BotGraph: Large Scale Spamming Botnet Detection

Yao Zhao

Yinglian Xie*, Fang Yu*, Qifa Ke*, Yuan Yu*, Yan Chen and Eliot Gillum‡

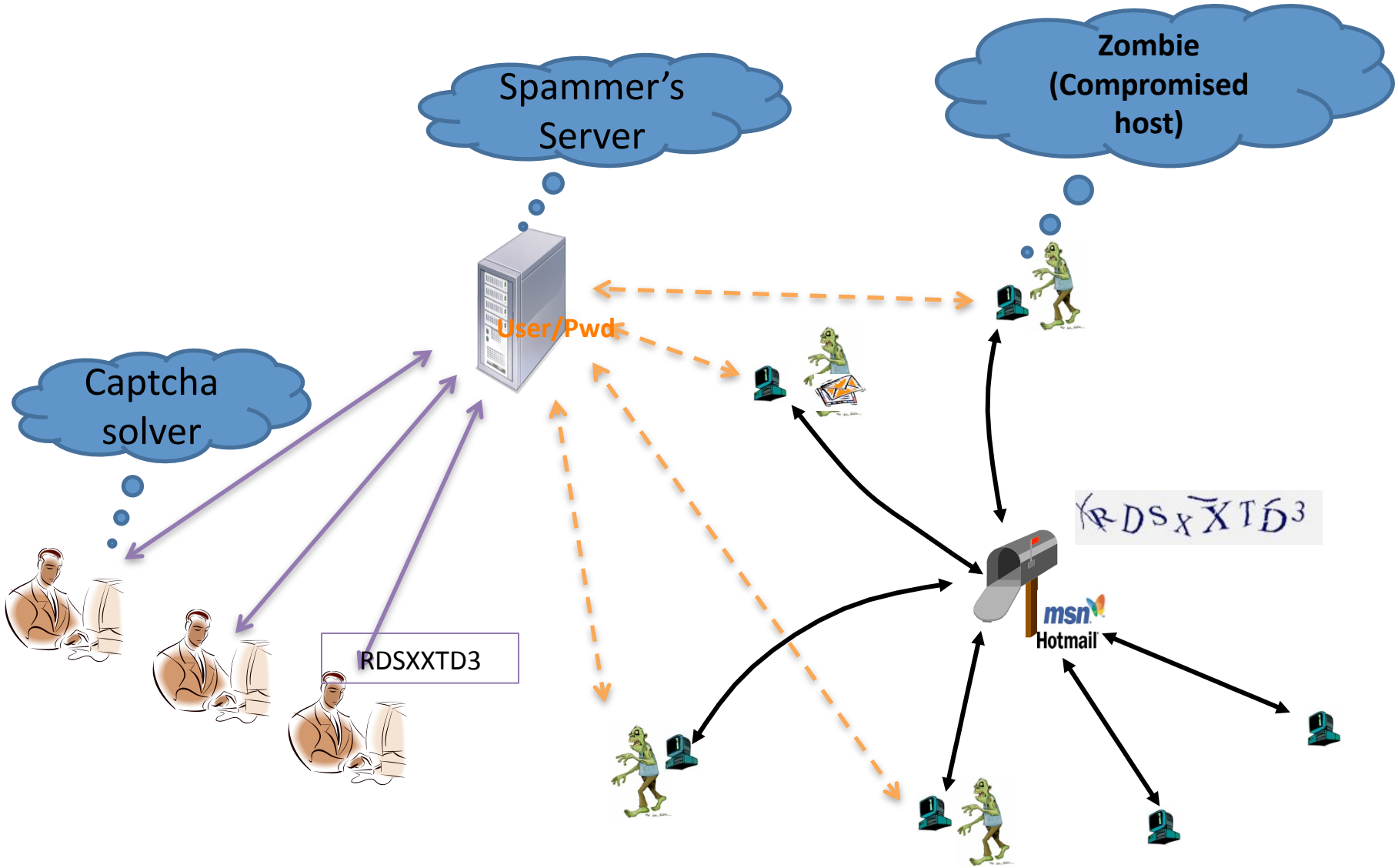
EECS Department, Northwestern University

Microsoft Research Silicon Valley*

Microsoft Cooperation ‡

Microsoft® Research

Web-Account Abuse Attack



Problems and Challenges

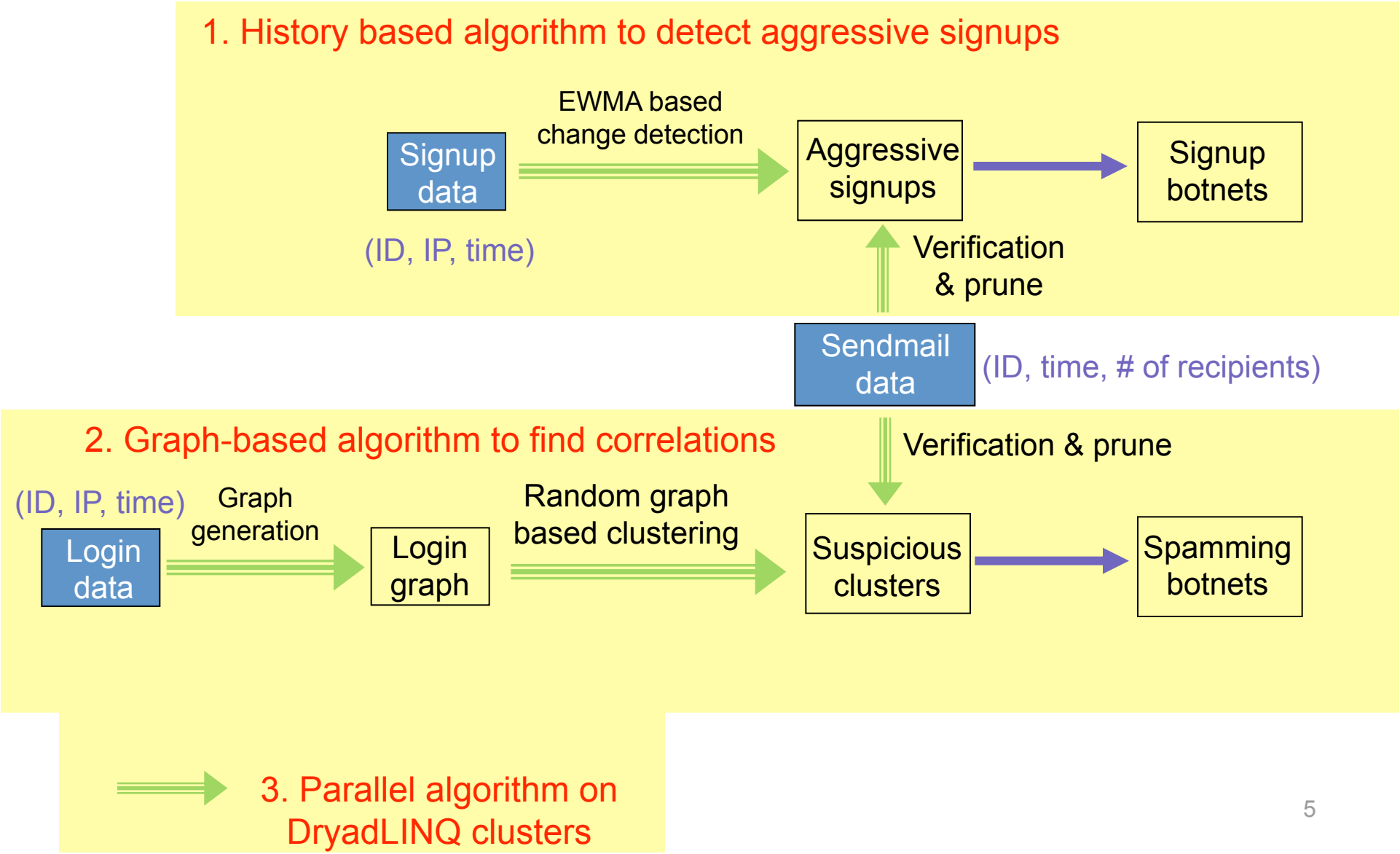
- Detect Web-account Abuse with Hotmail Logs
  - **Input:** user activity traces (signup, login, email-sending records)
  - **Goal:** stop aggressive account signup, limit outgoing spam
- Algorithmic challenge:
  - Attack is stealthy: individual account detection difficult
  - Attack is large scale: finding correlated activities
  - Low false positive and false negative rate
- Engineering challenge:
  - Large user population: >500 million accounts
  - Large data volume: 300GB-400GB data per month

The BotGraph System

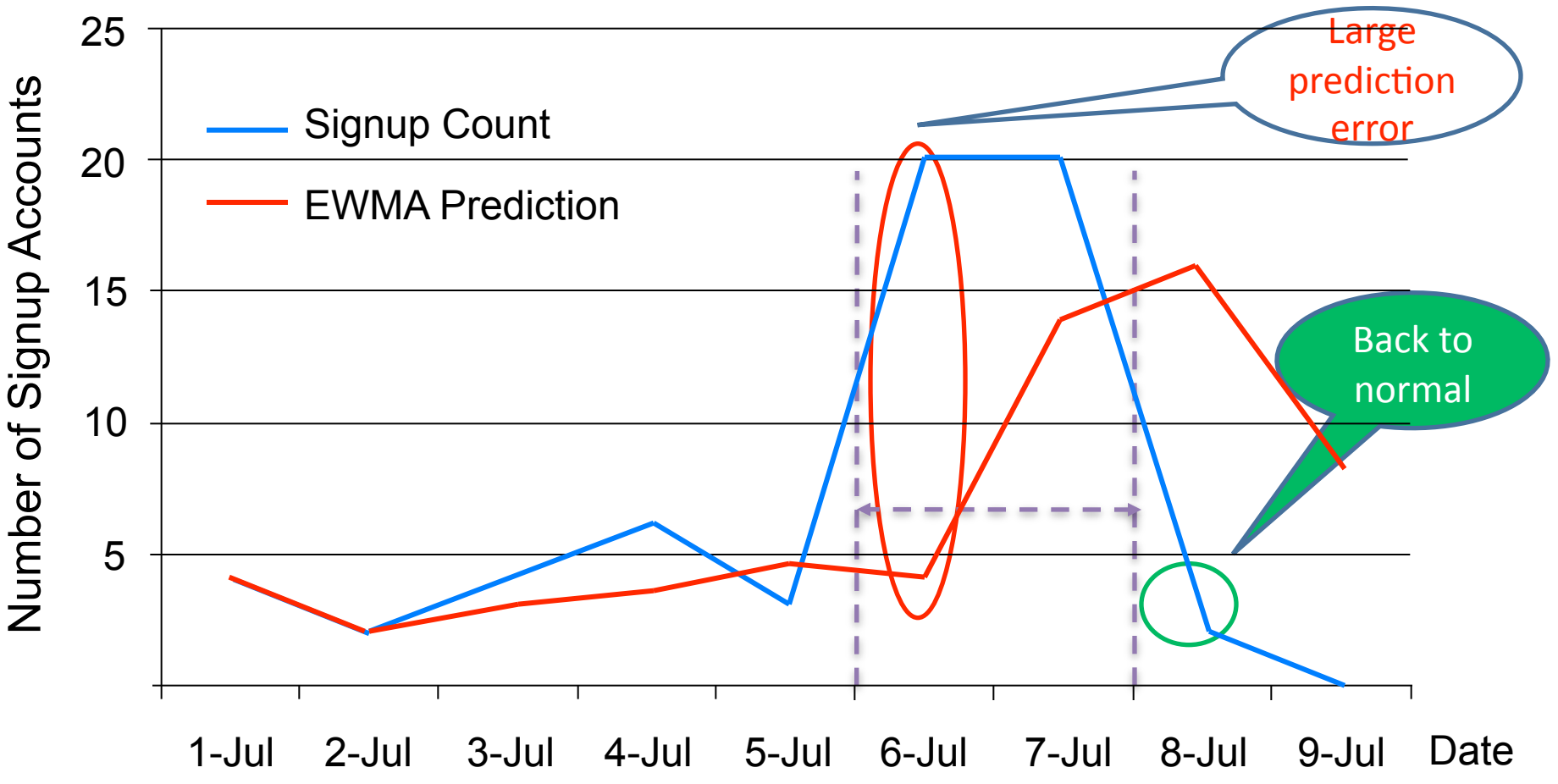
- **A graph-based approach to attack detection**
  - A large user-user graph to capture bot-account correlations
  - Identify 26M bot-accounts with a low false positive rate in two months
- **Efficient implementation using Dryad/DryadLINQ**
  - Graph construction/analysis is not easily parallelizable
  - hundreds of millions of nodes, hundreds of billions of edges
  - Process 200GB-300GB data in 1.5 hours with a 240-machine cluster

The first to provide a systematic solution to the new attack

System Architecture

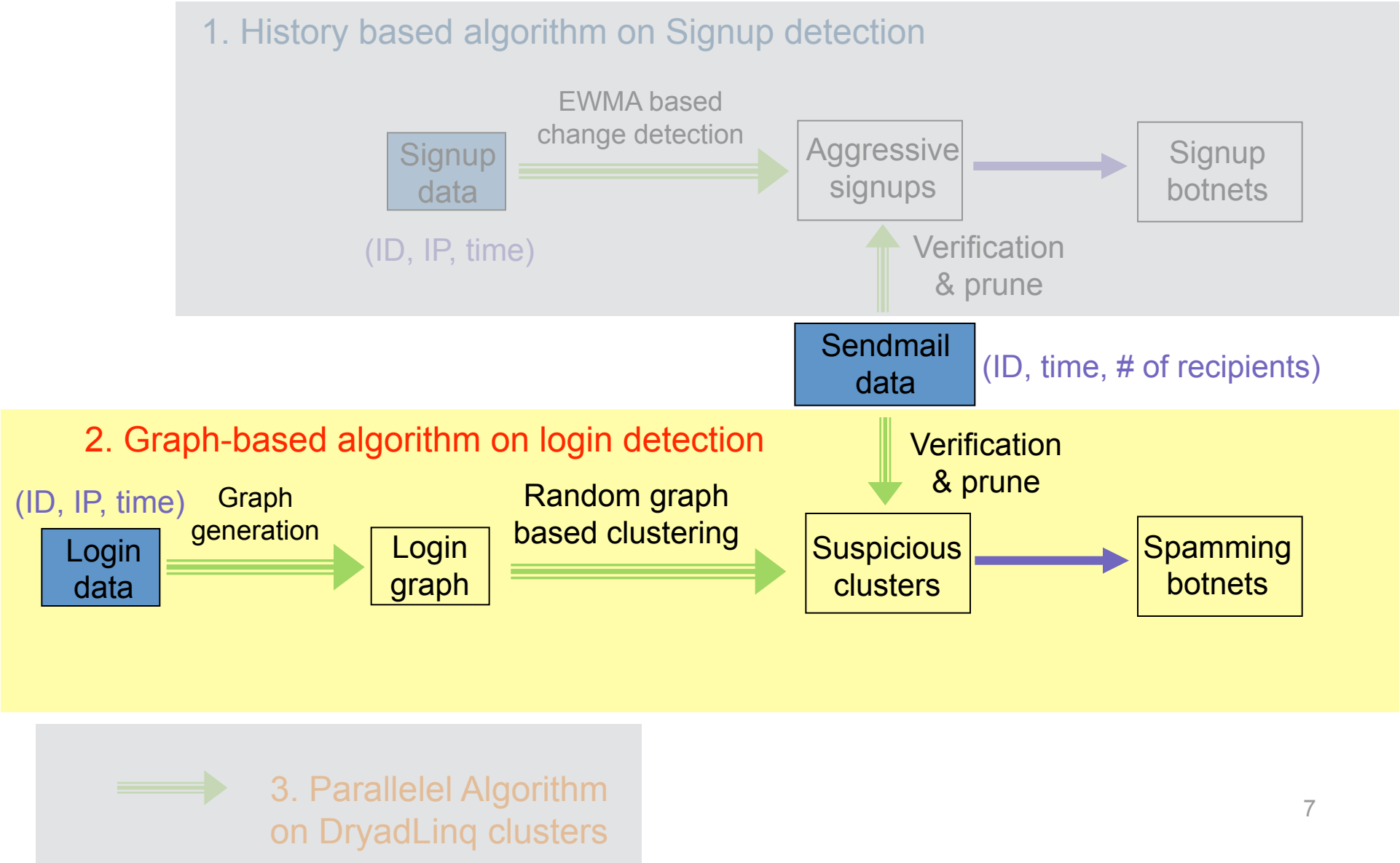


Detect Aggressive Signups



- Simple and efficient
- Detect 20 million malicious accounts in 2 months

System Architecture

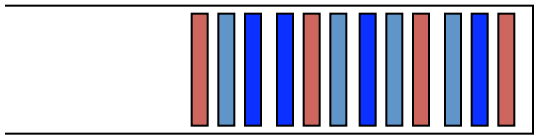


Detect Stealthy Accounts by Graphs

- Observation: bot-accounts work collaboratively

A user-user graph to model behavior similarities

- Normal Users
  - Share IP addresses in one AS with DHCP assignment
- Bot-users

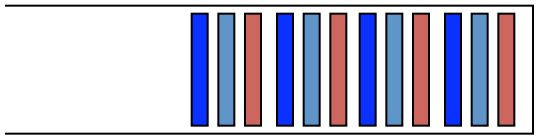


Detect Stealthy Accounts by Graphs

- Observation: bot-accounts work collaboratively

A user-user graph to model behavior similarities

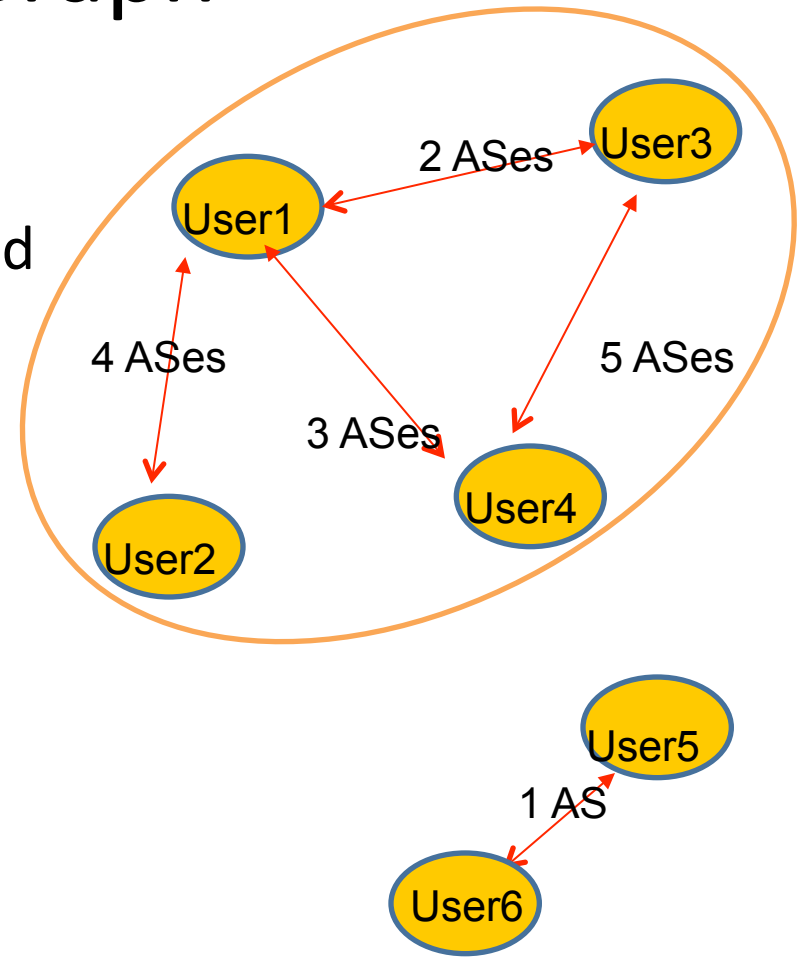
- Normal Users
  - Share IP addresses in one AS with DHCP assignment
- Bot-users
  - Likely to share different IPs across ASes



User-user Graph

- Node: Hotmail account
- Edge weight: # of ASes of the shared IP addresses
  - Consider edges with weight>1

- Key Observations
  - **Bot-users form a giant connected-component while normal users do not**
  - Interpreted by the **random graph theory**



Random Graph Theory

- Random Graph $G(n,p)$
  - n nodes and each pair of nodes has an edge with probability p and average degree $d = (n\text{-}1) \cdot p$
- Theorem
  - If $d < 1$, then with high probability the largest component in the graph has size less than O(log n)

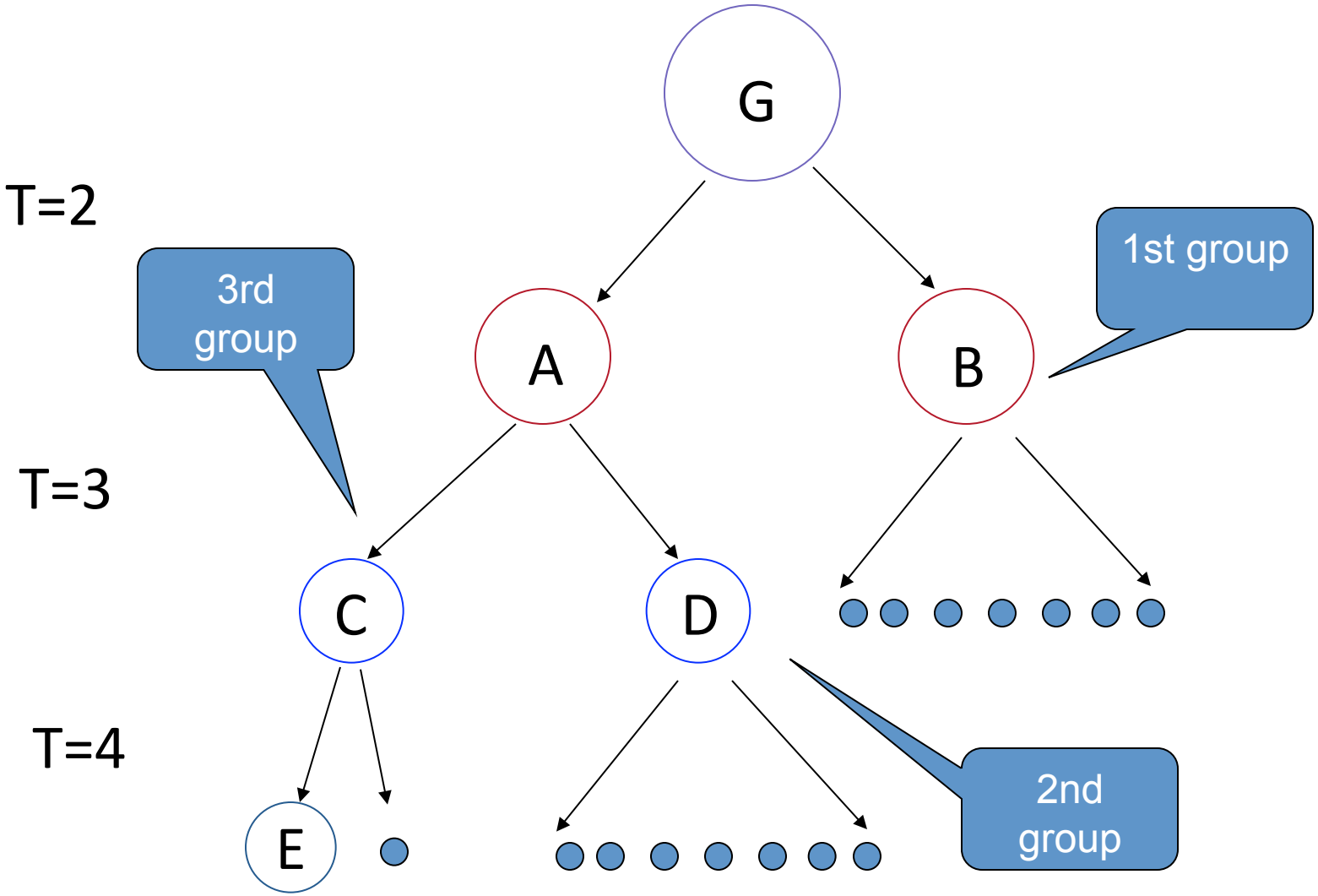
    ⟹ **No large connected subgraph**
  - If $d > 1$, with high probability the graph will contain a giant component with size at the order of O(n)

    ⟹ **Most nodes are in one connected subgraph**

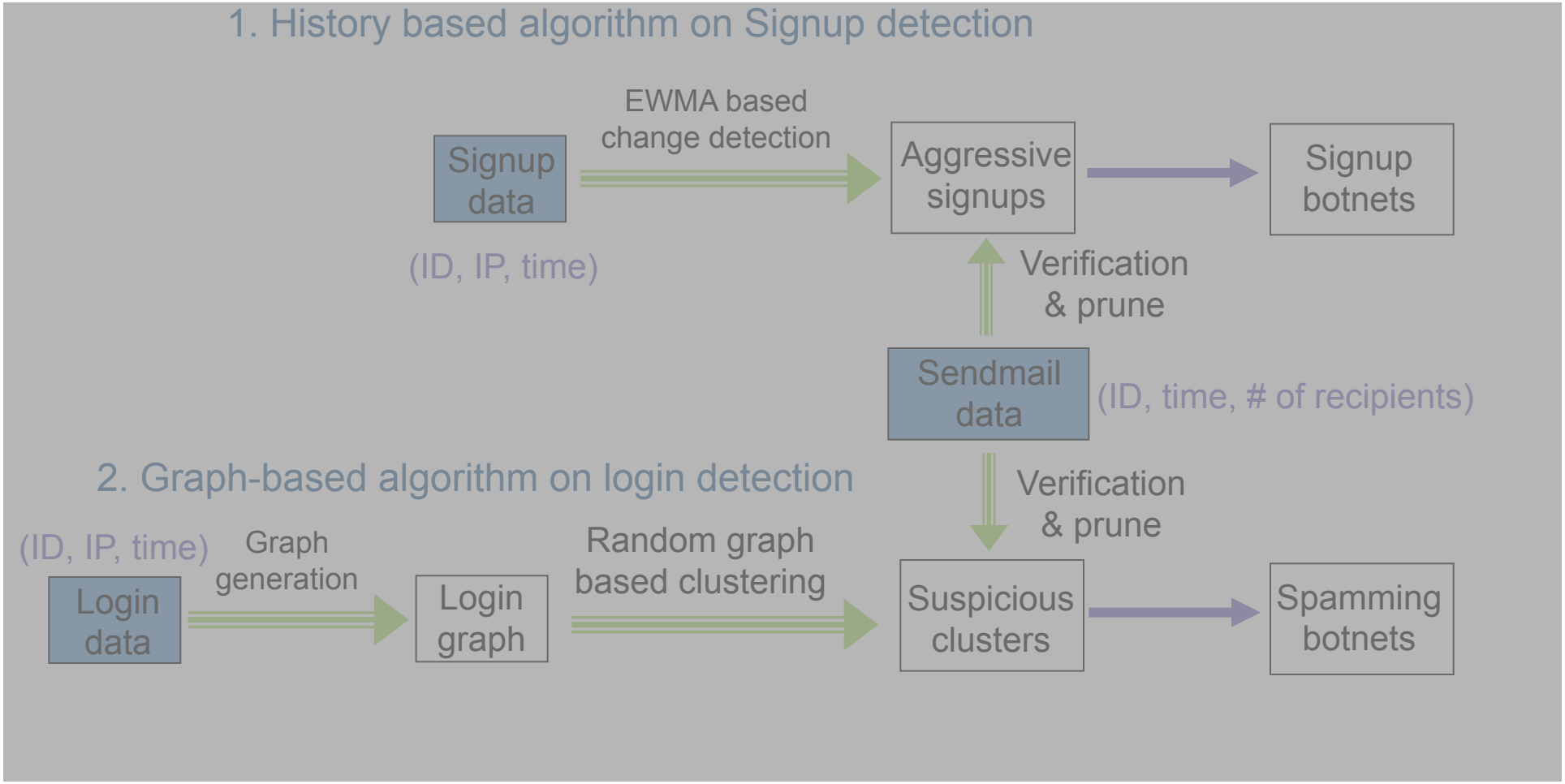
Graph-based Bot-user Detection

- **Step 1:** detect giant connected-components from the user-user graph
- **Step 2:** hierarchical algorithm to identify the correct groupings
  - Different bot-user groups may be mixed
  - Difficult to choose a fixed edge-threshold
  - Easier validation with correct group statistics
- **Step 3:** prune normal-user groups
  - Due to national proxies, cell phone users, facebook applications, etc.

Hierarchical Bot-Group Extraction



System Architecture



3. Parallelel Algorithm on DryadLINQ clusters

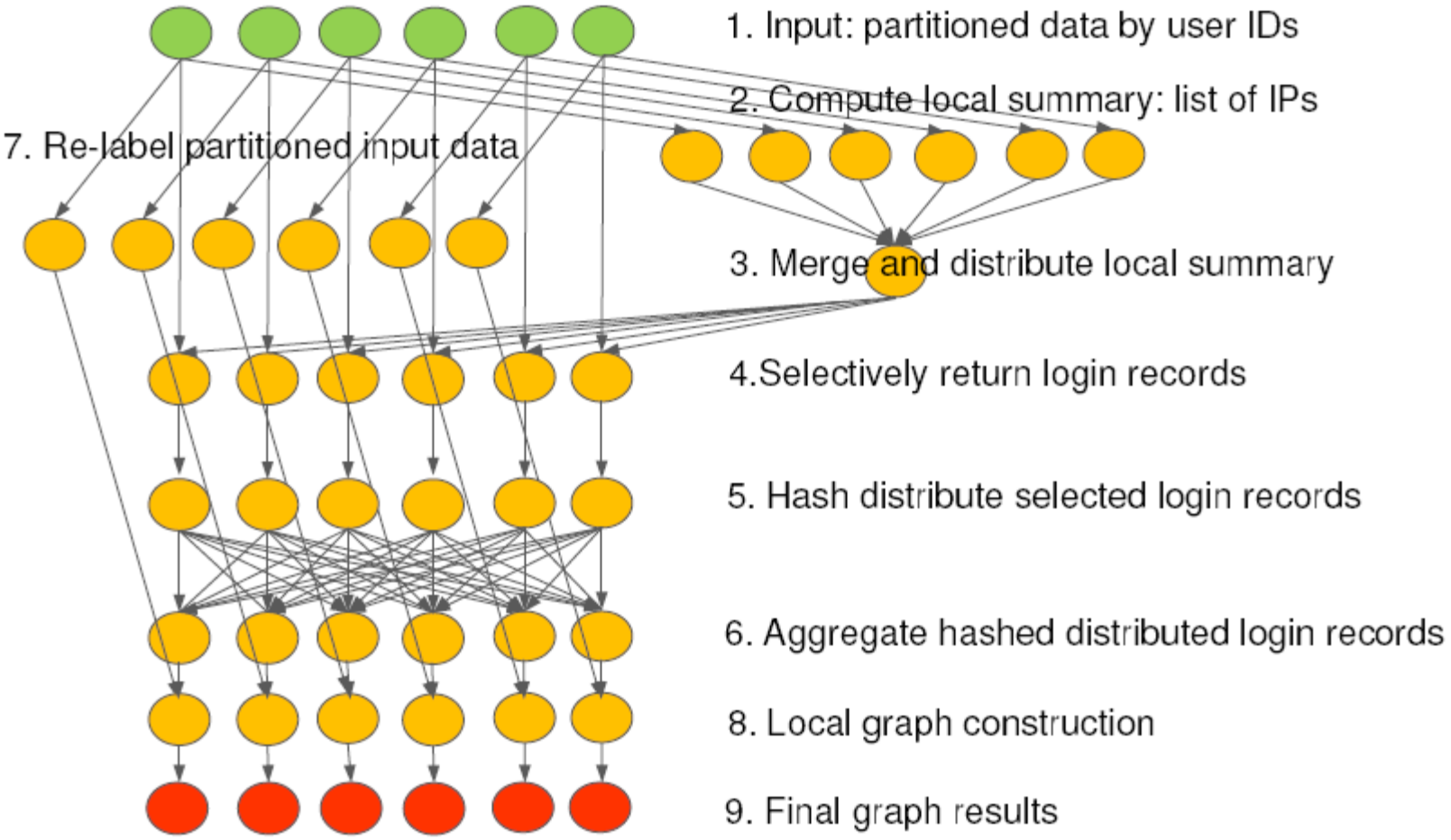
Parallel Implementation on DryadLINQ

- EWMA-based Signup Abuse Detection
  - Partition data by IP
  - **Can achieve real-time detection**
- User-User Graph Construction
  - Two algorithms and optimizations
  - **Process 200GB-300GB data in 1.5 hours with 240 machines**
- Connected Component Extraction
  - Divide and conquer
  - **Process a graph of 8.6 billion edges in 7 minutes**

Graph Construction 1: Simple Data Parallelism

- Potential Edges
  - Select ID group by IP (Map)
  - Generate potential edges (ID_i, ID_j, IP_k) (Reduce)
- Edge Weights
  - Select IP group by ID pair (Map)
  - Calculate edge weight (Reduce)
- Problem
  - **Weight 1 edge is two orders of magnitude more than others**
  - **Their computation/communication is unnecessary**

Graph Construction 2: Selective Filtering



Comparison of Two Algorithms

- Method 1
  - Simple and scalable
- Method 2
  - Optimized to filter out weight 1 edges
  - Utilize Join functionality, data compression and broadcast optimization

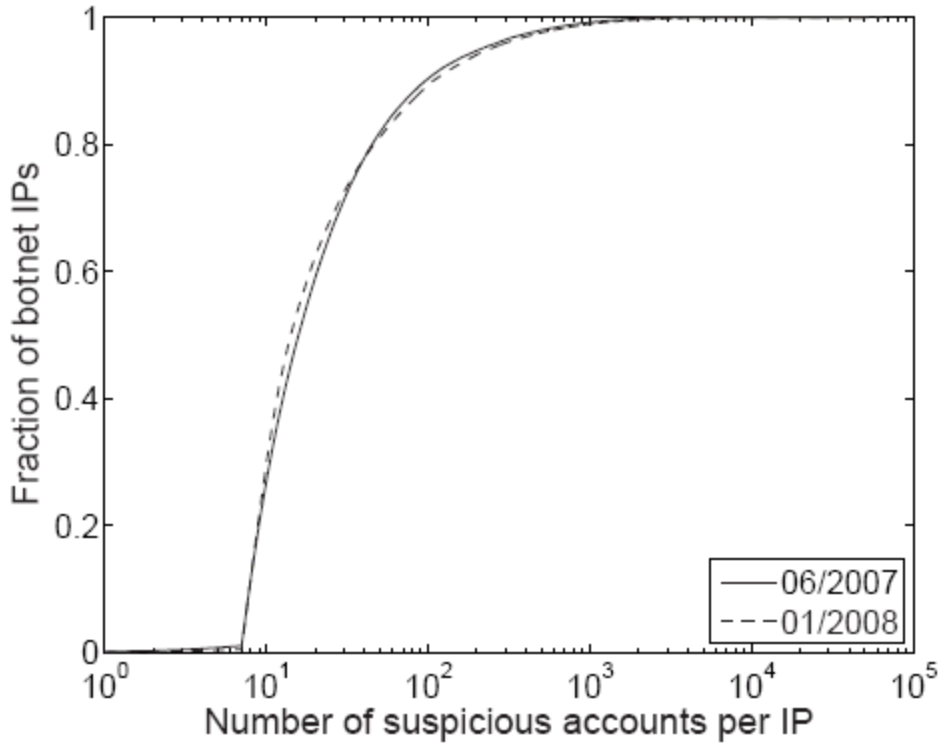
	Communication data size	Total running time
Method 1 (no comp.)	2.71 TB	135 min
Method 1 (with comp.)	1.02 TB	116 min
Method 2 (no comp.)	460 GB	28 min
Method 2 (with comp.)	181 GB	21 min

Detection Results

- Data description
  - Two datasets
    - Jun 2007 and Jan 2008
  - Three types of data
    - Signup log (IP, ID, Time)
    - Login log (IP, ID, Time)
      - 500M users and 200~300GB data per month
    - Sendmail log (ID, time, # of recipients)

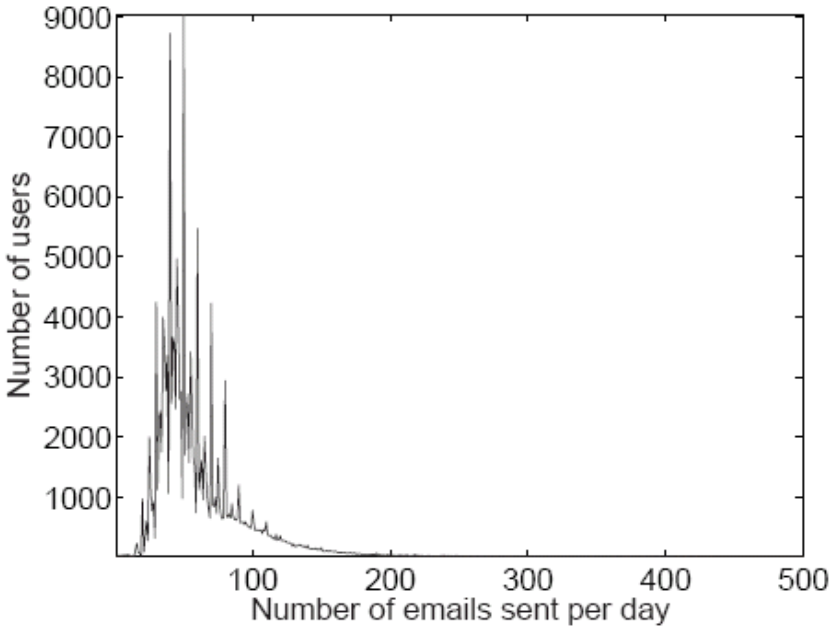
Detection of Signup Abuse

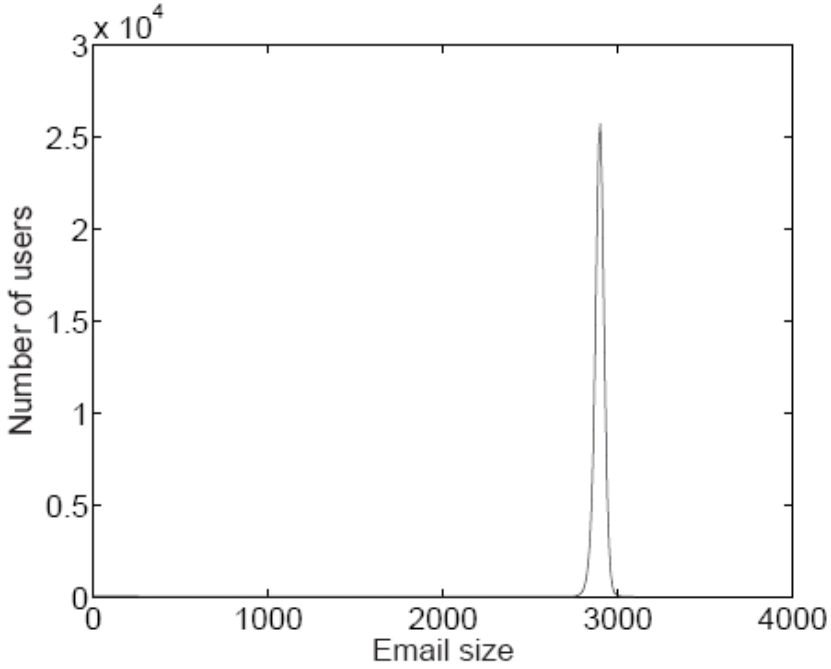
Month	06/2007	01/2008
# of bot IPs	82,026	240,784
# of bot-user accounts	4.83 M	16.41 M
Avg. anomaly window	1.45 day	1.01 day



Detection by User-user Graph

Month	06/2007	01/2008
# of bot-groups	13	40
# of bot-accounts	2.66M	8.68M
# of unique IPs	2.69M	1.60M





Validations

- Manual Check
  - Sampled groups verified by the Hotmail team
  - Almost no false positives
- Comparison with Known Spamming Users
  - Detect 86% of complained accounts
  - Up to 54% of detected accounts are our new findings
- Email Sending Sizes per Group
  - Most groups have a sharp peak
  - The remaining contain several peaks
- False Positive Estimation
  - Naming pattern (0.44%)
  - Signup time (0.13%)

Possible to Evade BotGraph?

- Evade signup detection: Be stealthy
- Evade graph-based detection
  - Fixed IP/AS binding
    - Low utilization rate
    - Bot-accounts bound to one host are easy to be grouped
  - Be stealthy (sending as few emails as normal user)

⟹ **Severely limit attackers' spam throughput**

Conclusions

- **A graph-based approach to attack detection**
  - Identify 26M bot-accounts with a low false positive rate in two months
- **Efficient implementation using Dryad/DryadLINQ**
  - Process 200GB-300GB data in 1.5 hours with a 240-machine cluster

Large-scale data-mining for network security is effective and practical

Q & A?

Thanks!