Graph-based Detection of Anomalous Network Traffic

Do Quoc Le

Supervisor: Prof. James Won-Ki Hong

Distributed Processing & Network Management Lab
Division of IT Convergence Engineering
POSTECH, Korea
lequocdo@postech.ac.kr

2012. 06. 22
Contents

- Introduction & Motivation
- Related Work
- Graph-based Network Traffic Modeling
- Graph Metrics
- Anomaly Detection & Attack Identification
- Validation
- Conclusion
Introduction & Motivation

- The Internet continues to grow in size and complexity
  - Security has become a critical issue.
  - The occurrence of traffic anomalies (DDoS, flash crowds, port scans and worms).

- Challenges:
  - Increasingly sophisticated attacks.
  - Attacks are often hidden in existing applications, e.g. IRC, HTTP, or Peer-to-Peer: Worm scans or botnet C&C traffic.

- Methods for detecting traffic anomalies.
  - Signature-based techniques
    - Cannot detect anomalies caused by unknown attacks.
  - Anomaly-based techniques: (Machine learning, data mining the statistical analysis, etc.)
    - Generate a huge number of false alarms.
    - Time consuming.
    - Cannot detect anomalies whose traffic is similar with normal applications (traffic volume, number of packets, number of flows and average packet size).
Goal: Improve detection accuracy and the ability of the state of art techniques for anomaly detection.

Solution:
- Using a graph-based method to monitor network traffic and analyze the structure of communication patterns to detect anomalies and identify attacks.

Why we study the structure of communication patterns in network traffic?
- Each attack has its own structure.
- Communication patterns’ structure changes when attacks occur.
- Can identify when attacks occur that can be difficult to detect using conventional means.
Contribution

- One of the first works using a Traffic Dispersion Graphs (TDGs) to detect anomalies
  - Focus on structural characteristics of networks.
  - Improve performance and ability of the state of the art techniques.
  - Support intuitive visualization of traffic patterns.
- Introduce a new metric to analyze network traffic communication patterns overtime
- Implement an online anomaly detection system in an Enterprise network based on the proposed method
- Evaluate the approach by analyzing real attack traces
Related Work

- **Zhou et al. [1]** proposed a network traffic anomaly method based on graph mining
  - Mining time-series graphs.
  - Mining edge weight.
  - Entropy of four attributes: source and destination IP address, source and destination port.
  - The drawback: Enormous size → computational complexity.

  We analyze unlabeled graphs and just concentrate on their nodes

- **Godiyal et al. [2]** used a graph matching method to identify attacks
  - Applying isomorphism algorithm for whole traffic flow → very time consuming.

  We identify attacks in abnormal network traffic only
Related Work (cont.)

- Iliofotou et al. [3] use TDG to model network traffic as series of related graphs over time
  - Using graph metrics
    - Degree, degree distribution
    - Entropy of degree distribution
    - Graph edit distance
  - Solving problem of traffic classification, possible application to anomaly detection.

We model network traffic as TDG over time using new metrics.
Network Traffic Modeling

- Traffic Dispersion Graph (TDG)
  - Each node → IP address.
  - Each edge → interaction (flow) between two nodes.
**TDG Visualization**

- **HTTP**
  - Many disconnected components
  - Very few nodes with in and out degrees
    - Web proxies?

- **Slammer Worm**
  - UDP Dst. port 1434
  - Many high out-degree nodes
  - Many disconnected components
  - The majority of nodes have only in-degree
    - Nodes being scanned

*Source: Illofotou et al.*
Graph Metrics on TDGs

- **What we have seen so far:** “Visualization is useful by itself”
  - However, it requires a *human operator.*

- **Next step?**
  - Translate *visual intuition* into *quantitative measures.*

- **How to quantitatively characterize properties of TDGs?**
  - **Step 1:** represent traffic as a sequence of graph snapshots.
  - **Step 2:** use metrics that quantify differences between graphs.

What are the differences in communication structure between $G_x$ and $G_y$?
Graph Metrics on TDGs – Static metrics

❖ **Node degree**
  - In-degree
  - Out-degree

❖ **Degree distribution**
  - Show an approximate power-law.

❖ **Maximum degree (Kmax)**
  - One of metrics to detect DDoS attack.

❖ **Degree Assortativity**
  - Measure the tendency for nodes to be connected to similar nodes in term of their degree.

❖ **Entropy of degree distribution**
  - Quantify heterogeneity of network: \( H(X) = - \sum_{k=1, k_{\text{max}}} P(k) \log(P(k)) \)

Where \( P(k) \) is the probability that a node has degree \( k \).
Graph Metrics on TDGs – Dynamic metrics

- **Graph edit distance**

  \[ d(G_i, G_j) = |V_i| + |V_j| - 2|V_i \cap V_j| + |E_i| + |E_j| - 2|E_i \cap E_j| \]

  Where \( V_i \), \( E_i \) and \( V_j \), \( E_j \) are the numbers of nodes and edges in graph \( G_i \) and \( G_j \), respectively.

- **dK-2 distance metric**
  - Based on dK-series concept
    - Structure analysis - dK-n series: \( n=1,2,3,... \)
    - Look at inter-dependencies among topology characteristics.
    - dK-n series are degree correlations within simple connected graphs of size \( n \).
    - dK-2 describes joint node degree distribution.
  - \( dK-2 \text{ distance}(G,G') = \text{Euclidean distance between } dK-2(G) \text{ and } dK-2(G') \)
Anomaly Detection & Attack Identification

- Using graph metrics to detect abnormal network traffic.
- Anomalies: attacks which change communication structure in network (DDoS attacks, Internet worms and scanning).
- The overall process consist of two parts: anomaly detection and attack identification.

Figure 4. Overall detection process.
Anomaly Detection & Attack Identification

- **Anomaly Detection**
  - **Step 1**: Sampling network traffic and generating network flows.
  - **Step 2**: Creating TDG (Dot format) from network flows in time sampling intervals.
  - **Step 3**: Calculating adjacency matrices of the TDG and calculating graph metrics of the TDG.
  - **Step 4**: Comparing values of graph metrics of the TDG with their threshold value.
    - Graph metric value < Threshold → normal TDG.
    - Graph metric value > Threshold → abnormal TDG.

*Figure 5. Detailed anomaly detection process.*
Anomaly Detection & Attack Identification

**Attack Identification**

- **Attack pattern:**

- **Attack identification:**

  - Unknown anomalies
  - Abnormal TDGs

  **Graph-subgraph isomorphism (VF2)**

  - Yes: Attack types
  - No: attack pattern generation process

  **Figure 7. Attack pattern generation process.**

  **Figure 8. DDoS attack pattern in DDoS CAIDA trace.**

  **Figure 9. Peacomm P2P botnet pattern.**

  **Figure 11. Attack identification process.**
Validation

❖ Off-line analysis

❖ Trace
  • DARPA 1999 Dataset
    – Week 1 and week 3: no attack (for training data).
    – Week 2: 43 attacks belonging to 18 labeled attack types are used for system development.
    – Week 4 and week 5: 201 attacks belonging to 58 attack types (including 40 new attacks).
  • POSTECH trace in 2009. 7. 9.
    – Contain a famous DDoS attack on July 7, 2009 in South Korea.
  • CAIDA DDoS trace in 2007.
  • P2P Botnet trace (Peacomm) from a honeynet.

❖ On-line analysis
  • Real-time anomaly detection
    • Testing with port scanning attack
Validation (DARPA dataset)

**DARPA 1999 Dataset**

![Graphs showing Kmax per minute over one day (Monday, Week 5) with normal and attacking traffic.](image)

**Figure 12.** Kmax per minute over one day (Monday, Week 5) with normal and attacking traffic.

![Graphs showing dK-2 distance value per minute over one day (Monday, Week 5) with normal and attacking traffic.](image)

**Figure 13.** dK-2 distance value per minute over one day (Monday, Week 5) with normal and attacking traffic.
Validation (DARPA dataset)

• DARPA 1999 Dataset

Table 2. Performance of the Graph-based method using Kmax and dK-2 distance metric on Monday, Week5 traffic.

<table>
<thead>
<tr>
<th>Total instances</th>
<th>Attacking instances</th>
<th>DR</th>
<th>FPR</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1320</td>
<td>122</td>
<td>100</td>
<td>1.25</td>
<td>98.86</td>
</tr>
</tbody>
</table>

Table 3. Number of attack instances detected for each attack type on Monday, Week5 traffic.

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Number of attack instances for each attack type</th>
<th>Number of detected attack instances for each attack type</th>
</tr>
</thead>
<tbody>
<tr>
<td>apache2-dos</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>arppoison-probe</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>dict-r2l</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>guesstelnet-r2l</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ipsweep-prob</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>ls-probe</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>neptune-dos</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>portsweep-probe</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>smurf-dos</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>udpstorm-dos</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>crashiis-dos</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Validation (POSTECH July, 2009)

POSTECH traces on July, 2009

<table>
<thead>
<tr>
<th>Date</th>
<th>DDoS Attack</th>
<th>Trace Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/31</td>
<td>No</td>
<td>30.7 GB</td>
</tr>
<tr>
<td>07/08</td>
<td>Yes</td>
<td>27.3 GB</td>
</tr>
</tbody>
</table>

Graphs showing the number of packets, size information, number of flows, Kmax, and Dk2 over time.
Figure 15. Kmax value over time of POSTECH’s trace on July 8th 2009.

Figure 16. dK-2 distance value over time of POSTECH’s trace on July 8th 2009.
Validation (POSTECH July, 2009)

- POSTECH traces on July, 2009

Postech Normal Trace in 2009

Postech DDoS Trace in 2009.7.9
Validation (Honeynet dataset)

- **Real P2P botnet traffic (Peacomm) trace**
  - We executed Trojan Peacomm binary files in a honeynet which consisted of 12 hosts.

- **Synthesized traffic dataset**
  - We injected P2P botnet (Peacomm) trace into normal POSTECH traffic trace.
Validation (Honeynet Dataset)

❖ Results

- Results:
  - dK-2 Matrices
  - Normal
  - Anomaly

- Graphs showing time series data for different metrics such as Size (GB), Flows, and dK-2 distance.
Validation (Real-time anomaly detection)

The real-time anomaly detection system

- Network Traffic
- Flow Generation
  - Flow Store (binary files)
- TDGs Generation
- Graph Metrics Analysis
- TDG Graphs (dot files)
- Time-series Chart Generation
- TDG graphs image creation
- Graph-based Analysis Results
- Anomaly Classification
- Notification (Email, SMS, and etc.)
- User Interfaces
- User

Figure 22. Real-time Anomaly Detection System: Function diagram.

Figure 23. Real-time Anomaly Detection System: User Interface.
Validation (Real-time anomaly detection)

- **Real-time anomaly detection system testing**
  - We implemented a Port scanning attack from a host in the dormitory network of our campus to a host outside our campus network.
    - Using TCP Port Scanning tool to generate 100 Port scanning instances
  - Result: DR = 100% and FP = 0.

Figure 24. dK2 distance and Kmax value during TCP Port scanning attacks.
Conclusion & Future Work

❖ Conclusion

❖ Provide a new approach for anomaly detection.
   • Improve performance of the state of the art techniques.
❖ Implement a real-time anomaly detection system based on the proposed method.
❖ New way to analyze network traffic for anomaly detection that offers clear visualization.

❖ Future work

❖ Developing a classifier that determines the thresholds automatically and in a statistical way.
❖ Validating our approach with other traces.
❖ Using a combination of our metrics and other effective metrics to increase accuracy in terms of anomaly detection and attacks identification.
References

Q & A

Cảm ơn

THANK YOU

감사합니다
Communication pattern between hosts

- Can be represented as graph
- Communication graphs for anomalous traffic
  - Some of them are difficult to detect with conventional methods
    - Conventional methods: monitoring entropies in number of flows, etc

More difficult to detect

POSTECH Master Thesis Defense 29/26
### Table 2. Performance of the Graph-based method using $K_{\text{max}}$ and $d_{K-2}$ distance metric on Monday, Week5 traffic.

<table>
<thead>
<tr>
<th>Method</th>
<th>Total instances</th>
<th>Attacking instances</th>
<th>DR</th>
<th>FPR</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed method</td>
<td>1320</td>
<td>122</td>
<td>100 %</td>
<td>1.25 %</td>
<td>98.86 %</td>
</tr>
<tr>
<td>Wavelet-based method</td>
<td>1320</td>
<td>122</td>
<td>99%</td>
<td>56.97%</td>
<td>53.30%</td>
</tr>
</tbody>
</table>
PROCEDURE Match(s)
    INPUT: an intermediate state s; the initial state \( s_0 \) has \( M(s_0) = \emptyset \)
    OUTPUT: the mappings between the two graphs

    IF \( M(s) \) covers all the nodes of \( G_2 \) THEN
        OUTPUT \( M(s) \)
    ELSE
        Compute the set \( P(s) \) of the pairs candidate for inclusion in \( M(s) \)
        FOREACH \( p \) in \( P(s) \)
            IF the feasibility rules succeed for the inclusion of \( p \) in \( M(s) \) THEN
                Compute the state \( s' \) obtained by adding \( p \) to \( M(s) \)
                CALL Match(s')
            END IF
        END FOREACH
        Restore data structures
    END IF
END PROCEDURE Match

Source: P. Figgia
Considering two graphs Q and G, the (sub)graph isomorphism from Q to G is expressed as the set of pairs \((n,m)\) (with \(n \in G_1\), with \(m \in G_2\))
VF2 Algorithm

**Idea:** How to find candidate pair sets for a intermediate state?

Finding the (sub)graph isomorphism between Q and G is a sequence of state transition.

![Graph](image)

<table>
<thead>
<tr>
<th>Intermediate States</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
</tr>
<tr>
<td>(2,2)</td>
</tr>
<tr>
<td>s2</td>
</tr>
<tr>
<td>(2,2) (1,1)</td>
</tr>
<tr>
<td>s3</td>
</tr>
<tr>
<td>(2,2)(1,1)(3,3)</td>
</tr>
</tbody>
</table>
Let $s$ to be an intermediate state. Actually, $s$ denotes a partial mapping from $Q$ to $G$, namely, a mapping from a subgraph of $Q$ to a subgraph of $G$. These two subgraphs are denoted as $Q(s)$ and $G(s)$.

All neighbor vertices to $Q(s)$ in graph $Q$ are denoted as $N_Q(s)$, and all neighbor vertices to $G(s)$ in graph $G$ are denoted as $N_G(s)$. Candidate pair sets are a subset of $N_Q(s) \times N_G(s)$.

Assume that a pair $(n,m) \in N_Q(s) \times N_G(s)$. 
VF2 Algorithm

Candidate Pair Sets

(2, 2)  (1, 1)  (1, 4)
(3, 3)  (3,3)
PROCEDURE Match(s)
    INPUT: an intermediate state s; the initial state $s_0$ has $M(s_0)=\emptyset$
    OUTPUT: the mappings between the two graphs

    IF $M(s)$ covers all the nodes of $G_2$ THEN
        OUTPUT $M(s)$
    ELSE
        Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$
        FOREACH $p$ in $P(s)$
            IF the feasibility rules succeed for the inclusion of $p$ in $M(s)$ THEN
                Compute the state $s'$ obtained by adding $p$ to $M(s)$
                CALL Match($s'$)
                END IF
            END FOREACH
        Restore data structures
    END IF
END PROCEDURE Match
Drawing TDG

Drawing Network Traffic Graph?

18 Mar 06 17:06:39 18 Mar 06 17:06:39 udp 203.78.162.181.123 -> 203.78.171.149.123 1 0 76 0 UNK
18 Mar 06 17:06:39 18 Mar 06 17:06:39 udp 244.35.221.199.4710 -> 203.78.168.48.1026 1 0 553 0 UNK
18 Mar 06 17:06:39 18 Mar 06 17:06:39 udp 244.35.221.199.4967 -> 203.78.168.48.1027 1 0 553 0 UNK
18 Mar 06 17:06:39 18 Mar 06 17:06:49 udp 214.173.213.249.7001 -> 203.78.25.52.7003 8 0 612 0 UNK
18 Mar 06 17:06:39 18 Mar 06 17:06:49 udp 214.173.74.211.7001 -> 203.78.25.52.7003 8 0 612 0 UNK
18 Mar 06 17:06:39 18 Mar 06 17:06:48 udp 203.78.233.44.63775 -> 121.0.0.3.427 4 0 347 0 UNK

```
digraph SampleNetflowGraph {
    "203.78.162.181.123" -> "203.78.171.149.123",
    "244.35.221.199.4710" -> "203.78.168.48.1026",
    "244.35.221.199.4967" -> "203.78.168.48.1027",
    "214.173.213.249.7001" -> "203.78.25.52.7003",
    "214.173.74.211.7001" -> "203.78.25.52.7003",
    "203.78.233.44.63775" -> "121.0.0.3.427",
}
```
Figure 4: DDoS Attack Taxonomy

DDoS Attack

- Bandwidth Depletion
  - Flood Attack
    - UDP
    - Random Port Attack
    - Spoof Source IP Address?
  - ICMP Attack
    - Same Port Attack
    - Spoof Source IP Address?
- Amplification Attack
  - Smurf Attack
    - Spoof Source IP Address?
  - Fraggle Attack
    - Direct Attack
    - Loop Attack
- Resource Depletion
  - Protocol Exploit Attack
    - TCP SYN Attack
    - Spoof Source IP Address?
  - PUSH + ACK Attack
    - Spoof Source IP Address?
  - IP Address Attack
    - Spoof Source IP Address?
  - IP Packet Options Attack
    - Spoof Source IP Address?
Attack Templates

❖ Pattern Specification

digraph DDosPattern {
  victim [pos="6.681, 3.708"]; attacker0 [pos="0.736, 3.708"]; attacker1 [pos="3.389, 2.514"]; attacker2 [pos="4.931, 0.667"]; attacker3 [pos="7.292, 0.264"]; attacker4 [pos="9.361, 1.458"]; attacker5 [pos="10.181, 3.708"]; attacker6 [pos="9.361, 5.958"]; attacker7 [pos="7.292, 7.153"]; attacker8 [pos="4.931, 6.750"]; attacker9 [pos="3.389, 4.903"]; attacker0 -> victim [time=0, protocol="ICMP", info="Echo Request"]; attacker1 -> victim [time=0, protocol="ICMP", info="Echo Request"]; attacker2 -> victim [time=0, protocol="ICMP", info="Echo Request"]; attacker3 -> victim [time=0, protocol="ICMP", info="Echo Request"]; attacker4 -> victim [time=0, protocol="ICMP", info="Echo Request"]; attacker5 -> victim [time=1, protocol="ICMP", info="Echo Request"]; attacker6 -> victim [time=1, protocol="ICMP", info="Echo Request"]; attacker7 -> victim [time=1, protocol="ICMP", info="Echo Request"]; attacker8 -> victim [time=1, protocol="ICMP", info="Echo Request"]; attacker9 -> victim [time=1, protocol="ICMP", info="Echo Request"];}

DDoS Pattern
Attack Templates (1/3)

Figure 3: Host Scanning

Figure 4: Port Scanning
Figure 5: Smurf Attack

Figure 6: Fraggle Attack
Attack Templates (3/3)

Figure 7: DNS Amplification Attack

Figure 8: ICMP Flood Attack
Thresholds of POSTECH network

- **TCP**
  - Kmax: 5525
  - dK-2 distance: 11328

- **UDP**
  - Kmax: 15327
  - dK-2 distance: 23608

- **ICMP**
  - Kmax: 1425
  - dK2: 2996
Figure 3: Schematic representation of possible connections from hosts in the MiniPop versus other hosts.
Validation (DARPA dataset)

❖ **DARPA 1999 Dataset**
- Week 1 and week 3: no attack (for training data).
- Week 2: 43 attacks belonging to 18 labeled attack types are used for system development.
- Week 4 and week 5: 201 attacks belonging to 58 attack types (including 40 new attacks).

❖ **The traffic data on Monday, Week 5 of DARPA Dataset**
- Including 122 attack instances.
- Attacks that change communication structure in network graph:
  - Smurf, apache2, udpstorm, portsweep and etc.
Validation

- We use standard measurements such as detection rate (DR), false positive rate (FPR) and overall classification rates (CR) to evaluate our approach.
  - True Positive (TP): The number of anomalous instances that are correctly identified.
  - True Negative (TN): The number of legitimate instances that are correctly classified.
  - False Positive (FP): The number of instances that were incorrectly identified as anomalies, however in fact they are legitimate activities.
  - False Negative (FN): The number of instances that were incorrectly classified as legitimate activities however in fact they are anomalous.
  - \[ DR = \frac{TP}{TP + FN} \]
  - \[ FPR = \frac{FP}{TN + FP} \]
  - \[ CR = \frac{(TP + TN)}{(TP + TN + FP + FN)} \]
Peacomm

- **Connect to Overnet**
  - The bot publishes itself on the Overnet network and connects to peers. The initial list of peers is hard coded in the bot.

- **Download Secondary Injection URL**
  - The bot uses hard coded keys to search for and download a value on the Overnet network. The value is an encrypted URL that points to the location of a secondary injection executable.

- **Decrypt Secondary Injection URL**
  - The bot uses a hard coded key to decrypt the downloaded value, which is a URL.

- **Download Secondary Injection**
  - The bot downloads the secondary injection from a web server using the decrypted URL.

- **Execute Secondary Injection**
  - The bot executes the secondary injection, possibly scheduling future upgrades on the peer-to-peer network or scheduling bot stat tracking at some other resource.

http://static.usenix.org/event/hotbots07/tech/full_papers/grizzard/grizzard_html/
Figure 2: Number of Remote IPv4 Addresses Contacted Over Time for Duration of Infection
Assortativity vs Disassortativity

Assume nodes have some intrinsic property which separates them into different classes.

Construct assortativity matrix $\mathbf{E}_{x,y}$: In this case $x=\{A,B\}$, $y=\{A,B\}$ representing the fraction of links between nodes of type $(x,y)$, with the condition $\sum_{x,y} \mathbf{E}_{x,y} = 1$.

$$\mathbf{E} = \begin{pmatrix} 4/9 & 6/9 \\ 6/9 & 2/9 \end{pmatrix} \times \frac{1}{2} = \begin{pmatrix} 2/9 & 3/9 \\ 3/9 & 1/9 \end{pmatrix}$$

For undirected graphs count twice and divide by 2.
Coefficient of assortativity

\[ \Delta \varepsilon = \text{sum diagonal terms} - \text{sum off-diagonal terms} \]

Example: \[ \Delta \varepsilon = (2/9 + 1/9) - (3/9 + 3/9) = -1/3 \]

What does it mean?

Consider a case in which links occur completely at random between A and B:

What is \( \Delta \varepsilon \) in this case? (Undirected graphs)

Examples:

\[ \varepsilon = \begin{pmatrix} a & b \\ b & a \end{pmatrix} \]

Then \[ \Delta \varepsilon = (2a) - (2b) \] (Norm: \( 2a + 2b = 1 \))

(For \( a = b \) then \( \Delta \varepsilon = 0 \))

In general:

\[ \varepsilon = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \]

Then \[ \Delta \varepsilon = (a+c) - (2b) \] (Norm: \( a + c + 2b = 1 \))
Coefficient of assortativity

\[ \Delta \varepsilon = \text{sum diagonal terms} - \text{sum off-diagonal terms} \]

\[ \Delta \varepsilon > 0 \quad \Rightarrow \quad \text{Like nodes tend to be more likely linked} \]

\[ \Delta \varepsilon < 0 \quad \Rightarrow \quad \text{Unlike nodes tend to be more likely linked} \]

Values of assortativity different than zero mean possible presence of correlations (>0) or anticorrelations (<0) in the Network.

Nodes can be classified according to their degree
Examples for node-node degree assortativity

Rings:

\[ N = 7 \quad L = 7 \]

\[ \Delta \eta = 1 \quad \text{because} \quad \eta_{2,2} = 1 \quad \text{and zero otherwise} \]

What is the effect of shortcuts:

\[
\eta = \begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 3/10 & 1/10 \\
3 & 0 & 0 & 3/10 & 3/10 \\
4 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

\[ \Delta \eta = (3/10) - (4/10 + 3/10) = -4/10 = -0.4 \]
Graph Metrics on TDGs – dK-2 distance

- **Structure analysis - dK-n series: n=1,2,3,…**
  - Look at inter-dependencies among topology characteristics
  - dK-n series are degree correlations within simple connected graphs of size n

- Some examples:
  - dK-1 describes node degree distribution
  - dK-2 describes joint node degree distribution
  - dK-3 captures clustering coefficient

Source: Ben Zhao (June 22, 2011)
Napster - Queries

1. Peers register with central server, send list

2. Peers send queries to central server which has content index of all files

3. File transfers happen directly between peers

Last point is common to all P2P file sharing networks and is their main strength as it allows them to scale well!

Source: Th. Strufe, VL Peer-to-Peer Networks, 2011
Gnutella (2nd generation)

Gnutella – How it Works

Overlay network

Join

- To join, peer needs address of one member, learn others
- Queries are sent to neighbors
- Neighbors forward queries to their neighbors (flooding)
- Replies routed back via query path to querying peer

Source: Th. Strufe, VL Peer-to-Peer Networks, 2011
KaZaA (3rd generation)

KaZaA – Hierarchy

- Ordinary nodes belong to one Supernode
  - Can change SN, but not common
- Supernodes exchange information between themselves
  - Supernodes do not form a complete mesh

Source: Th. Strufe, VL Peer-to-Peer Networks, 2011
KaZaA – Finding Stuff

1. Peer sends query to its own supernode

2. Supernode answers for all of its peers and forwards query to other supernodes

3. Other supernodes reply for all of their peers

Source: Th. Strufe, VL Peer-to-Peer Networks, 2011
DHT: Motivation

- Why we need DHTs?
- Searching in P2P networks is not efficient
  - Either centralized system with all its problems
  - Or distributed system with all its problems
  - Hybrid systems cannot guarantee discovery either
- Actual file transfer process in P2P network is scalable
  - File transfers directly between peers
- Searching does not scale in same way
- Original motivation for DHTs: More efficient searching and object location in P2P networks
- Put another way: Use addressing instead of searching

Other DHTs

- Many other DHTs exist too
  - Pastry, similar to Tapestry
  - Kademlia, uses XOR metric
  - Kelips, group nodes into $k$ groups, similar to KaZaA
  - Plus some others...
- Overnet P2P network (also eDonkey) uses Kademlia
  - Wide-spread deployed DHT
- All DHTs provide same API
  - In principle, DHT-layer is interchangeable
dK-2 value matrix

![dK-2 value matrix](image)

**Normal**

**Anomaly**