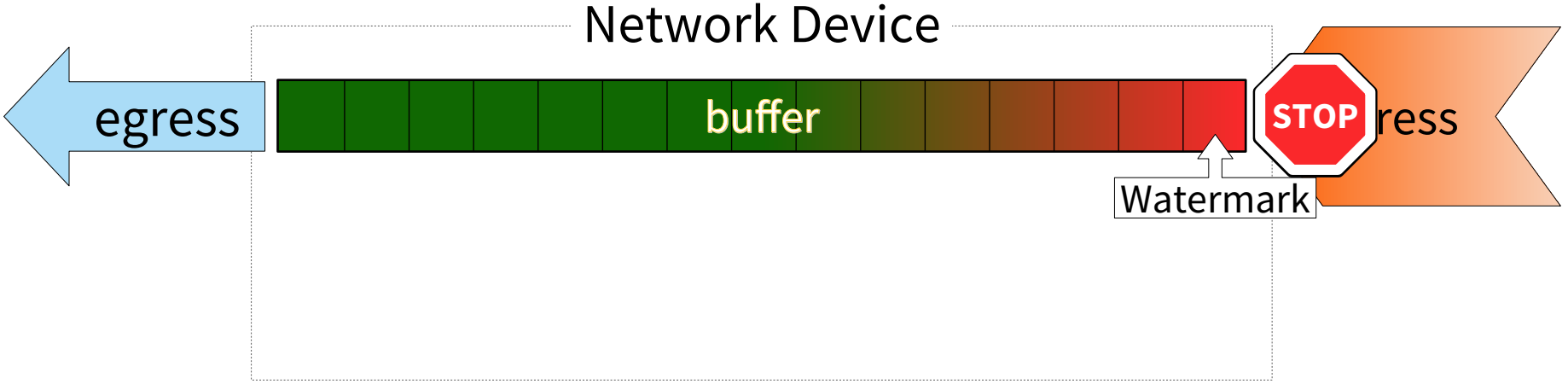
Random Early Drop

- **Instruments**
  Impeding
  Impertinent
  Invasions

- **GUUG-Sommerworkshop 2026**
- Thomas Maus

Origins of RED

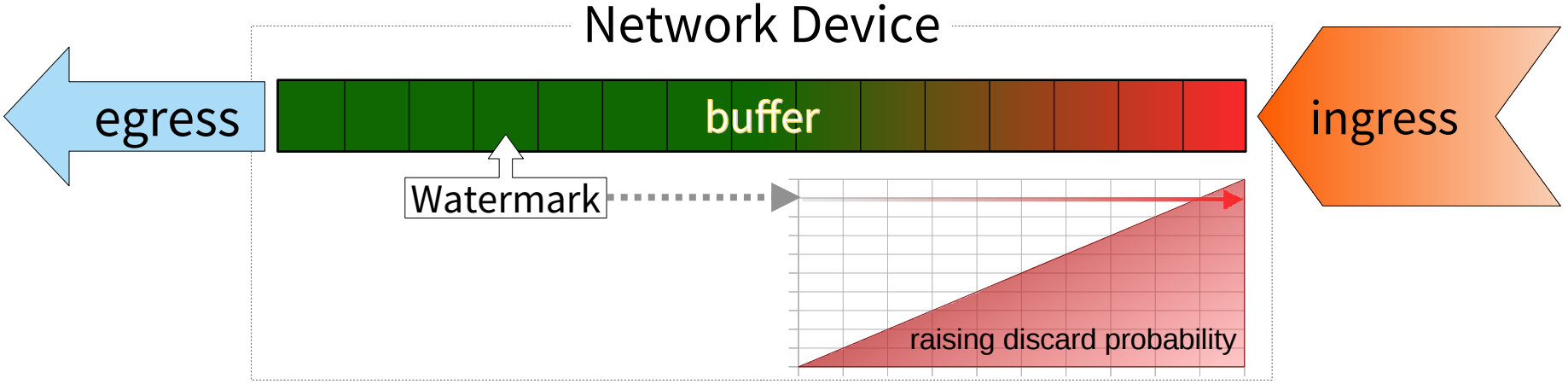
- RED = Random Early Drop (or Discard/Discovery)
- by Sally Floyd and Van Jacobson (early 90ies)
- congestion avoidance in networks
- (more) fair queuing discipline for network scheduling



- naive approach: buffer full → tail drop

Origins of RED

- RED = Random Early Drop (or Discard/Discovery)
- by Sally Floyd and Van Jacobson (early 90ies)
- congestion avoidance in networks
- (more) fair queuing discipline for network scheduling



- RED: discard early with metrics-controlled probability
- modern approach: *core idea still used*,
  better tailored to ECN, TCP window control, flows, …

Application to IT Security Domain

- benefits
  - diminish impact of DOS
  - retain some service level for legitimate users
  - impede login brute-forcing
- aptitude
  - even better suited than in congestion control
  - no adaption to specific protocol behavior need
  - enforcement of policy limits
- challenges
  - implementing suitable metrics
  - efficient algorithms & data structures

An Example Scenario: Geographical Infomation System

- context
  - regional planning GIS
  - pure inquiry system – no sensitive data → no 2FA
  - upkeep service for government + paying users (raison d'être)
  - provide best effort service for general public
- user population
  - anonymous users – limited to simple, lightweight functions
  - registered users – access to more + heavier functions
  - paying registered users – access even to massive analytics
  - governmental users – full access …
- usage structure
  - mostly users from nearby
  - occasional users from far away

Stress Scenario #1
Thwarting Login Brute-Forcing

- requirements
  - # login attempts per account and hour tolerated
  - no lockout! (provides DOS opportunities)
- implementation of login rate metrics?
- some guidance
  - no persistence needed
  - constant + small footprint – we aim at O(1)
    - in space (memory)
    - and time (processing)

Thwarting Login Brute-Forcing Implementation of Metrics

- sliding exponential average
  - on the arrival rate of login attempts
  - = reciprocal of time intervals
- well founded in queuing theory
  - non-stationary, in-homogeneous Poisson processes
  - sliding exponential average an efficient estimator
    - constant and small computational and memory needs
    - intuitive parametrisation
  - ➲ reliable, predictable, verifiable behavior!

Thwarting Login Brute-Forcing Implementation of Metrics

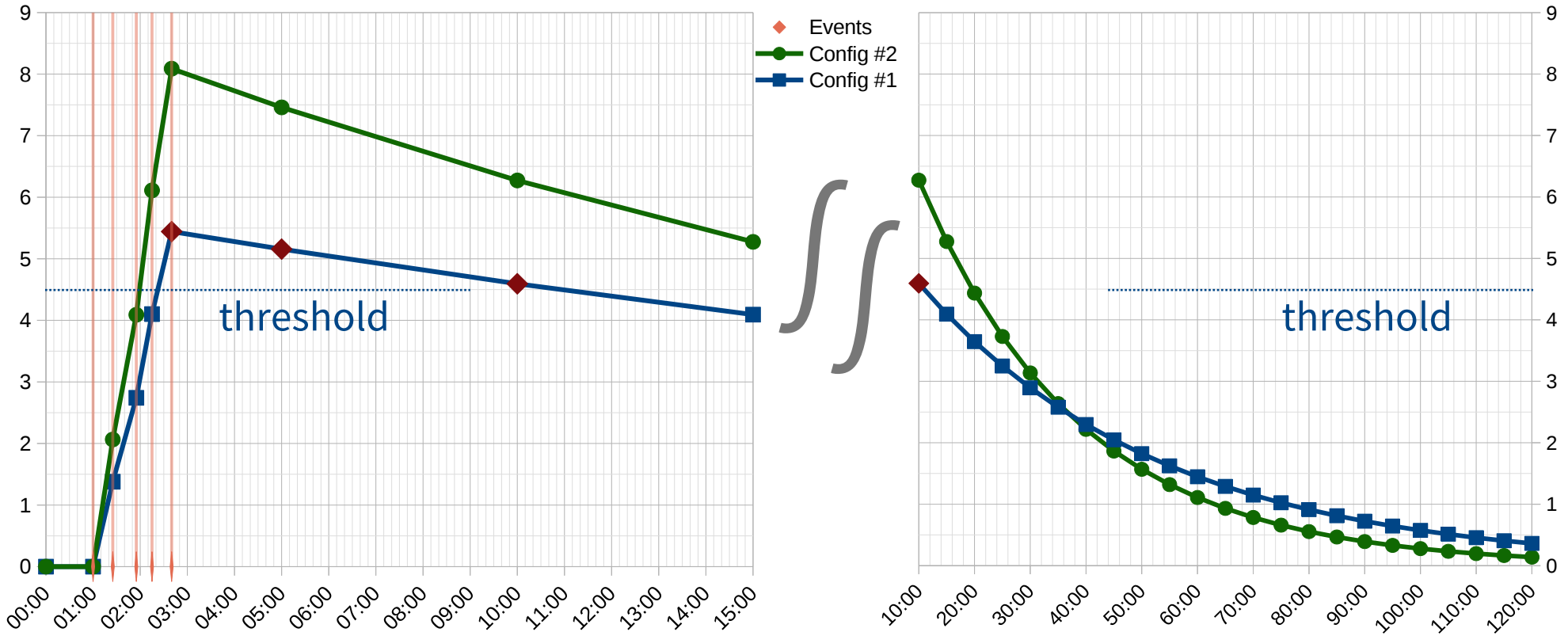
- per RED class = global parameters for all accounts
  - tolerated login attempts (events) per interval (e.g. 5/h)
  - half-live period of memory (e.g. 20 minutes) – forgetfulness ;-)
- per RED object = state per each account
  - timestamp and watermark of last event
  - 16 bytes – given 64bit-architectures (6 bytes would be viable!)
- per RED relevant event = per login trial
  - testing RED activation = 1 comparison
  - adjusting RED object state
    - 7 arithmetic operations (division, multiplication, sum)
    - 1 logical shift operation

Thwarting Login Brute-Forcing Behavior of Implementation

- configuration #1
  - **5** login attempts per 1 hour tolerated
  - **30** minutes half-live period of memory
  - RED activation threshold: **4.50** (the E in RED)
  - max. daily attempts without RED activation: **108**
- configuration #2
  - **10** login attempts per 1 hour
  - **20** minutes half-live
  - RED activation threshold: **9.63**
  - max. daily attempts: **231**

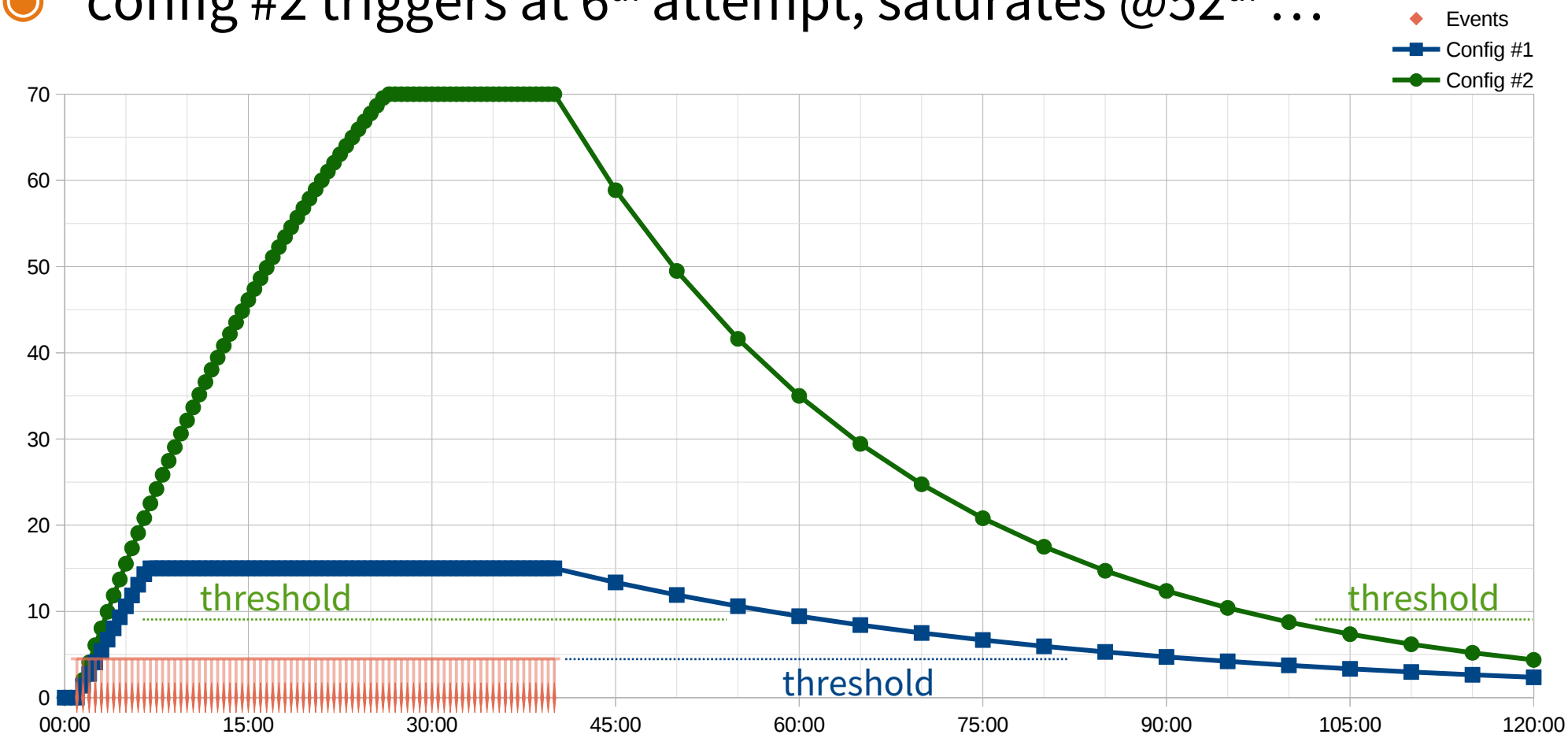
Thwarting Login Brute-Forcing Behavior of Implementation

- well relaxed user after holiday – password blackout …
  - 5 attempts within 2 minutes
  - config #1 just triggers …



Thwarting Login Brute-Forcing Behavior of Implementation

- brute-force attack – 2 attempts per minute
  - config #1 triggers at 5th attempt, saturates @13th …
  - config #2 triggers at 6th attempt, saturates @52th …



Thwarting Login Brute-Forcing Useful Actions

- boring
  - logging + SIEMing RED activation + saturation
- **fun** – preparing headaches for the attackers
  - delayed responses (seconds)
    - a pinch of tar-pitting
    - avoiding tale-telling timings!
  - randomly discard login attempts for the account
    - RED activation+consequences are announced in the login mask: ”Your account is under attack. To protect you …”
    - randomly chosen logins attempts will fail despite correct password!
    - we ramp up the discard rate
      - ideally based on password quality …
    - saturation: 1 out of N attempts is accepted (e.g. 1 out of 4)
    - legitimate users can still login with 4 or 5 attempts
  - activate CAPTCHAs …

Thwarting Login Brute-Forcing Headaches for the Attackers?

- wimps will walk away …
- the others are confronted with difficult choices
  - attackers don't know (if tale-telling timings were avoided)
    - if own or concurrent activity triggered RED
    - what the actual rate of random rejection is
  - attackers understanding + doing the math find
    - random rejection = false negatives = lost "valid logins"
    - limiting loss requires repeated attempts! → attack cost increased …
    - @ 75% rejection: ≤ 20% loss → factor **6**, ≤ 10% loss → factor **9**
    - @ 80% rejection: ≤ 20% loss → factor **8**, ≤ 10% loss → factor **11**
  - if CAPTCHAs are added to login mask
    - given indiscernibility of discard, wrong password, or capture failure
    - same math applies → attack cost further increased …
    - e.g. 75% rejection + 20% CAPTCHA failure → 80% FNR!
  - flying below the radar is not attractive either …

Thwarting Login Brute-Forcing More Headaches for the Attackers …

- we cascade several RED filters …
- prepend a geolocation-based filter
  - restrict far-away sources stronger than regional ones
  - geolocation inherently imprecise → permanent coarse filter
- put an IP-range-based filter in front
  - fine filter – as needed …
  - easy automatic house-keeping …
- interaction of cascaded filters
  - if request is discarded by a filter stage
    - later filters don't see and act upon it
    - their activation levels cool down …
  - otherwise: various useful request designations configurable
    - decision is final – allowing IP-range filters to exempt customer nets
    - filter decisions are stacked – only useful mode e.g. for geolocation

Experience
Less Headaches for the Defenders …

- benefits for defenders
  - system can fend quite well for itself
  - attack effectiveness choked off
  - choice of quick options to improve response
  - rear cover for the important work
    - analysis of attack characteristics
    - ingress blocking of characteristic malevolent traffic
  - user communication + perception
    - “security already takes care” …

Thwarting Login Brute-Forcing Variations

- very large user-base?
- account (non)-existence required indistinguishable?
  (remember talk “Tale Telling Timings”?)
- use hash-bins as RED objects
  - cuckoo or counting bloom filters
  - some adaption of class-level parameters
  - some house-keeping (forgetting) needed
  - various house-keeping strategies available
- hash-bins could be applied to IP addresses too

Stress Scenario #2
Application Overloading

- application level DOS
  - inconsiderate or malevolent users
  - frequently triggering resource intensive operations
  - real life incident: after bidding war …
    - bid winner's application subtly DOSed on critical date
    - only legitimate requests by legitimate users
    - concentrated on a single very inefficient function (massive development + testing blunder!)
    - for regulatory reasons no lockout possible!
    - long-lasting reputation damage for the bid winner!

Stress Scenario #2
Application Overloading

- application level solution
  - account, geolocation + IP-range filters as before
  - SessionID-based filters for anonymous access
  - global performance threshold controls if rejections happen
  - individual requests rejected with explanation
  - new SessionIDs start with saturated RED level!
- effects
  - considerate users can work mostly undisturbed
  - hogging users are slowed (and educated)
  - session hopping is disadvantageous!

Stress Scenario #3
Massive Functions

- sometimes even perfect algorithms + data structures can't prevent ultra-heavyweight operations
  - in our playground: the massive analytics functions
  - 15–20 minutes run-time each
  - ≈ 10 simultaneous analytics viable
- solution
  - stacked RED filters
    - global performance metric
    - account-level filter
  - requests not rejected but queued
  - queue sorted according account-RED-level

Conclusion

- RED provides
  - no immunity against DOS or Brute-Force
  - but a resilient and a self-stabilizing system
  - choice of effective quick response options
  - headaches for the attackers
  - = more time for investigation and countermeasures
- IMHO
  - versatile, valuable + interesting tool
  - instrument worth considering in the tool-box …

Thank You for Listening

- **Questions?**
- **Contributions**
- **Discussion …**

- Thomas Maus