

# Correlation-Based Load Balancing for Network Intrusion Detection and Prevention Systems

Anh Le, Ehab Al-Shaer, and Raouf Boutaba  
Presented by Urs Hengartner

# Outline

---

1. Introduction
2. Objectives and Approach Overview
3. Problem Formulation
4. Flow Correlations
5. Online Clustering Technique
6. Implementation and Evaluation
7. Conclusions

# Introduction

---

- ▶ **Network Intrusion Detection and Prevention Systems (NIDPS)**
  - ▶ Located at the edge of the internal and external networks
  - ▶ Detect and prevent unauthorized uses and malicious activities by monitoring traffic
  - ▶ E.g: Cisco IPS, Bro+, and Snort.
  
- ▶ **The increasing traffic volume causes overload conditions:**
  - ▶ Upgrade hardware and tune software
  - ▶ Use clusters of NIDPSs

# Introduction (cont.)

---

- ▶ Simple traffic distributions (such as hash-based distribution) have two drawbacks:
  - ▶ Correlated flows may not be sent to the same NIDPS
  - ▶ Loads of the NIDPSs are not balanced
- ▶ Loss of correlation information:
  - ▶ Decreases anomaly-based detection accuracy
    - ▶ Intrusions might evade the NIDPSs.
- ▶ Uneven loads of the NIDPSs:
  - ▶ An NIDPS might become overloaded even when overall load is low
    - ▶ Overloaded NIDPS drops packets, which defeats the purpose of using clusters

▶ 4 Our proposed Load Balancer addresses the above two problems.

# Objectives and Approach Overview

---

## ▶ Objectives:

- ▶ Minimize the loss of correlation information
- ▶ Keep the difference between loads of the NIDPSs within a specified bound

## ▶ Approach overview:

- ▶ Loads of NIDPSs are closely monitored, and load balancing level is specified by a **variance constraint**
- ▶ Correlated flows are grouped into **clusters**
- ▶ A new flow is added to a cluster which gives the best **benefit** and satisfies the variance constraint

# Problem Formulation

---

**Maximize:**

(1)  $\vec{X} \cdot \vec{B}$

**Constraints:**

(2)  $\vec{X} \cdot \vec{I} \leq F$

(3)  $\vec{X} \cdot \vec{G}_i \leq 1, \forall i \in [1, n]$

(4)  $\frac{1}{n} \sum_{i=1}^n \left[ (L_i + L_f(\vec{X} \cdot \vec{G}_i)) - (\mu + L_f \frac{\vec{X} \cdot \vec{I}}{n}) \right]^2 \leq V$

**Where:**

$\vec{X}$  : Solution vector of size  $m$

$\vec{B}$  : Benefit vector of size  $m$

$\vec{G}_i$  : Cluster-ownership vector of size  $m$  of NIDPS  $i$

$\vec{I}$  : Vector of 1's of size  $m$

$F$  : Maximum number of NIDPSs to assign  $f$

$L_i$  : Load of NIDPS  $i$

$\mu$  : Average load of all NIDPSs

$L_f$  : Predicted load of  $f$

$V$  : Upper bound for the new variance

Figure 3.2: Flow Assignment Optimization Problem

# Heuristic Flow Assignment Algorithm

---

---

**Algorithm 1** HeuristicFlowAssignment( $f$ )

---

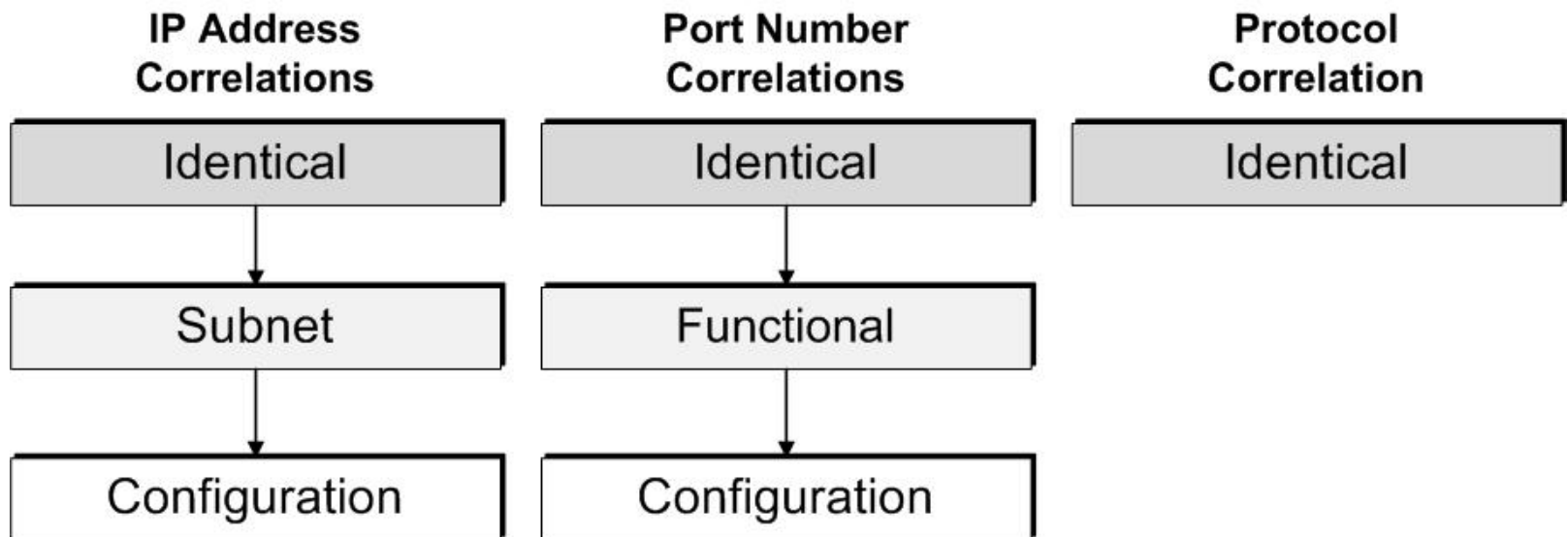
```
1:  solution_set =  $\emptyset$ 
2:  nidps_set = all NIDPSs
3:  for  $i$  from 1 to  $F$  do
4:    cluster_set = all clusters of nidps_set
5:    find cluster in cluster_set
   -    which satisfies variance constraint
   -    and has the biggest benefit
6:    if no cluster found then
7:      quit for loop
8:    end if
9:    solution_set = solution_set  $\cup$  cluster
10:   nidps = NIDPS which has cluster
11:   update load of nidps
12:   nidps_set = nidps_set  $\setminus$  nidps
13: end for
14: return solution_set
```

---

# Flow Correlations

- ▶ Given two flows, their **logical distance** is calculated as follows:

$$D(f_1, f_2) = \sum_{\forall i \in F} \alpha_i \delta_i(f_1, f_2)$$



# Online Clustering Technique

---

- ▶ Each cluster has a **centroid** and a **weight**
  - ▶ Centroid is a flow, representing the cluster
  - ▶ Weight of a cluster decays over time
- ▶ When a new flow  $f$  comes, the **benefit** of adding flow  $f$  to cluster  $j$  is calculated as follows:

$$B_j = (1 - D(f, c_j))W_{j,t}$$

- ▶ This formula ensures that:
  - ▶ The higher the weight, the higher the benefit
  - ▶ The more correlation between flow  $f$  and cluster  $j$ , the more benefit

# Benefit-Based Load Balancing Algorithm

---

---

## Algorithm 2 Benefit-based Load Balancing Algorithm

---

```
1:  while new flow  $f$  do
2:     $C = \text{HeuristicFlowAssignment}(f)$ 
3:    if  $C = \emptyset$  or  $\text{Benefit}(C) < T_{benefit}$  then
4:      create a cluster (centroid  $f$ , weight 1)
5:      assign it to the lowest load NIDPS
6:    else
7:      assign  $f$  to the clusters in  $C$ 
8:      update those clusters
9:    end if
10: end while
```

---

# Implementation and Evaluation

---

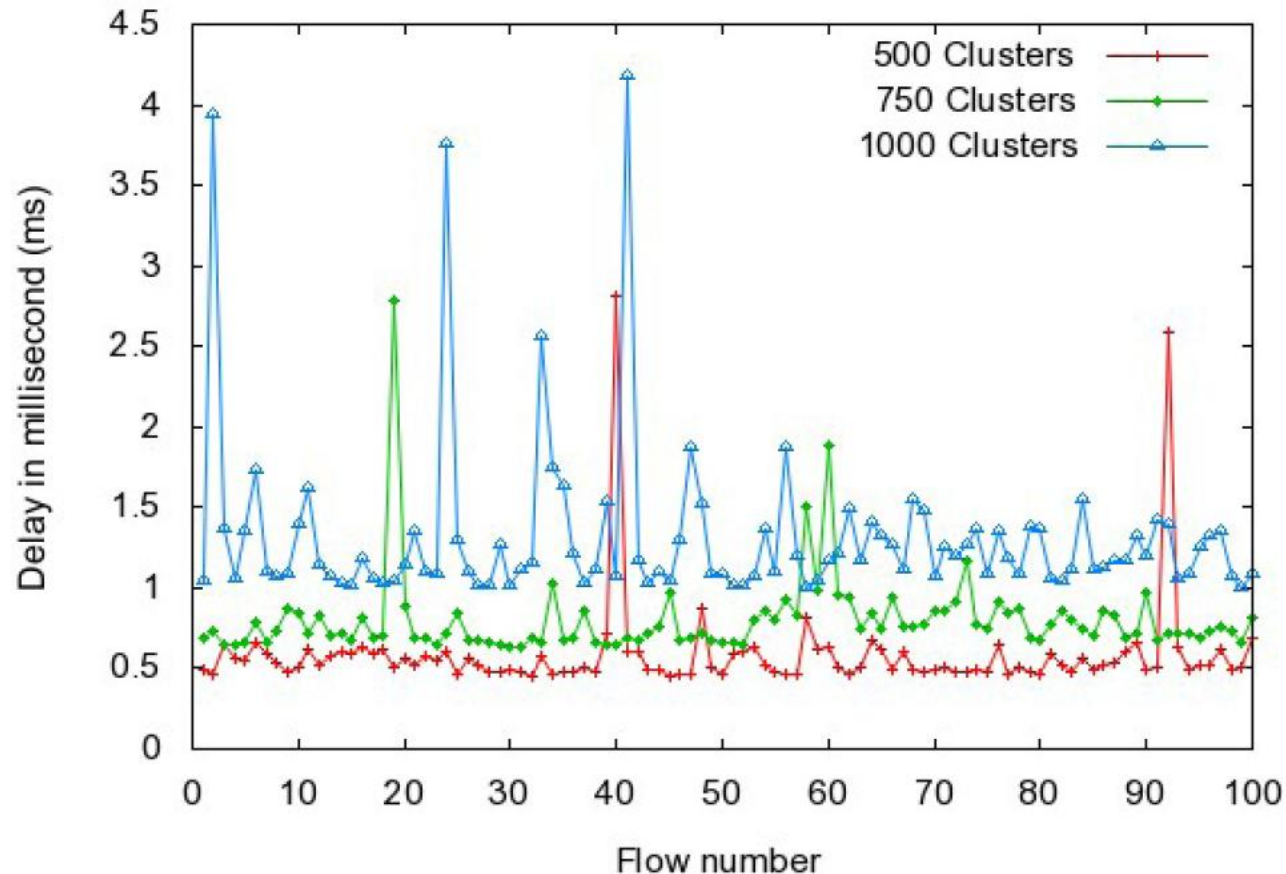
## ▶ Implementation:

- ▶ A prototype Load Balancer is implemented using libpcap
  - ▶ The BLB and a simple additive hash-based algorithms are implemented
- ▶ A distributed denial of service (DDoS) detector using Cumulative Sum algorithm is implemented

## ▶ Evaluation:

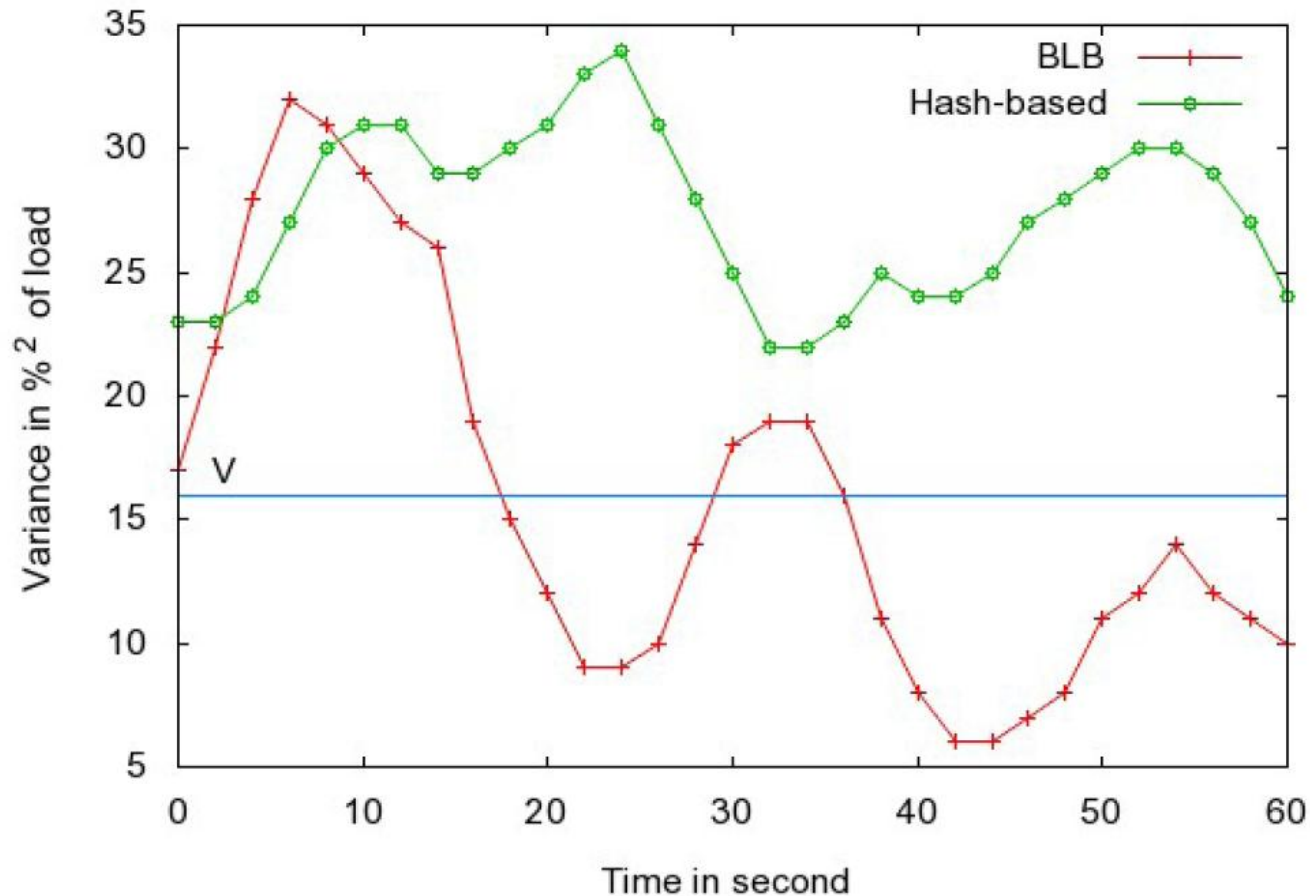
- ▶ Performance:
  - ▶ System overhead
  - ▶ Variance
- ▶ Security
  - ▶ DDoS attacks
  - ▶ Port scans

# Evaluation – Performance: System Overhead



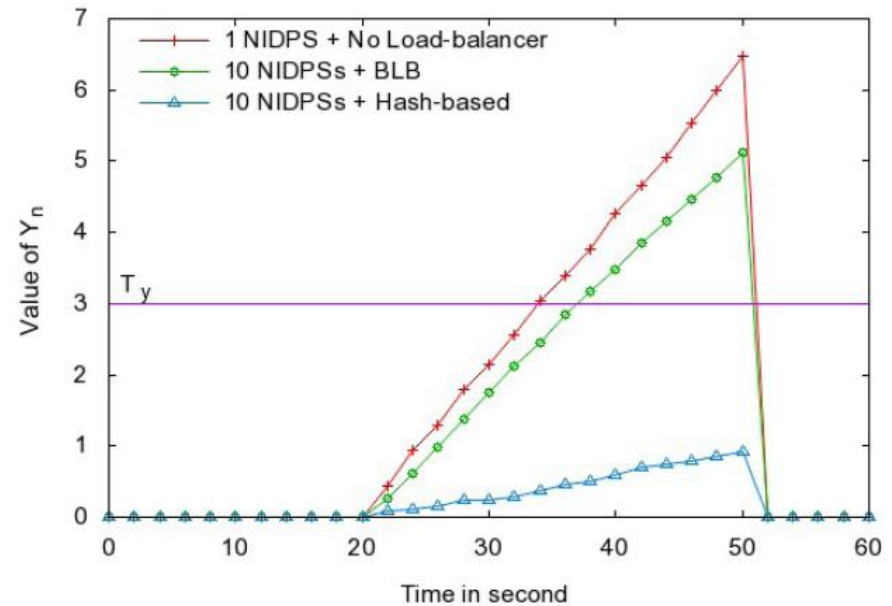
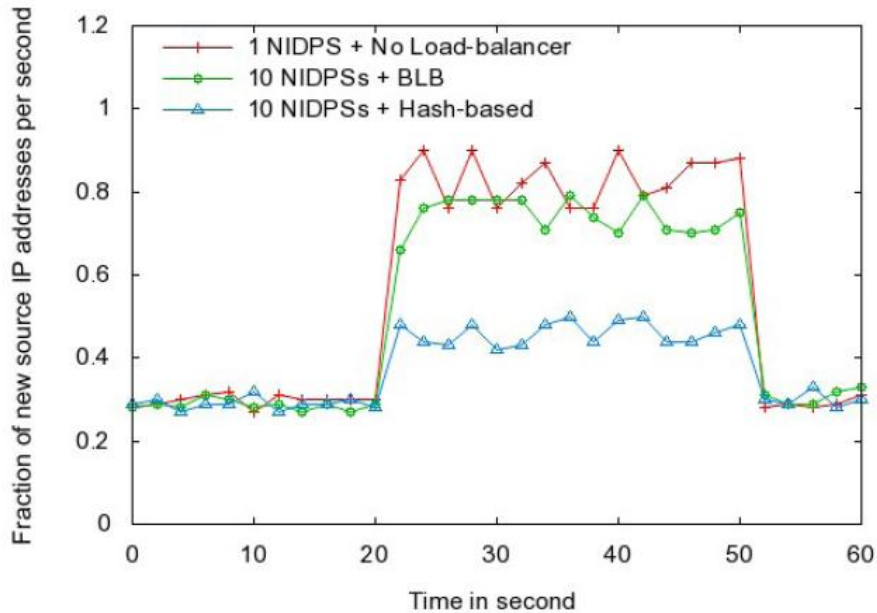
Effect of Number of Clusters on System Overhead

# Evaluation – Performance: Variance



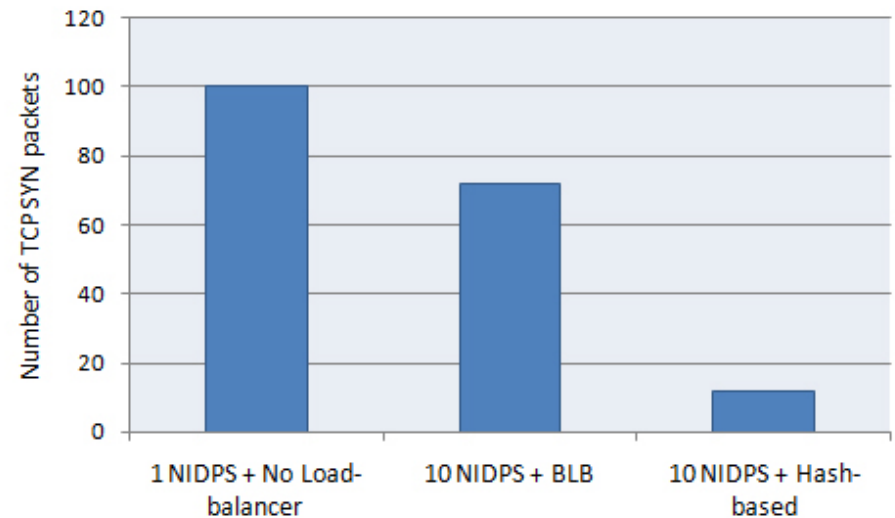
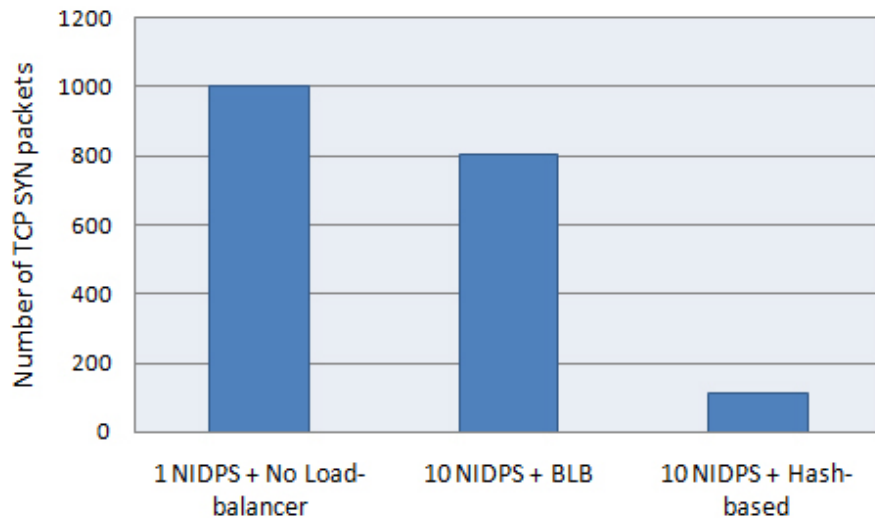
Effect of Algorithms on the Variance

# Evaluation – Security: DDoS



Effect of Algorithms on Fraction of new source IP addresses and on  $Y_n$

# Evaluation – Security: Port scan



Effect of Algorithms on the Grouping of Portscan Packets (left) and Portsweep Packets (right)

# Conclusions

---

- ▶ The problem of load balancing in the context of network intrusion detection and prevention is challenging.
  - ▶ A traffic distribution scheme should take into account both the traffic correlation and the load balancing
- ▶ We proposed the Correlation-Based Load Balancer to provide real-time traffic distribution with the following properties:
  - ▶ The loss of correlation information is minimized
  - ▶ The difference between loads of the NIDPSs is kept within a specified bound
- ▶ The evaluation results show that the Load Balancer has
  - ▶ Low overhead
  - ▶ Improved load balancing
  - ▶ Improved detection accuracy of DDoS attacks and port scans