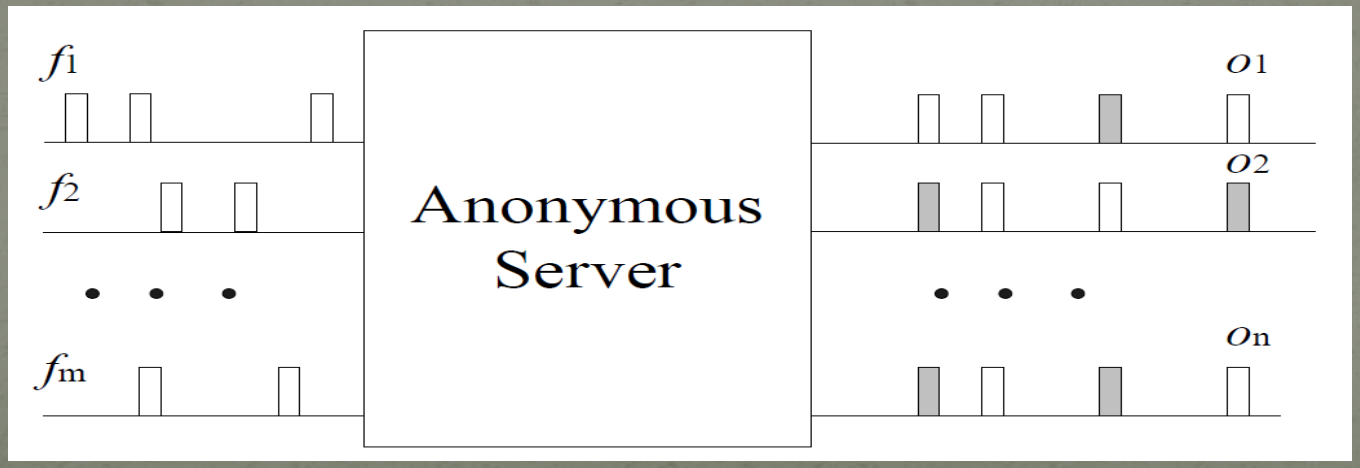
- Output pattern: pre-determined regardless of input
- Straightforward output patterns:
  - Constant - send packet at a constant rate
  - Exponential- sends packets according to a randomly generated schedule e.g. Poisson process
- Two limitations
  - The independently generated schedule imposes additional limitations on the sending rate of servers in the anonymity
    - Long delay, drop packet
  - Need to know the average sending rate of the incoming flow to work efficiently
    - Padding rate depends on incoming rate
    - Padding rate too high: most bandwidth are wasted on dummy packets
    - Padding rate too low: incoming packets are dropped

Dependent Link Padding Schemes

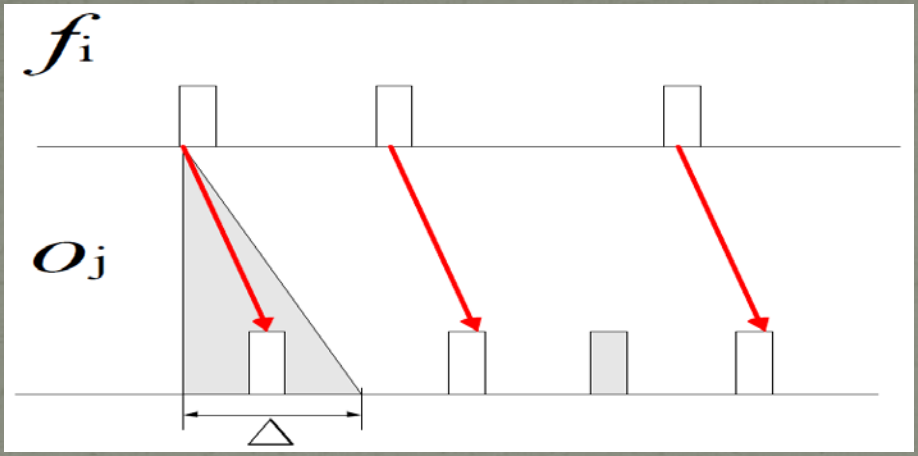
- Output pattern: determined online depending on input
- Advantages:
  - Provide strict delay bounds with no packet drops
  - Dynamically adapt to the traffic rate changes in incoming flows
  - Use minimum transmission rate to provide maximum anonymity when packets are not dropped
- How to produce output with given input?

Assumptions

- Input flows are about of the same rate in Poisson
- All packets belong to a flow (link) are sent to the same output flow (link)
- Single anonymity server (mix) with a strict delay bound
- The mix does not drop any packet
- All output links show the same output to maximize the anonymity



Matching Attacks



- An outgoing flow o_j is a matched schedule for f_i if and only if

$$N_{f_i}(t') - N_{f_i}(t) \leq N_{o_j}(t'+\Delta) - N_{o_j}(t)$$

for any time interval of [t, t'], t'≥t, where N_{f_i} is the total number of packets arrived at flow f_i until time t.

Proposed DLP Algorithm

- Notations:
  - $F = (f_1, f_2, \ldots, f_m)$ a set of incoming flows
  - $S(F)$ is a sequence of sending times t_s, $s = 1, 2, \ldots$ at which the server sends either one message packet or one dummy packet on the outgoing link.
    - Each sending opportunity in $S(F)$ is called a token.
  - P_{ij} is the *j*th packet in flow f_i
  - t_{ij} is the arrival time for P_{ij}
  - $R_{S(F)} = \lim_{s \to \infty} \frac{s}{t_s}$ is sending rate
- Input: the packet timing of flows in F
- Output: a schedule $S(F)$ [a matched schedule for all flows in F].

Proposed DLP Algorithm

Dependent Link Padding Algorithm

Parameters: Packet arrival time t_{ij} for all flows $f_i \in \mathcal{F}$

Output: A matched schedule $S(\mathcal{F})$ for all flows $f_i \in \mathcal{F}$

01: Take a new packet P_{ij} according to the arrival sequence.

02: **if** there is an unused token with $t_s \geq t_{ij}$ for f_i

03: Schedule P_{ij} at t_s

04: Mark the token as used for f_i

05: **else**

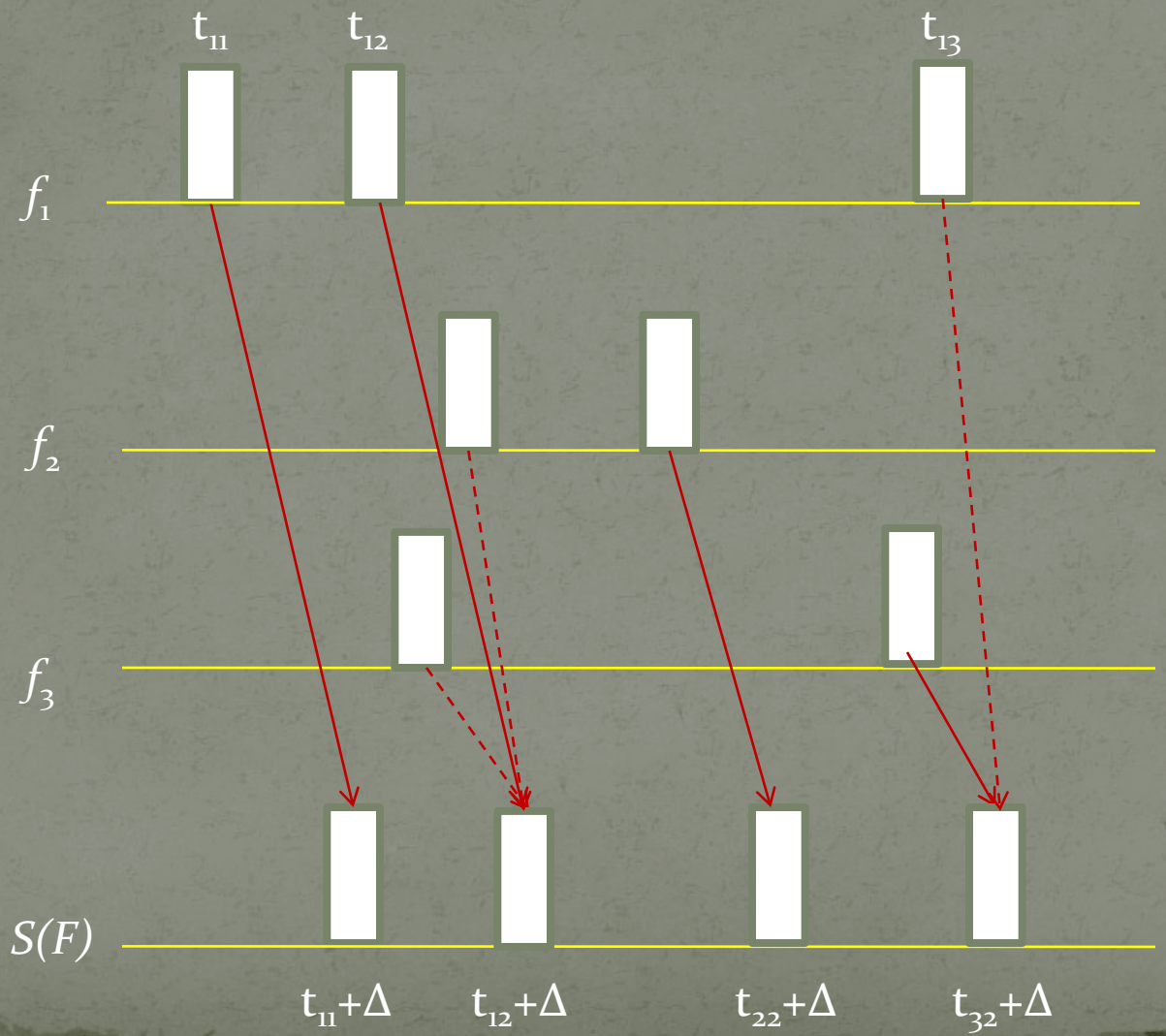
06: Add a new token at $t'_s = t_{ij} + \Delta$ in $S(\mathcal{F})$, which can be used by all flows in $\mathcal{F}$

07: Schedule P_{ij} at time t'_s and mark the token as used for f_i.

08: **endif**

09: Go to step 01 until no more packet arrives.

Using DLP to generate S(F)



Claims

- The server will send no packets when there are no incoming flows
- All n output links start and ending transmitting at the same time
- Avoid matching attack or watermarking attack
- The dummy traffic is minimized (max efficiency)
- Sending rate proportional to log(m)

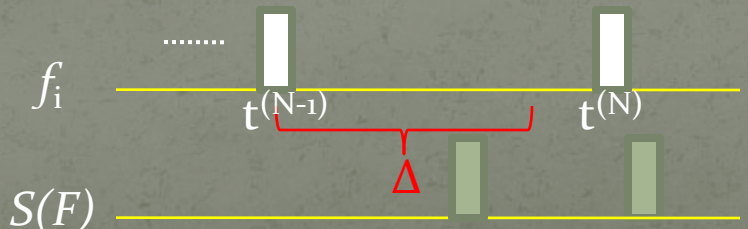
Optimality of DLP

- *THEOREM 1: the schedule S(F) generated by DLP uses the minimum number of tokens among all matched schedules of the incoming flow set of F.*

- DLP algorithm is optimal at time $t^{(1)}$ when $P^{(1)}$ arrives.
  - The packets in the union of all incoming flows according to their arrival time are denoted as $P^{(1)}$, $P^{(2)}$, $P^{(3)}$ …
  - E.g. $P^{(1)}$ = P11, $P^{(2)}$ = P12, $P^{(3)}$ = P31

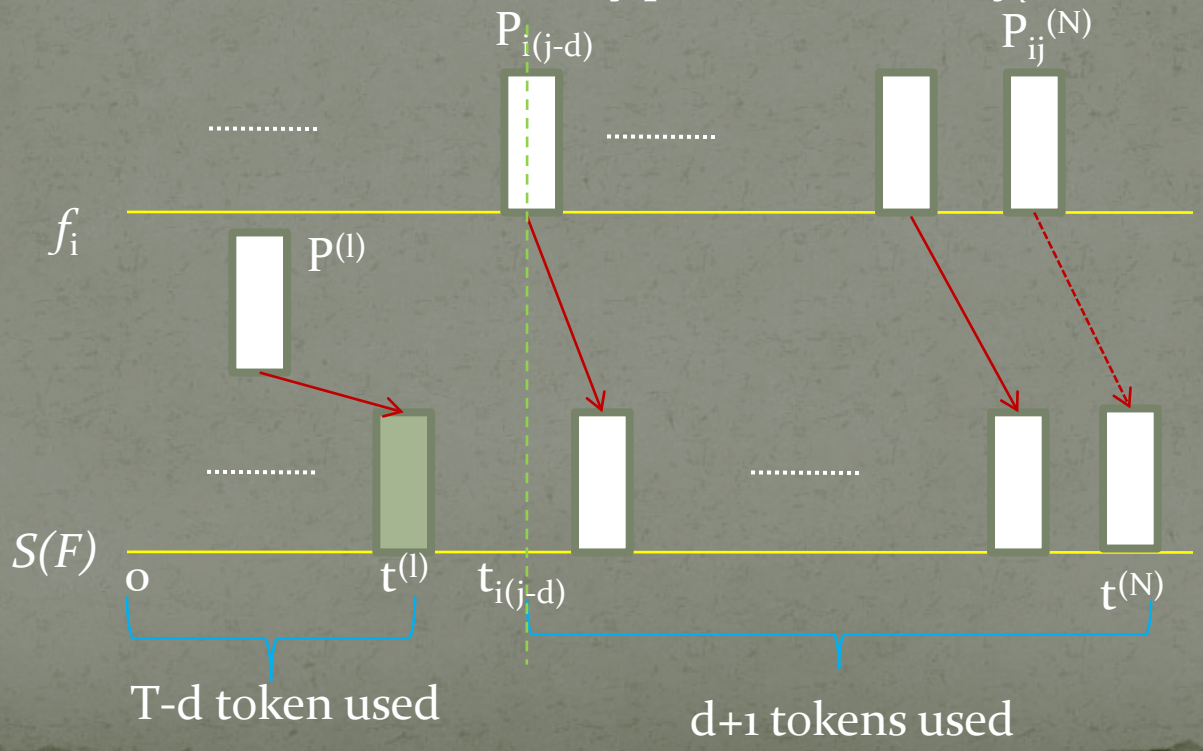
Proof

- Suppose DLP is optimal at time $t^{(k)}$ for all $k \leq N\text{-}1$ and the algorithm uses T tokens at time $t^{(N-1)}$.
- When the packet $P^{(N)}$ arrives, there are three different cases:
  - Case 1: DLP finds an unused token for packet $P^{(N)}$
    - The number of tokens used is not increased.
  - Case 2: DLP cannot find an unused token for packet $P^{(N)}$ and $t^{(N)} > t^{(N-1)} + \Delta$
    - Uses T+1 tokens at time $t^{(N)}$

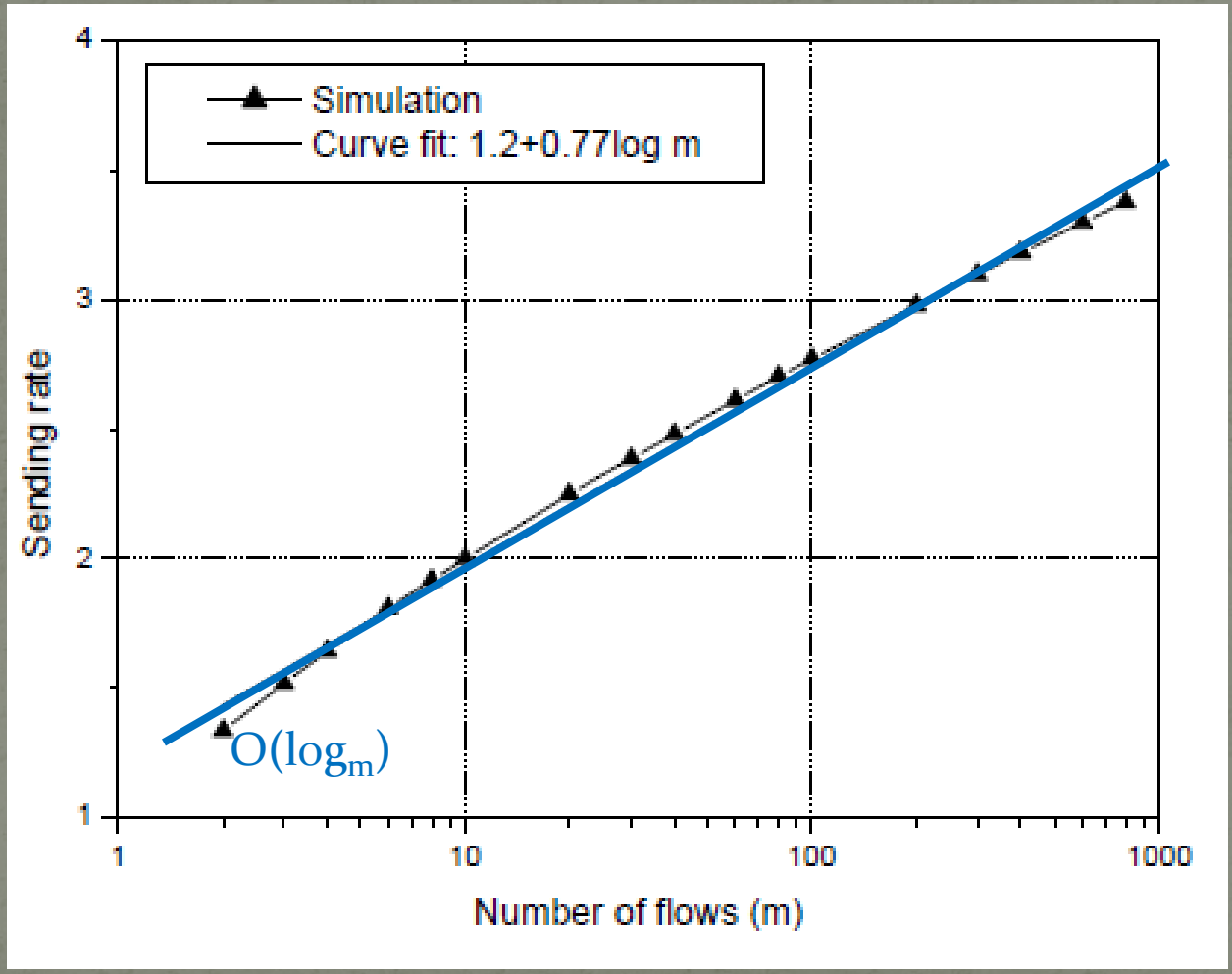


Proof

- Case 3: DLP cannot find an unused token for packet $P^{(N)}$ and $t^{(N)} \leq t^{(N-1)} + \Delta$
  - If DLP sends no dummy packets for flow f_i
  - If DLP sends at least one dummy packets for flow f_i

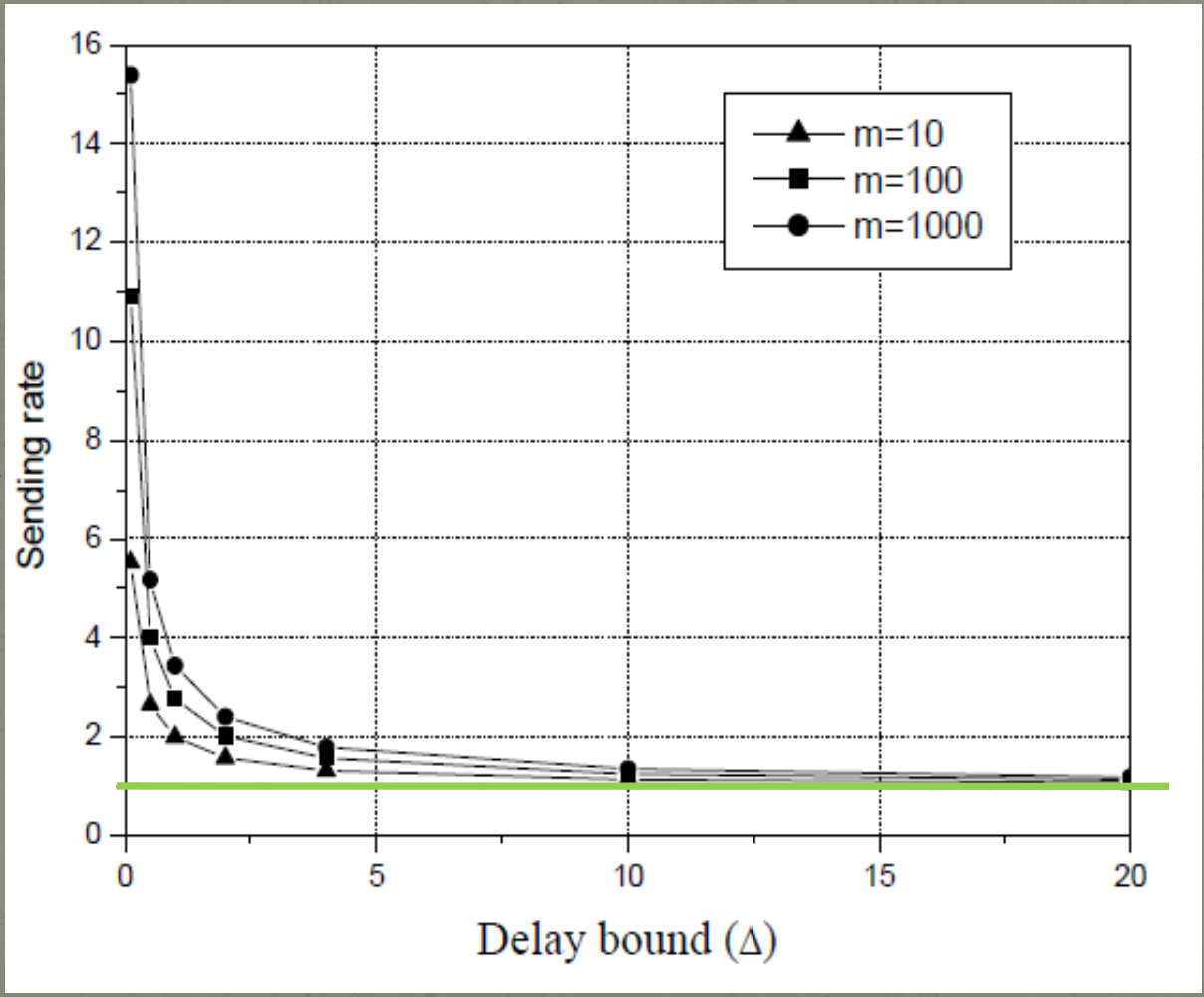


Experiment on the sending rate



Experiment on delay bound

$$R_{S(F)} = \lim_{s \to \infty} \frac{s}{t_s}$$



Comparison with ILP Algorithms

- Assume different outgoing link in ILP schemes have the same sending rate.
- For ILP schemes, the probability that some packet cannot meet the delay bound is non-zero when the incoming flow is a Poisson process.
  - May drop some packets
- Unfair to directly compare the DLP algorithm with ILP algorithm
  - Provide a DLP heuristic algorithm which drops some token to reduce the sending rate of the generated schedule.

DLP Heuristic Algorithm

- Notations:
  - d is the number of message packets sent by the token
  - |F| is the size of the incoming flow set of DLP algorithm
  - u=d/|F| is token utility
  - $1/|F| \leq u \leq 1$
    - Each token scheduled by DLP should at least send one message packet and it can at most send one packet for each flow
- A token is used only if its utility is larger than a given threshold U.
- If a token is not used due to low utility, all packets scheduled at this token will be rescheduled or dropped if its delay bound cannot be met.

DLP Heuristic Algorithm

DLP Heuristic Algorithm

Parameters: Packet arrival time t_{ij} for all flows $f_i \in \mathcal{F}$
Utility threshold U.

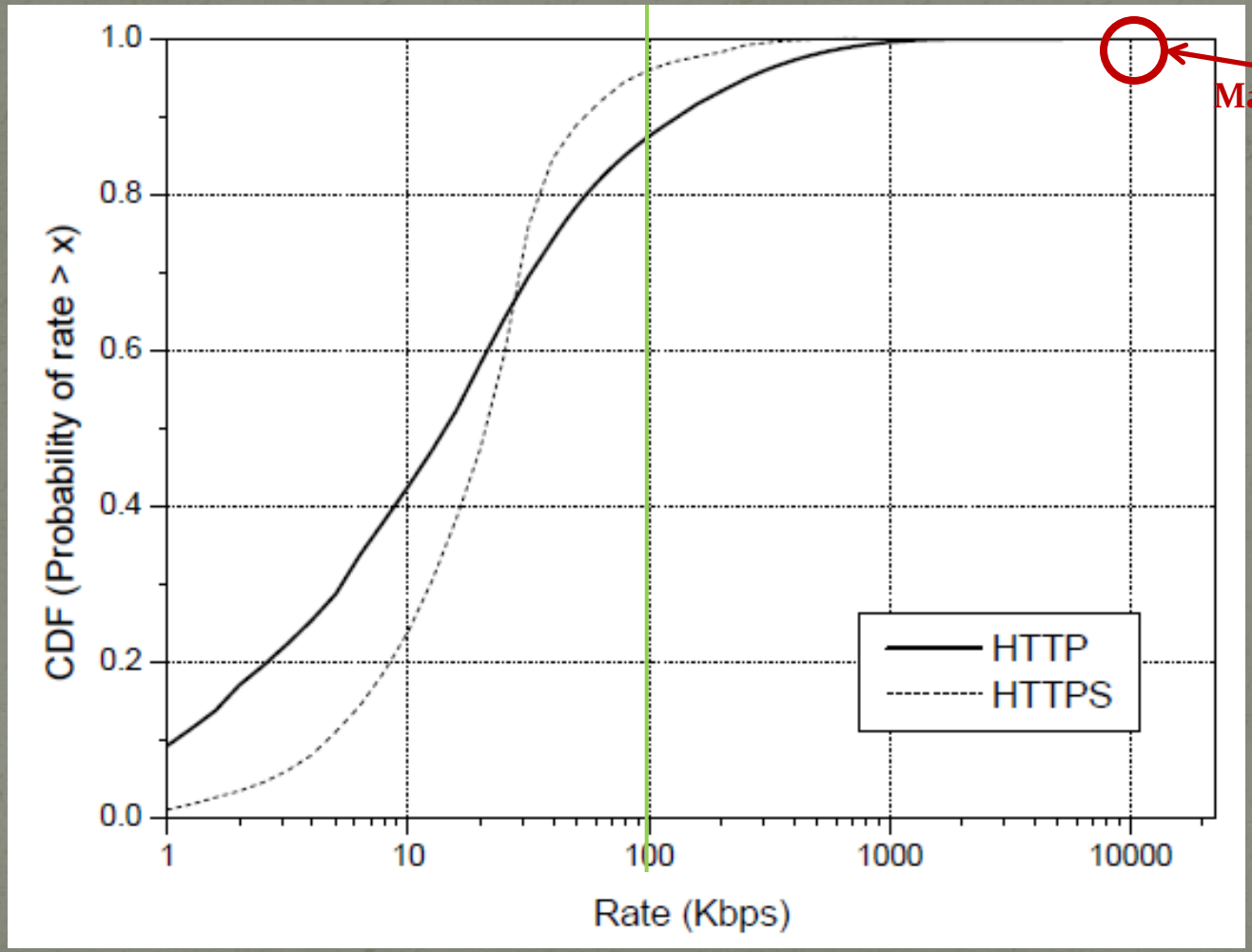
Output: A sending schedule with utility of at least U

01: Put new packet P_{ij} into a FIFO queue for the flow f_i
02: Repeat step 01 until there is a packet P has been in the queue for Δ time units
03: **if** more than $U|\mathcal{F}|$ queues are non-empty
04: Add a new token and send one packet for each flow immediately
05: **else**
06: Drop the packet P.
07: **endif**
08: Go to step 01 until no more packet arrives.

Simulation on Network Trace

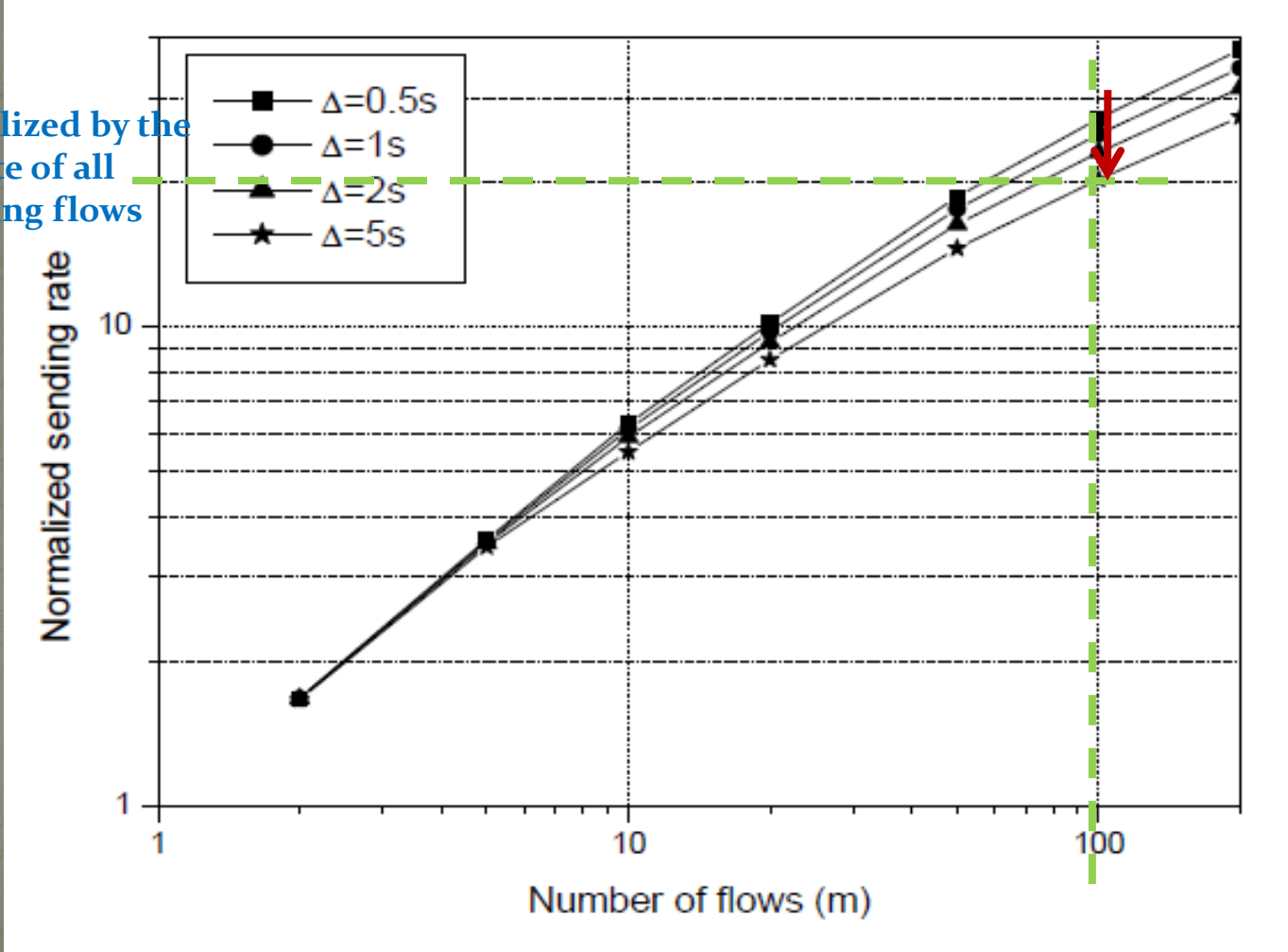
- Use the trace file from NLANR Auckland-VIII data set
- Randomly pickup *m* flows from the trace as the incoming flows
- Pick up a new flow when a old flow terminates so that the number of incoming flows stays at *m*.
- Segment all packets to a normalized length of 512 bytes as in many anonymity systems

Rate distribution of http and https flows

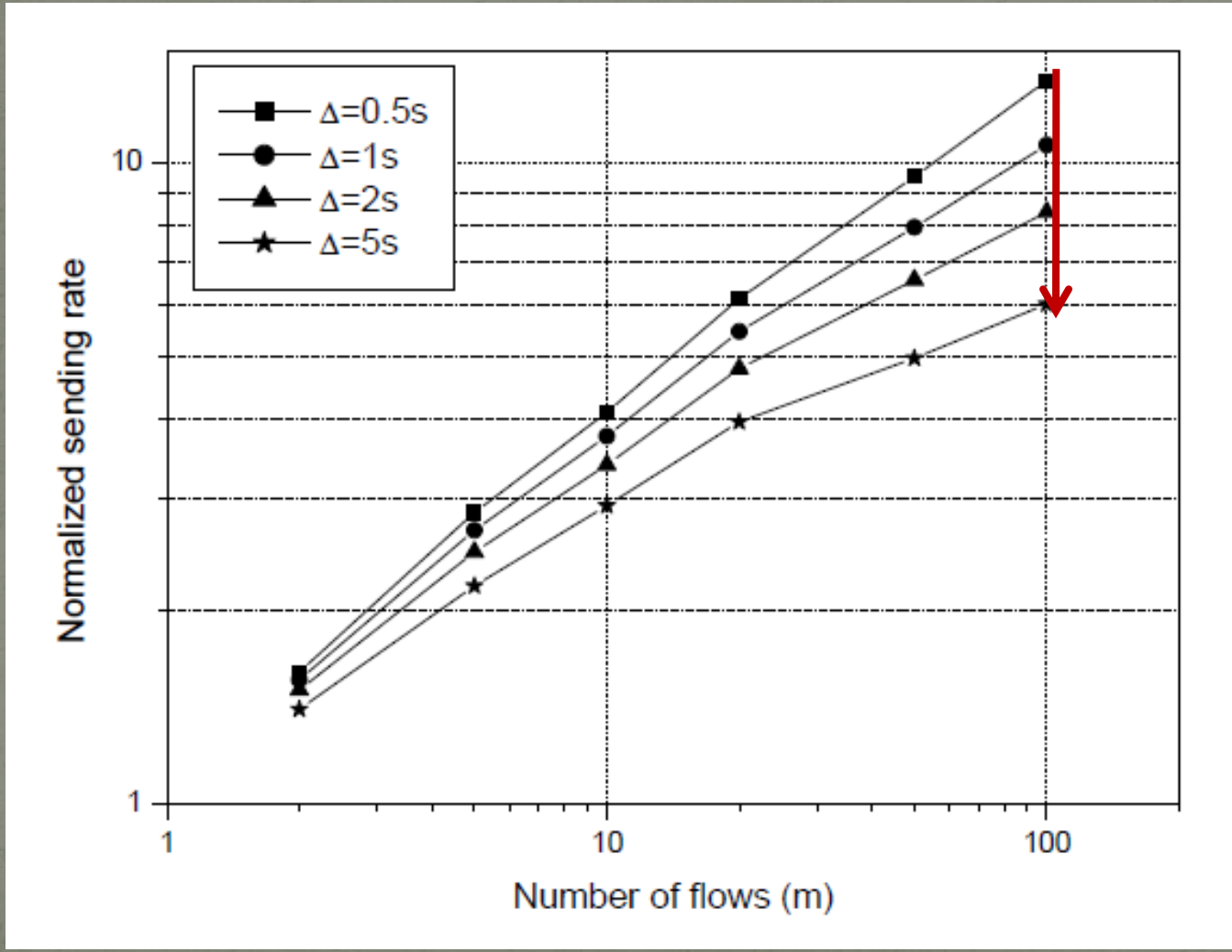


DLP sending rate for http flows under different delay constrains

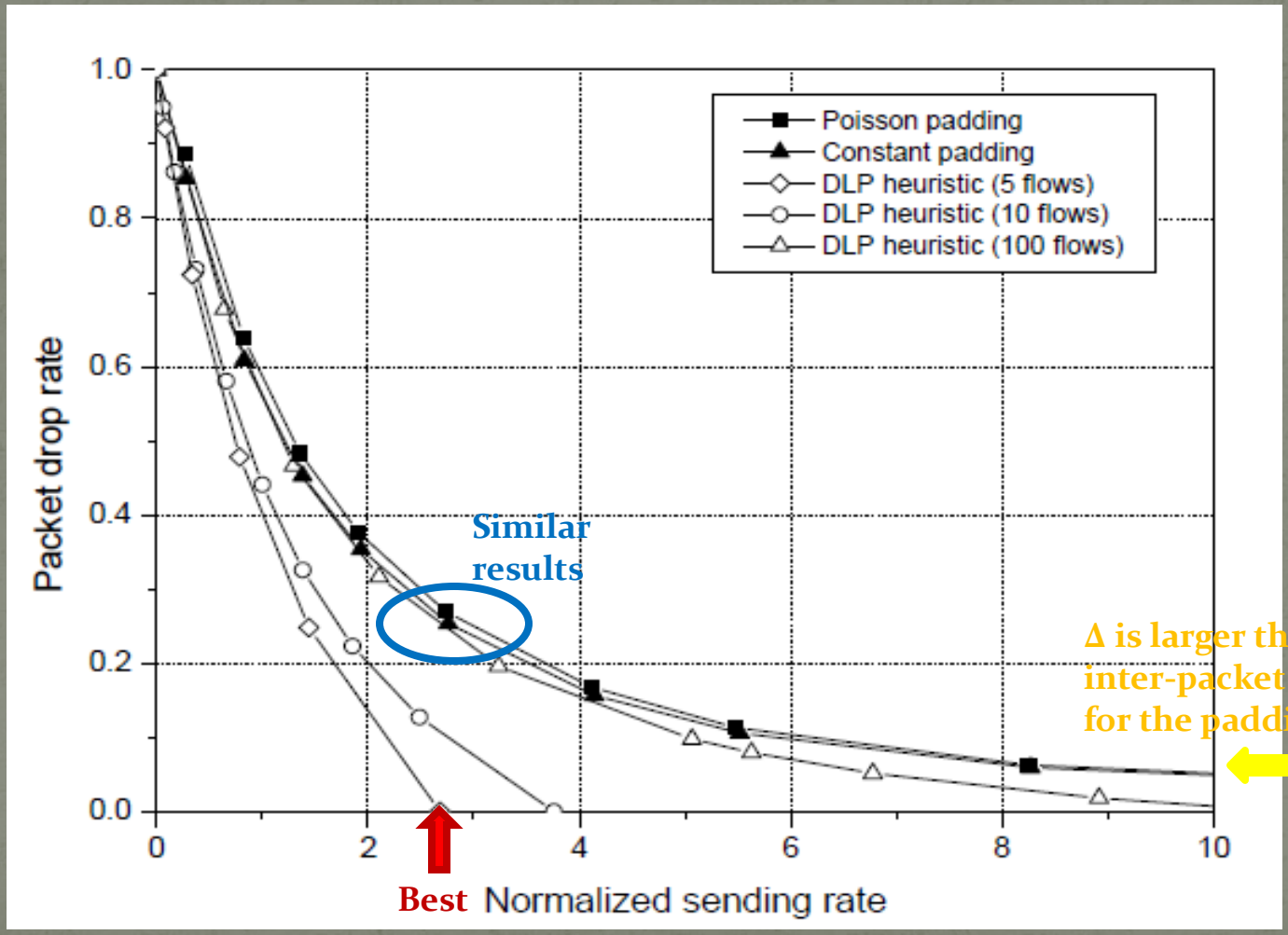
Normalized by the avg. rate of all incoming flows



DLP sending rate for https flows under different delay constrains



Packet Drop Rates (https traffic)



Conclusion

- DLP uses minimum rate to provide full anonymity when there are strict delay constraints on the anonymity systems

- Drawback: the rate of the optimal padding algorithm increases very fast when the number of user flow increases.