

Abstracting and Visualizing Host Behaviour through Graphs

Eduard Glatz

Computer Engineering and Networks Laboratory

ETH Zurich (Switzerland)

eglatz@tik.ee.ethz.ch



Motivation

- Research in behavioural host profiling
 - Dominant and new session structures/application mixes
 - Evolution of host profiles over time
- Investigation of security incidents
 - Is this IP address a server or a client?
 - What services is this IP address providing?
- Teaching
 - Explain how Berkeley sockets work
 - Show complex communication patterns

Idea

- Common tools focus on traffic as a whole
- Browsing through flow lists might be a solution - but is unattractive when lists get very long
- Summarization techniques for flow lists exist, but are specialized on anomaly detection
- Idea: *develop your own tool*

Host Behaviour seen in Traffic Data

Traffic types

- Application traffic (user-triggered)
- Basic lookup traffic (application-triggered), e.g. DNS
- Infrastructure traffic (system-triggered), e.g. DHCP

Host profiling

1. What application mixes are prevalent?
2. Which roles are incorporated by hosts? (e.g. client, server, P2P role)
3. How do these two properties depend on each other?

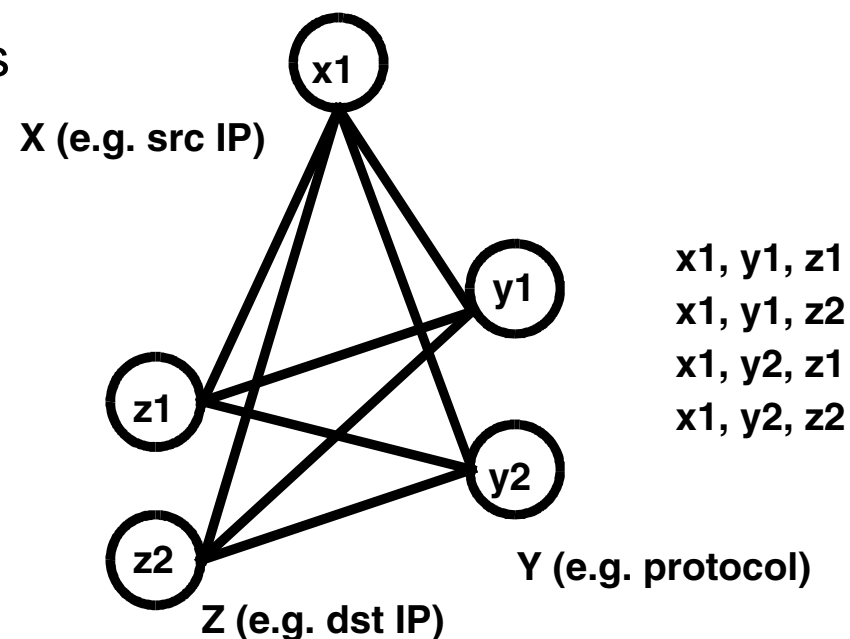
How to represent Host Traffic?

Idea: *use graphs*

- Nodes correspond to flow attributes
- Links show flow attributes that appear together
- Result: very dense/noisy graph

Problem:

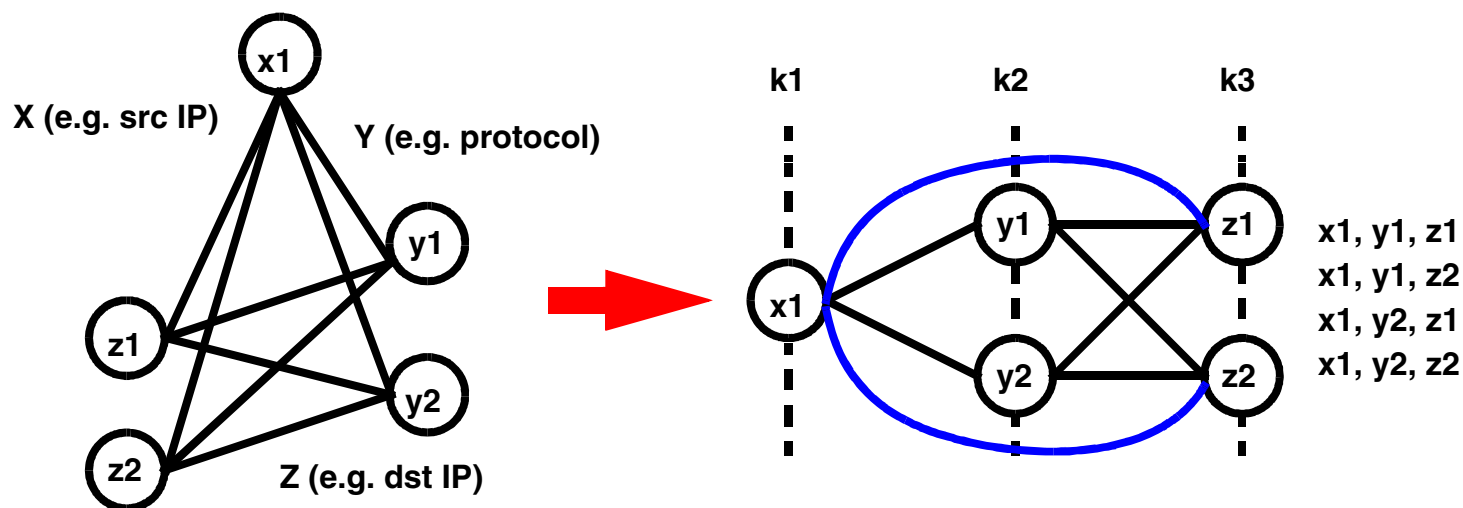
- Which relationships are most interesting to illustrate?



Transaction Visualization by k-Partite Graphs

Approach:

- K-partite graphs plus abstraction, e. g.

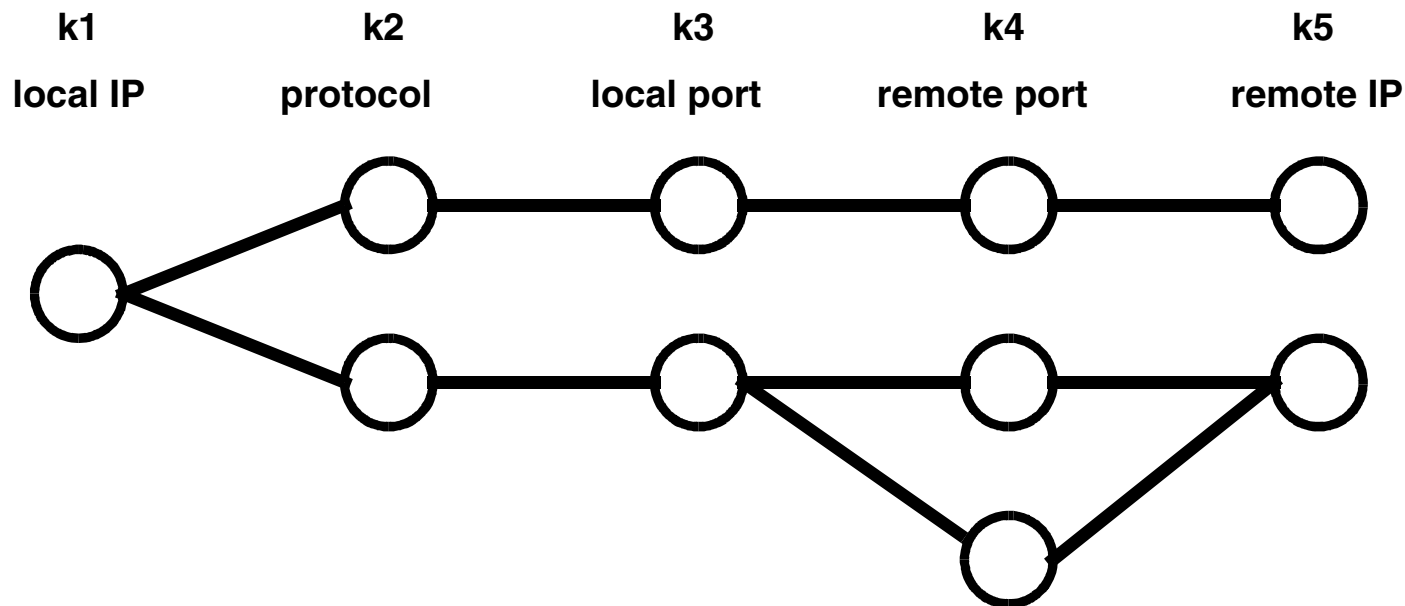


Abstraction:

- Purge blue lines and re-arrange partitions as needed to keep most interesting edges

Host Application Profile (HAP) Graphlet

We propose: Host traffic visualized by 5-partite graph



- Terminology: local/remote instead of source/destination
- Optionally annotate nodes with attribute values (not shown)

HAP Graphlet

- Visualizes BSD socket based communication in a straight forward way:
 - IP addresses assigned to first/last partition (k1, k5) show layer 3 connectivity
 - Central partitions (k2..k4) show layer 4 connectivity
 - Respects port number uniqueness (per protocol, per IP address)

Properties of HAP Graphlets

- Near-planar structure
- Shows remote IPs and ports associated with local port numbers (flows grouped per application)
- Host roles are apparent:
 - Server role
 - Client role
 - Peer role

What Graph Structures can we expect?

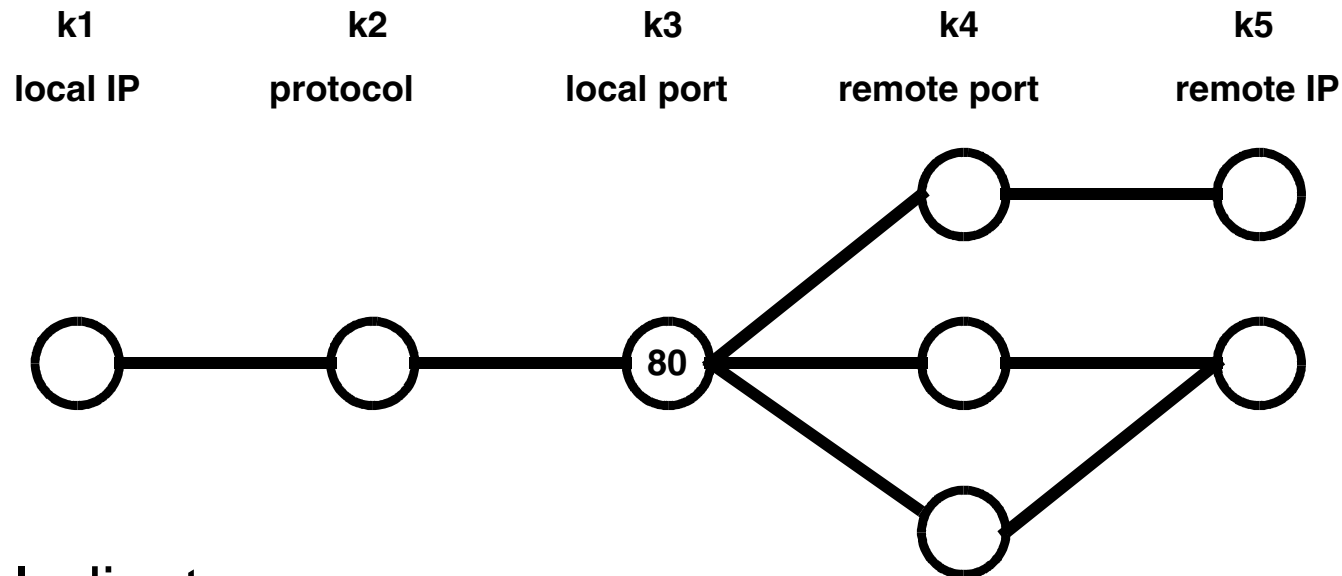
One session per application per peer:

- Client/server host roles
- P2P host roles

Complex sessions (applications) that use one or more connections:

- To handle different tasks in parallel (e. g. control and data exchange)
- To improve performance (parallel flows to same remote endpoint)

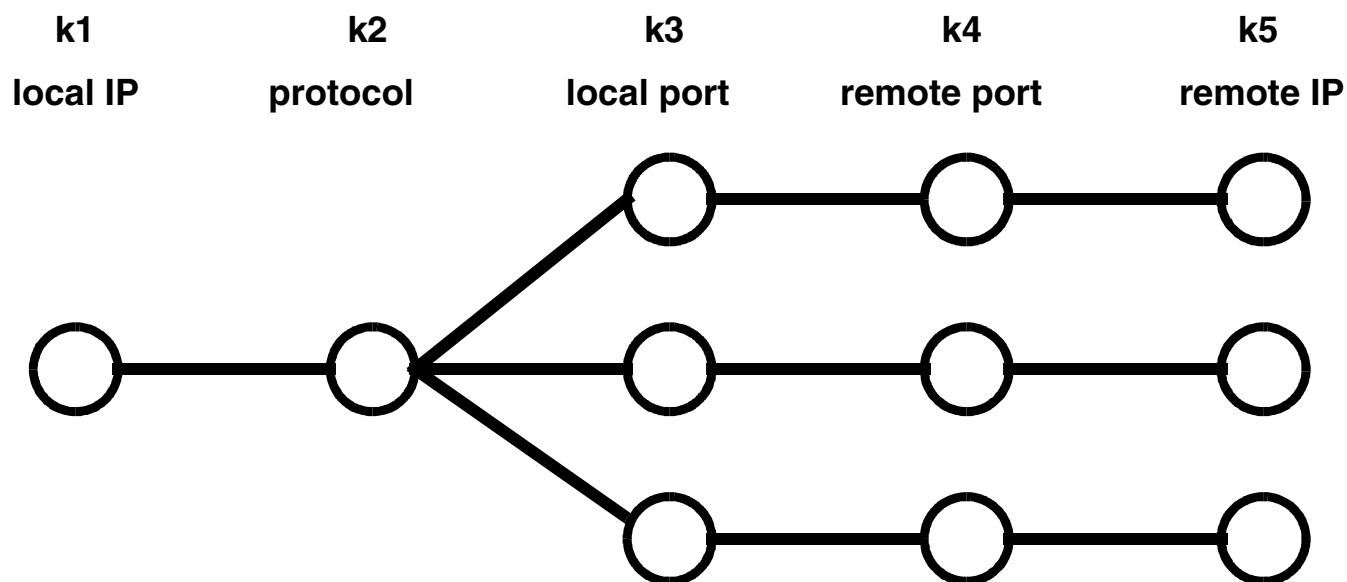
Server Role



■ Indicators:

- K3 -> k4 out-degree $d1 > 1$ (multiple remote connections)
- K3 -> k5 (virtual) out-degree $d2 > 1$ (multiple clients)
- Often: $d1 > d2$ (parallel connections)

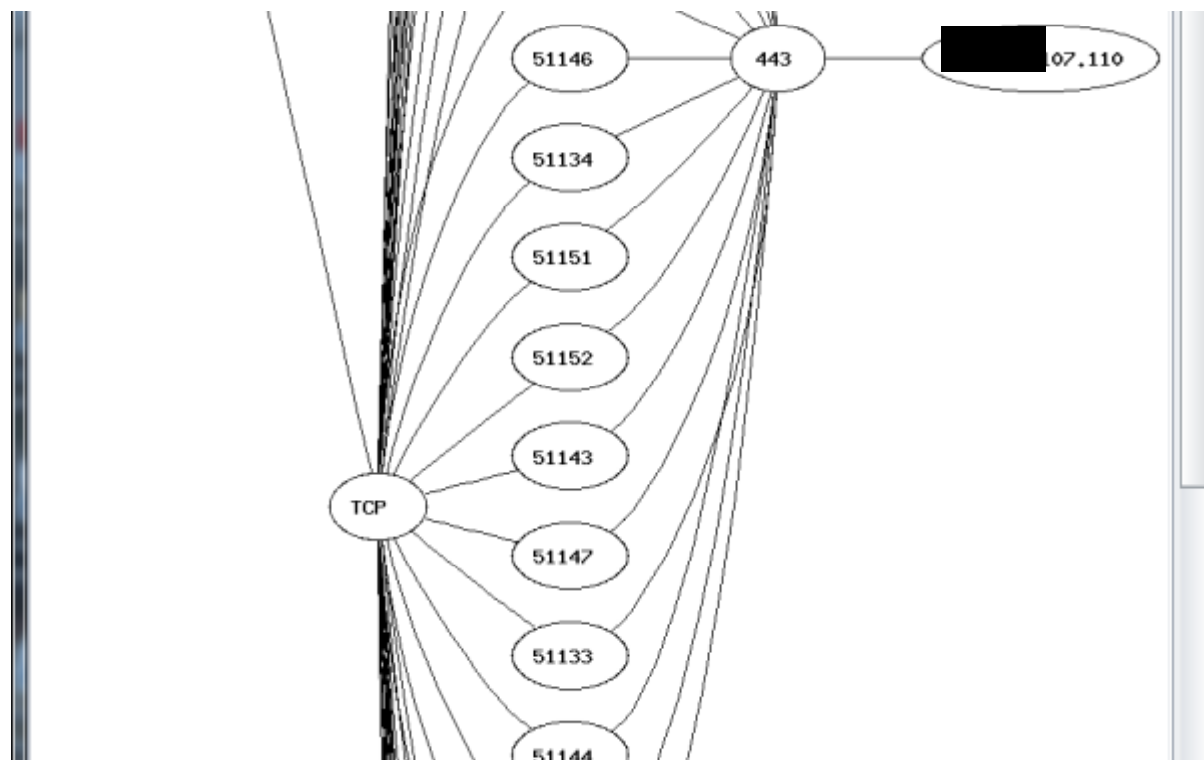
Peer-to-Peer Role



■ Indicators:

- Many remote peers connected, and involved port numbers all above 1024
- Hard to confirm: needs additional data sources

- Ideally, HAP graphlet fits into available screen area
- But ...



Role Summarization

Idea:

- Compress *per-role subgraphs*

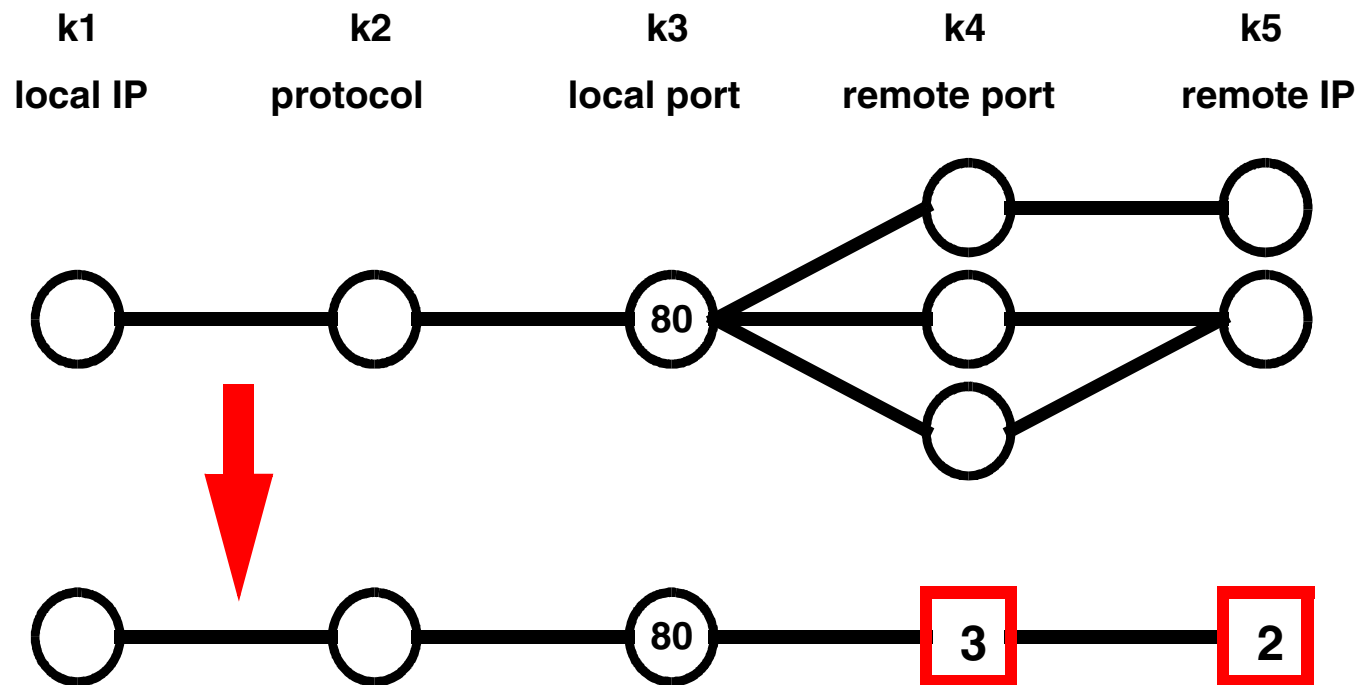
Prerequisite:

- Roles can be associated with sub-graphs

Methodology:

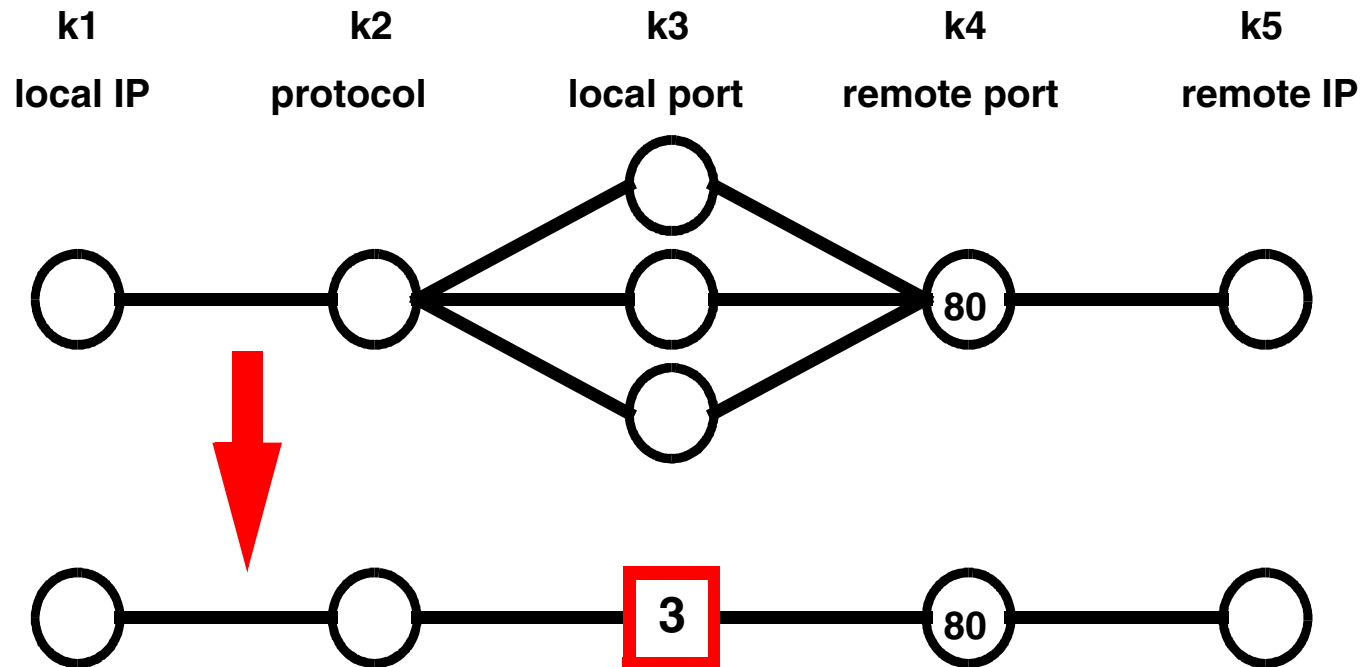
- Decompose graphlet into role-related subgraphs
- Replace such sub-graphs by summary sub-graphs
- Ignore graph partitions without role assignment
- Decomposition and replacement algorithm depends on role types (server/client/p2p roles)

Server Role Summary



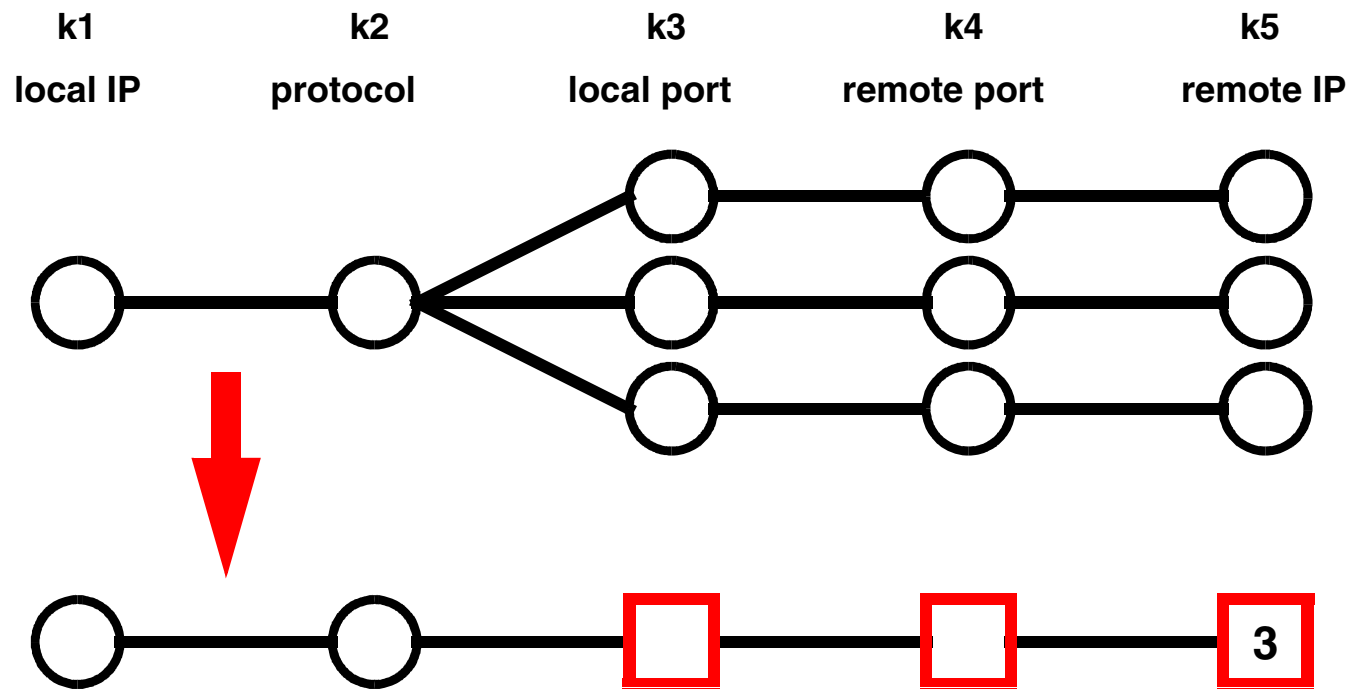
- Replace server-role related sub-graph
- **Node** annotations mark #connections and #clients

Client Role Summary



- Replace client-role related sub-graph
- **Node** annotation marks number of connections

Peer-to-Peer Role Summary

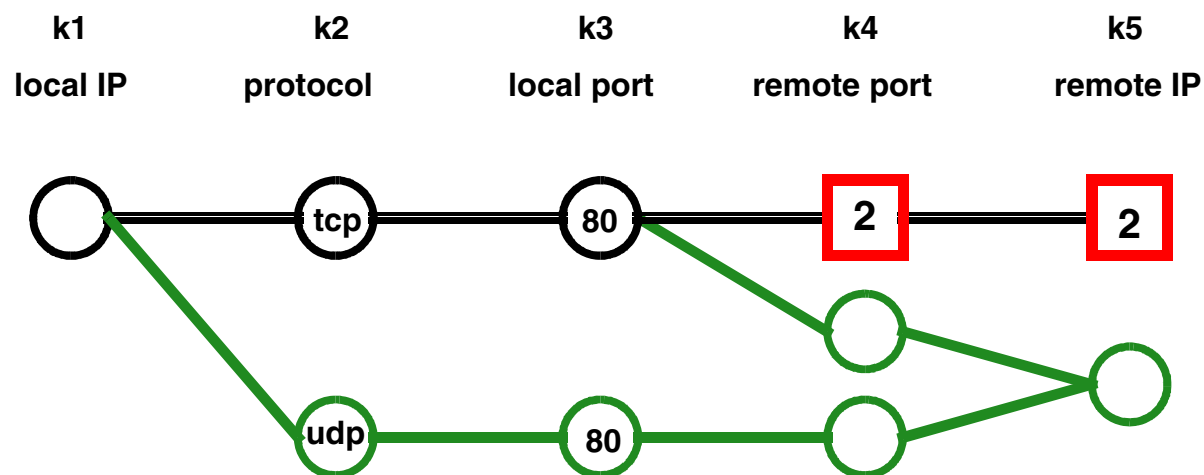


- Replace P2P-role related sub-graph
- **Node** annotation marks number of peers

Minimizing Information Loss

Scenario 1 (server role):

- One or more clients use double server connectivity through two protocols (e. g. control and data connections)
- Full summarization cannot include both connection paths



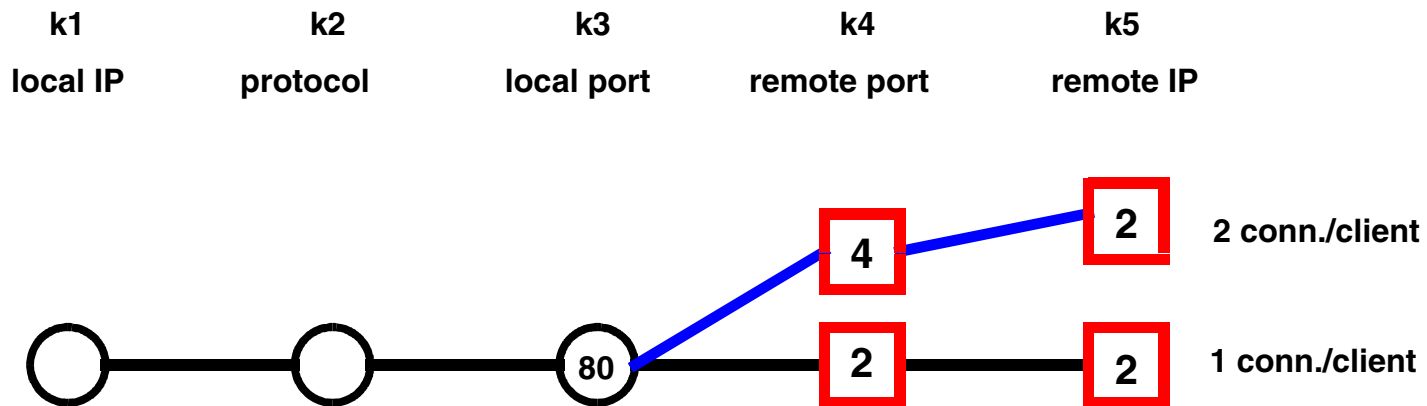
Approach:

- Do not summarize affected client(s)
- Use available screen height as a constraint

Minimizing Information Loss

Scenario 2 (server role):

- One or more clients use parallel connections to server
- Full summarization gives average parallelization degree only



Approach:

- Split summary into suitable parallelization groups
- Use available screen height as a constraint

Productive and Unproductive Traffic

Fact:

- A considerable part of Internet traffic is unproductive (e. g. scanning, misdirected flows)

But:

- We are mainly interested in productive traffic to characterize host behaviour
- When scan traffic enters the picture, then we want to identify it as such

Scan Traffic Filtering

- Mark and optionally remove scan traffic from visualization
- How to distinguish scan from productive traffic?
 - Hypothesis:
productive traffic is bidirectional,
e. g. involves bilateral interaction on the transport layer
- Methodology:
 - Pair unidirectional flows in opposite direction that use identical endpoints
 - Look „over the fence“ (i. e. observation interval borders) when searching a buddy for a within-interval unidirectional flow

Evaluation of Filtering Approach

- We inspected two dark IP ranges maintained by our ISP for scan traffic monitoring (over ~2400 h)
 - Captured at least traffic observed by network telescopes
 - Our advantage: we can do it over all address ranges
- Limitations:
 - NTP: in symmetric mode NTP source sends periodically unacknowledged messages to peers subscribed (RFC 958)
 - Multicast Source Discovery Protocol (MSDP) : works unidirectional
 - Discard protocol (tcp port 9)
 - Situations of multi-connected applications which run over different interfaces (and one connection is unidirectional)

The Tool: HAPviewer

- Unix/Linux application with graphical user interface
- Typical use cases:
 - Qualitative studies of roles incorporated by hosts
 - Investigations of complex connection structures (e.g. of P2P applications)
 - Identifying unknown service ports
 - Evaluations of scan traffic
 - Teaching of Berkeley socket model

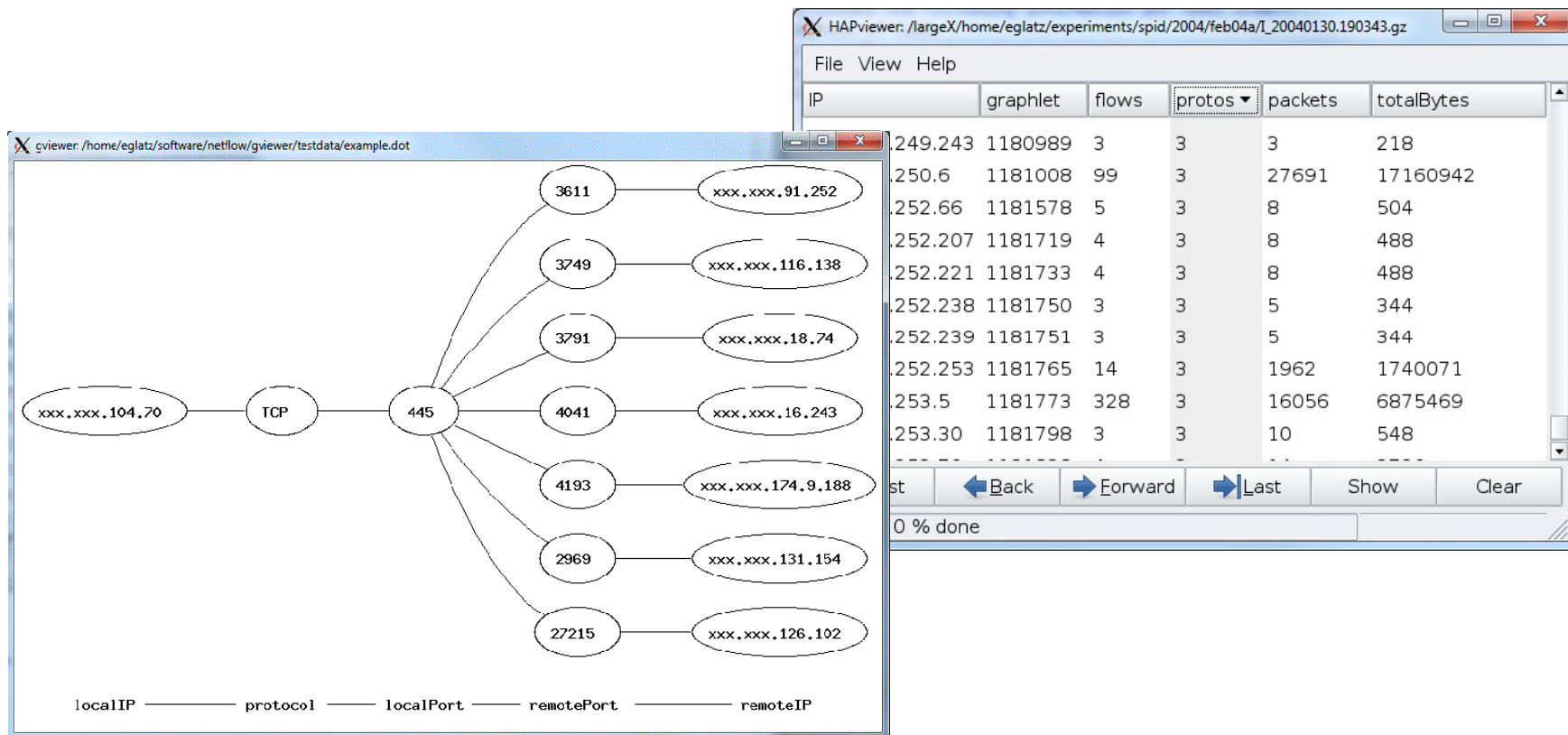
Tool Architecture and Technologies

- Technology overview:
 - Unix/Linux based
 - C++
 - Gtk over gtkmm (C++ wrapper)
 - Graphviz
 - Pcap++
- Availability:
 - Available under dual licence GPL/BSD

Contributions

- Graph-based host traffic visualization
 - Illustrates used protocols and the functional role of local and remote port numbers
- Techniques for filtering unwanted traffic
 - Identification of scan traffic or misdirected flows
 - Prerequisite for proper service port identification
- Open-source visualization tool
 - Processes efficiently several millions of flows
 - Provides techniques for summarizing our graph-based profiles
 - Options for manipulating profiles and displaying graphs

Questions?



Tool will be demonstrated