Relationship Discovery with NetFlow to Enable Business-Driven IT Management

Andreas Kind
IBM Zurich Research Laboratory
ank@zurich.ibm.com

Dieter Gantenbein
IBM Zurich Research Laboratory
dga@zurich.ibm.com

Hiroaki Etoh
IBM Tokyo Research Laboratory
etoh@jp.ibm.com

***Abstract*— The understanding of relationships and dependencies between business processes and the underlying IT infrastructure is important for enabling business-driven IT management. This work uses the widespread NetFlow feature to derive direct and indirect traffic relationships in IT infrastructures. We define an algorithm for relationship discovery with NetFlow and describe the application of the discovery approach in a large production environment.**

I. Introduction

Today, IT service teams are increasingly faced with complex distributed IT infrastructures. The infrastructures became complex because of rapid growth and piecemeal extension in order to support new or changed business-level processes and value chains. They may also have become complex through mergers and acquisitions—or simply by aging, given today's accelerated technology pace.

Modern e-business environments tightly link customer and supplier systems with the internal computing infrastructure. Hence the performance of the end-to-end business processes becomes critically dependent on the availability of the underlying computing infrastructure. Detecting IT assets and understanding their properties, relationships, and roles in business processes can be key for operational efficiency and business process optimization (see Figure 1).

IT infrastructures are no longer just business-critical in the sense that failures and suboptimal performance negatively impact business. In fact, business innovation itself can be driven bottom-up through dedicated provisioning and use of IT resources. However, IT infrastructure transition requires good knowledge about the status quo of an enterprise's complex IT environment. Only based on such knowledge can sustainable transition scenarios be defined and implemented to reduce operational expenditures, increase revenue, and innovate business processes.

Tracking computing devices as assets, including usage, helps in the provisioning and maintenance of efficient and optimized services. A precise understanding of the operational infrastructure and its users also plays a key role in the negotiation of outsourcing contracts and for planning mergers and acquisitions. Building an accurate inventory of computing assets is especially difficult in unknown heterogeneous systems and in networking environments without prior device instrumentation.

Classical methods for asset and inventory management quickly reach their limit in today's dynamic environments: Periodic physical inventories ("wall-to-wall") have the clear advantage of identifying the actual location of the devices, but require costly human visits and can detect neither mobile, currently out-of-office equipment nor the existence and use of contained assets. Financial asset tracking, while being an accepted process in its own right, cannot detect additional equipment brought in or remotely accessing the resources of an organization. Periodic self-assessment questionnaires to be filled out by individual end users or their cost-center managers are another, often complementary approach. Apart from the human effort they require and the inaccurate incomplete data that results, most forms pose questions that could easily be answered out of the infrastructure itself.

Well-managed computing infrastructures typically equip servers and end-user devices with software agents for the tracking of resources and the system as well as for application-performance monitoring. There are many situations, however, in which this cannot be assumed and used. In many organizations, there are a fair number of heterogeneous devices that are brought in ad-hoc and are not instrumented accordingly, for which instrumentation is not available, or on which instrumentation has been disabled. After a merger/acquisition, for example, we can hardly assume to find an encompassing management environment in place across the entire holding organization. However, a good understanding of the infrastructure provided and its users is essential, actually already prior to the acquisition or while negotiating an outsourcing contract.

The focus of this work is on extending the normal IT asset discovery function with relationship discovery. The motivation for this extension is that understanding the relationships and dependencies between IT assets as well as between IT assets and business processes is important for enabling business-driven IT management [1], [2]. Particularly, knowing the dependencies of business processes on specific IT components enables the optimization of IT infrastructures along business-level objectives. A full automated business mapping is difficult due to the many layers between real business requirements

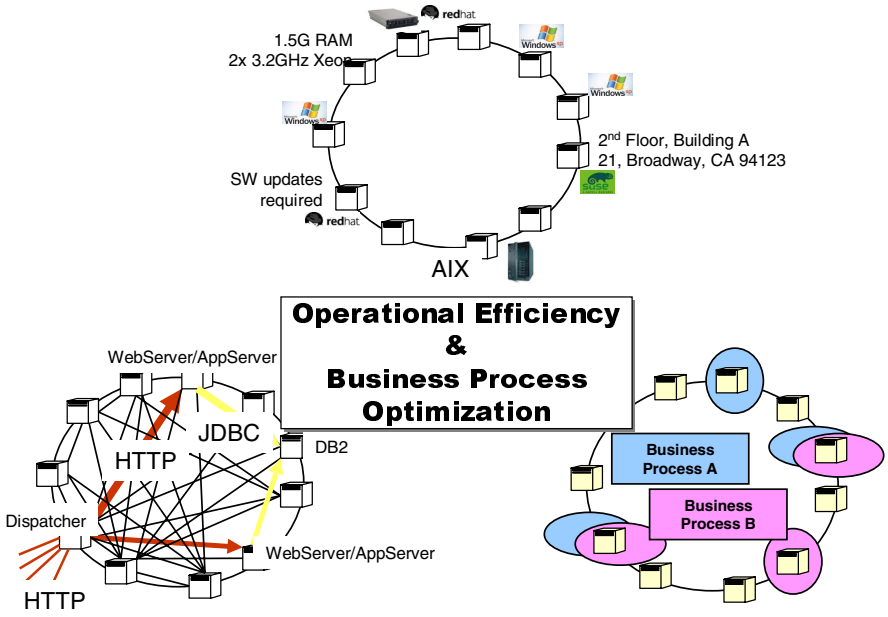

Fig. 1. Detecting IT asset properties (top), relationships (left), and business roles (right) for operational efficiency and business process optimization.

and the underlying infrastructure. This work is aimed at reducing the manual business decomposition process as much as possible.

The following four scenarios illustrate the benefit of relationship information for business-driven IT management in the IT services domain:

1) Problem statement:
   Identify opportunities for application co-location on single servers.
   Business objective:
   More accurate transition plans reduce actual transition costs. Smaller number of servers. Reduced future operation costs.
   Action based on relationship information:
   Suggest co-location of applications onto single server according to service dependencies and acceptable server loads.
2) Problem statement:
   Identify opportunities for server co-location in central data centers.
   Business objective:
   More accurate transition plans reduce actual transition costs. Reduction of total expenses by running fewer data centers.
   Action based on relationship information:
   Suggest relocation of servers to central data centers. Suggest closing smaller data centers.
3) Problem statement:
   Find optimal staging of server relocations.
   Business objective:
   Cut relocation costs.
   Action based on relationship information:
   Schedule relocations for minimal business impact.
4) Problem statement:
   Reduce complexity of IT infrastructure.
   Business objective:
   Gain flexibility of business model through flexibility in IT infrastructure (on demand credo). Increase revenue through ability to quickly adapt to changing business needs.
   Action based on relationship information:
   Decouple physical and logical components of different business processes and value chains.

In summary, relationship information can help bottom-up to understand the impact of IT-level changes on business-level processes. Furthermore, relationship information can show top-down the required IT-level changes of planned business-level decisions.

The remainder of the document is structured as follows. The next section describes our approach to relationship discovery based on NetFlow. We discuss direct as well as indirect traffic relationships and provide a framework and algorithm for traffic relationship discovery with confidence values. Section III describes the implementation of the approach in the Aurora network profiling system. In Section IV, the application of the relationship discovery approach in a large production environment is presented. The paper finishes with conclusions in Section V.

II. Relationship Discovery

Networked IT assets and their relationships can be discovered actively and passively. To actively discover network assets, several different techniques can be employed. They all share the principle that the network needs to be explored exhaustively from a starting point by using a repetitive algorithm that walks the entire network up to an endpoint

or until the entire IP address range has been exhausted. We distinguish between walking the network topology with the Simple Network Management Protocol (SNMP), while discovering attached devices and their usage, UDP- and TCP-based host-stack and services scanning, and detailed protocol-specific scanning, e.g. for Windows Remote Procedure Calls (RPC) and Server Message Block (SMB), as well as HTTP, SSH, FTP, and Telnet.

Passive network mapping enables the discovery and identification of network assets in a purely passive fashion, i.e. without generating any kind of traffic that stimulates target machines in order to discover their presence. Network packet sniffing is an approach that interfaces directly with the raw traffic, whereas higher-level subscriptions to network-traffic flows become readily available from Remote Monitoring (RMON) and NetFlow meters. NetFlow [3] currently is the most widespread protocol for traffic profiling. To our knowledge it has not been used for advanced IT asset relationship discovery.

Almost every router and many network switches can be configured to export information about traffic flows with the NetFlow protocol. The widespread availability of NetFlow makes this protocol highly useful for IT asset and relationship discovery. Basically, a NetFlow record maps a flow given by a tuple containing the source/destination IP addresses, the protocol, and the source/destination ports to another tuple containing the start/end timestamps and packet as well as octet volumes:

$$(src, dst, proto, src\_port, dst\_port, \ldots) \rightarrow (start, end, \#pkts, \#octs, \ldots)$$

A key advantage of a discovery method based on NetFlow is that, apart from configuration, no changes to existing networking architectures are required, no credentials are needed, and privacy is not affected as packet payload information is not provided with NetFlow[1].

Already the raw flow records contain basic traffic relationship information. Source and destination have a direct traffic relationship with respect to the protocol and port information. Before we introduce the notion of indirect traffic relationships, a formal definition of direct traffic relationships is provided in the next section.

A. Direct Traffic Relationships

We distinguish between *flows* and *flow events*. A flow event is semantically a NetFlow record, i.e., a flow associated with time and traffic volume information. We define a flow f as

$$f = (src, dst, proto, srv, rcvd) \text{ in } F \, .$$

The src and dst fields are given by IP addresses, $proto$ is the transport or link-level protocol (e.g., TCP, UDP, ICMP), srv denotes the TCP/UDP service port, and the $rcvd$ flag determines whether the flow is received from the server. The flag is used to indicate whether dst or src is the receiver of the unidirectional flow. The set of all flows is denoted as F. A flow event is then defined as

$$e = (f, t_s, t_e, octs, pkts) \text{ in } E \, . \quad (1)$$

The start and end timestamps of a flow event are given as t_s and t_e. The set of all flow events is denoted by E, and the set of all flow events of a given flow f is defined as

$$E(f) = \{e \text{ in } E \mid f_e = f\} \, .$$

We assume that the service port srv can be mapped to an application. Typically, more than one service port can be mapped into the same application. For instance, the various mail ports, such as *smtp* (25), *pop3* (110), *imap* (143), etc., are mapped into the *MAIL* application. In future, the application will be determined by the NetFlow exporter by deep packet analysis. The latter aspect is particularly important with applications using unregistered or dynamic ports.

The resulting direct relationship information from ongoing NetFlow analysis already reveals the role of most identities discovered. Hosts are identified as servers when clients access a particular service on this machine. When analysis runs for extended time periods, it can also be discovered whether clients use a secondary server for load-balancing or fail-over reasons.

A continuative question is how servers and services depend on each other and how they support an entire business application, such as a customer relationship management (CRM) system or a financial accounting system. Such dependency information can only be identified by detecting indirect traffic relationships which are discussed in the next section.

B. Indirect Traffic Relationships

The basic idea of detecting indirect traffic relationships is to identify flow correlations, i.e., flow pairs (or even flow chains) that occur significantly more often than other flow pairs (or flow chains). A flow pair is likely to occur more often if the service related to one of the flows in the flow pair requires the service determined in the other flow in the flow pair. An example for such a flow pair is a flow representing a database access and a flow representing a directory access for authentication. The reason for the correlation of these two flow might be that a user has to authenticate with the directory server prior to database access.

We now present an algorithm for indirect traffic relationship discovery for flow pairs. The detection of flow chains could, in principle, be derived from this algorithm but is not further considered in this work. First, we assume flow predicate functions

$$f_p : F \times F \rightarrow \{1, 0\}$$

[1]This may no longer be true for the emerging IP Flow Information Export (IPFIX) [4] standard.

to compare two flows. For example, the following flow predicate function returns 1 if (i) the destination of the first flow is identical with the source of the second flow and (ii) the source of the first flow is not the destination of the second flow (i.e., not simply a reply).

$$f_p(f_1, f_2) = \begin{cases} 1 & \text{if} \quad dst_1 = src_2 \quad \wedge \\ & \quad\;\; src_1 \neq dst_2 \\ 0 & \text{otherwise} \end{cases}$$

with

$$f_1 = (src_1, dst_1, proto_1, srv_1, rcvd_1) \quad \text{and}$$
$$f_2 = (src_2, dst_2, proto_2, srv_2, rcvd_2)\,.$$

We can now define the set of all flow-event pairs P for any two flows f_1, f_2 in F with $f_1 \neq f_2$.

$$\begin{aligned} P(f_1, f_2) = \{(e_1, e_2) \mid \quad & 0 \leq t_s(e_2) - t_s(e_1) < t_{\max} \wedge \\ & f_p(f(e_1), f(e_2)) = 1 \\ & \\ & \text{with} \quad e_1 \text{ in } E(f_1) \quad \text{and} \\ & \qquad\;\; e_2 \text{ in } E(f_2) \\ & \} \end{aligned}$$

As shown in Figure 2, two flow events are considered a flow-event pair if their starting times differ only within $t_{\max}$. A good value for $t_{\max}$ is 10 s. In the example, the total number of flow events $|E|$ is 4 and the number of flow-event pairs $|P(f_1, f_2)|$ is 2.

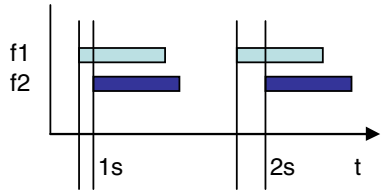

Fig. 2. Flow pair example with two flow-event pairs.

If the starting times of flow events are too far apart, these flow events are not identified as a flow-event pair (see Figure 3). In a variation of this scheme, also the difference of the flow end times or the time difference between the end of the first flow and and start of the second flow may be used. Using the above definition for flow-event pairs and a time-dependent correlation distribution, we can now specify a flow correlation function $c()$ with (e_1, e_2) in the set of flow-event pairs given by $P(f_1, f_2)$:

$$c(f_1, f_2) = \frac{\sum D(t_s(e_2) - t_s(e1))}{|P(f_1, f_2)|} \tag{2}$$

The distribution D takes into account that the likelihood that two flows are correlated depends on the offset of the start times up to $t_{\max}$. We use a simple distribution table

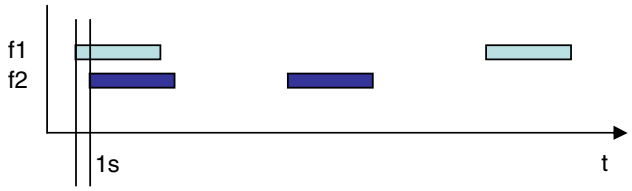

Fig. 3. Flow pair example with one flow-event pair.

$$\begin{aligned} D(0) &= 1.00 \\ D(1) &= 0.99 \\ D(2) &= 0.98 \\ &\ldots \\ D(9) &= 0.01 \end{aligned}$$

to approximate a distribution function as shown in Figure 4.

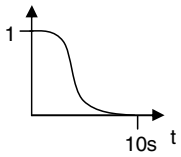

Fig. 4. Time-dependent correlation distribution.

The sum of distributions in the denominator of Equation (2) is normalized by the number of flow-event pairs (f_1, f_2). If all flow-event pairs (f_1, f_2) start with the same start times (i.e., $D(0) = 1.0$), the correlation value of $c(f_1, f_2)$ will yield 1. The correlation value in the example in Figure 2 yields $c(f_1, f_2) = 1.7/2 = 0.85$; the correlation value in the example in Figure 3 yields $c(f_1, f_2) = 0.9/1 = 0.9$. The correlation value for the latter example is higher because in this case the second occurrences of the flow events are not close enough to be considered a flow-event pair. This example shows that the correlation value has to be combined with a correlation confidence value. The confidence in the second example should be less than in the first example because the calculation of the correlation value is based on fewer flow-event pairs, thus there is less evidence despite the higher correlation value.

C. Correlation Confidence

The purpose of the confidence value is to put the correlation value into perspective based on two considerations:

- The more flow-event pairs, the higher the confidence that the correlation value is correct.
- The higher the number of flow-event pairs relative to the number of all flow events, the higher the confidence that the correlation value is correct.

The latter consideration is to be added to address situations in which the total number of flow events is very high and a large number of flow event pairs are wrongly identified. The function $c'()$ addresses these considerations:

$$c'(f_1, f_2) = (1 - 1/(|P(f_1, f_2)| + 1)) * |P(f_1, f_2)|/|E|$$

The function is defined as a product of two factors addressing the two considerations above. If we use to the two examples in Figures 2 and 3, the confidence values yield $0.66 * 1 = 0.66$ for the first and $0.5 * 0.5 = 0.25$ for the second example. Thus the confidence value for the second example is much lower than that for the first example. The combination of correlation values and confidence value is a useful metric for indirect relationship detection.

D. Flow Correlation vs. Service Correlation

The correlation algorithm as defined in the preceding sections detects flow correlations. In certain situations, however, it may be interesting to detect service correlations rather than flow correlations. Service correlation means that a multitude of flows from different sources are treated as one identical flow as long as they are related to the same service and destined to the same target server.

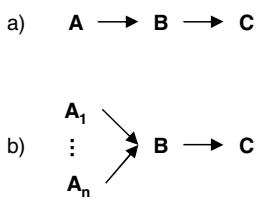

Fig. 5. Flow correlation a) and service correlation b).

Figure 5 shows the difference between flow and service correlation. In a), flows $A \rightarrow B$ and $B \rightarrow C$ are identified as correlated if $A \rightarrow B$ and $B \rightarrow C$ start often enough about at the same time. In b), flows $A_1 \rightarrow B$ to $A_n \rightarrow B$ are treated as events of the same flow if they are related to the same service. Flows $A_i \rightarrow B$ and $B \rightarrow C$ are, therefore, identified as correlated if any flow $A_i \rightarrow B$ and $B \rightarrow C$ start often enough about at the same time ($0 \leq i \leq n$). Service correlation can be detected using the algorithm in Sections II-B and II-C by ignoring the source address of f_1 in the flow correlation and confidence function.

III. Implementation

The correlation algorithm for relationship detection was implemented in the Aurora network profiling system [5], [6]. Aurora is a flow-based network profiling tool for network monitoring in high-speed networks. It is based on novel techniques for collecting, storing, and analyzing NetFlow and IP Flow Information Export (IPFIX) records.

The implementation of the correlation and confidence functions operates incrementally on a series of 12 five minute NetFlow files (i.e., covering one hour) along the following steps:

1) Parse NetFlow records and sort flow events by start times.
2) Walk through sorted flow events with time window of $t_{\max}$ and update correlation values of flow pairs which occur as flow event pairs in the time window.
3) Compute confidence values for all flow pairs identified.
4) Sort identified flow pairs by $c * c'$.
5) Generate output with top flow correlations in the Extensible Markup Language (XML).
6) Merge last 24 hourly correlation files into a daily correlation file; merge last 30 daily correlation files into a monthly correlation file.

Here is an excerpt of the XML output generated:

```
<?xml version="1.0" ?>
<flow-correlation>
  ...
  <flows ids="753" total="5301">
    <counters>
      <packets>314730</packets>
      <octets>39323000</octets>
    </counters>
    <flow id="-581" proto="6" appl="23"
          service="873" tos="0x00">
      <endpoint type="src" address="*"
                port="873" domain="K Bld" />
      <endpoint type="dst" address="9.4.4.138"
                port="0" domain="AIX Srv" />
      <counters>
        <flows>1</flows>
        <packets>7</packets>
        <octets>542</octets>
      </counters>
    </flow>
    ...
    <flow id="2" proto="6" appl="16"
          service="730" tos="0x00">
      <endpoint type="src" address="9.4.20.44"
                port="730" domain="K Bld" />
      <endpoint type="dst" address="9.4.9.135"
                port="0" domain="AIX Srv" />
      <counters>
        <flows>1</flows>
        <packets>7</packets>
        <octets>542</octets>
      </counters>
    </flow>
    ...
  </flows>
  <correlations type="undirected" total="14">
    <correlation srcid="-9" dstid="2"
                 value="71" confidence="36"/>
    <correlation srcid="182" dstid="2"
                 value="33" confidence="54"/>
    ...
  </correlations>
  <correlations type="directed" total="22">
    <correlation srcid="25" dstid="176"
                 value="88" confidence="34"/>
    <correlation srcid="-11" dstid="47"
                 value="27" confidence="14"/>
    ...
  </correlations>
</flow-correlation>
```

The correlation file is structured into two sections. The first section defines flows which are referred to from correlations in the second section. Flows with negative flow ids denote aggregated flows. These flows have a wildcard as source address and are used for service correlations as described in Section II-D.

The example contains directed and undirected correlations. The flow pair in Figure 2 is an example for a directed correlation. Flow f_2 clearly occurs after f_1. However, time resolution limits may lead to flow pairs with identical start times. In these cases, it is not possible to determine if f_1 depends on f_2 or *vice versa*. Such correlations are denoted as undirected.

Our current effort is on combining the relationship information obtained via Aurora with host and application information from other discovery systems. The objective is to enrich the relationship information with more precise information regarding devices (e.g., operating systems, software versions, physical locations) [7] and business transactions (e.g., SQL database queries) [8]. Also part of the effort is to define and implement the reconciliation process for strong relationship information into a configuration management database (CMDB).

IV. Application

The Aurora system was recently tested in a CRM production environment. Using hundreds of servers, the environment serves over 30,000 clients. A key aspect of the test was to detect relationships within such a large IT infrastructure. It was possible to identify direct traffic relationships and dependencies between many key servers in the CRM infrastructure, including backup servers, SQL databases, directory servers (LDAP), middleware servers (MQSeries) as well as application and web servers. Figure 6 visualizes a small part of the relationships discovered. Each node is a (potentially virtual) host with unique IP address. The services discovered are shown with names and port numbers next to the host names. For example, a host with *ssh (22)* denotes a secure shell server. The nodes are colored according to the segments they belong to. The edges between nodes provide information about the intensity of the direct traffic relationships. Indirect relationships are shown in other visualizations generated by Aurora.

The information collected was used to build a service decomposition of the CRM infrastructure, which (if updated continuously) is more accurate and complete as well as cheaper to provide than manually created decompositions. It establishes the relationships and hierarchy between the CRM business processes, the application components, and the underlying hardware. It can help identify the hardware, software and process elements that will be affected by a change to the infrastructure. For example, if a change is to be made to a given server and requires some downtime, the support team will be able to determine all the applications and processes that will be affected during this downtime. This will enable a better planning of the change and allow all affected areas to be notified ahead of time.

The CRM production environment in which we conducted the tests is segmented into firewalled subnets. Servers in each subnet are connected by switches which are not NetFlow-capable. We used the switched port analyzer (SPAN) feature to mirror VLAN traffic to spare Gbit ports. A NetFlow/IPFIX meter (part of Aurora) was connected to the spare ports to generate NetFlow records for the relationship detection component in Aurora. The meter accesses packet-header information to maintain a flow table. Payload information is not accessed.

The generated NetFlow records provided enough relationship information although at certain times the entire VLAN volume at a switch did not fit into a single mirror port. There was no impact on the switch operation during these periods. Traffic at four high-end switches were monitored to cover about 90% of the CRM traffic. We expect to see NetFlow and IPFIX on more high-end switching equipment in the future, so that the SPAN feature will not have to be used any more. A deployment of the relationship discovery approach will then be again simpler.

The tests also revealed current limitations with the relationship discovery approach. Important for correctly associating traffic flows with applications and business processes is the mapping of transport layer ports to TCP/UDP service names. In general, ports are used according to the IANA port assignments [9]. However, some applications (especially in the web services context) use unpublished or dynamically assigned ports. Although the Aurora system has heuristics to cover some of these cases (e.g., passive FTP, VoIP calls), other applications can only be correctly assigned by protocol parsing or configuration. Vendors have announced that application detection based on protocol parsing will be added in routing and switching equipment with NetFlow v9 and IPFIX in the near future.

Another limitation of the approach comes with the fact that in web services environments based on Java Enterprise Edition (J2EE) many TCP connections stay open and are not established with every client transaction request. In these cases, the timestamps in NetFlow cannot be used for flow correlation. However, when NetFlow exporters are configured with short idle and export periods, volume peaks or silent periods can be considered for correlation detection in the future. An alternative is also to focus on dependency detection during the boot process of the infrastructure or certain parts of it.

V. Conclusions

Understanding the relationships and dependencies between IT assets as well as between IT assets and business processes is key for enabling business-driven IT management. Particularly, knowledge of the dependencies of business processes from specific IT components opens the door to a variety of business-driven IT management strategies. We presented a relationship discovery approach based on NetFlow that proved advantageous for enabling business-driven IT management when tested in a large production CRM environment.

A further conclusion from the application of the relationship discovery approach is that passive discovery techniques have clear advantages in larger data centers which are strictly partitioned into firewalled network segments. Servers in protected segments cannot be discovered with other methods, such as active scanning.

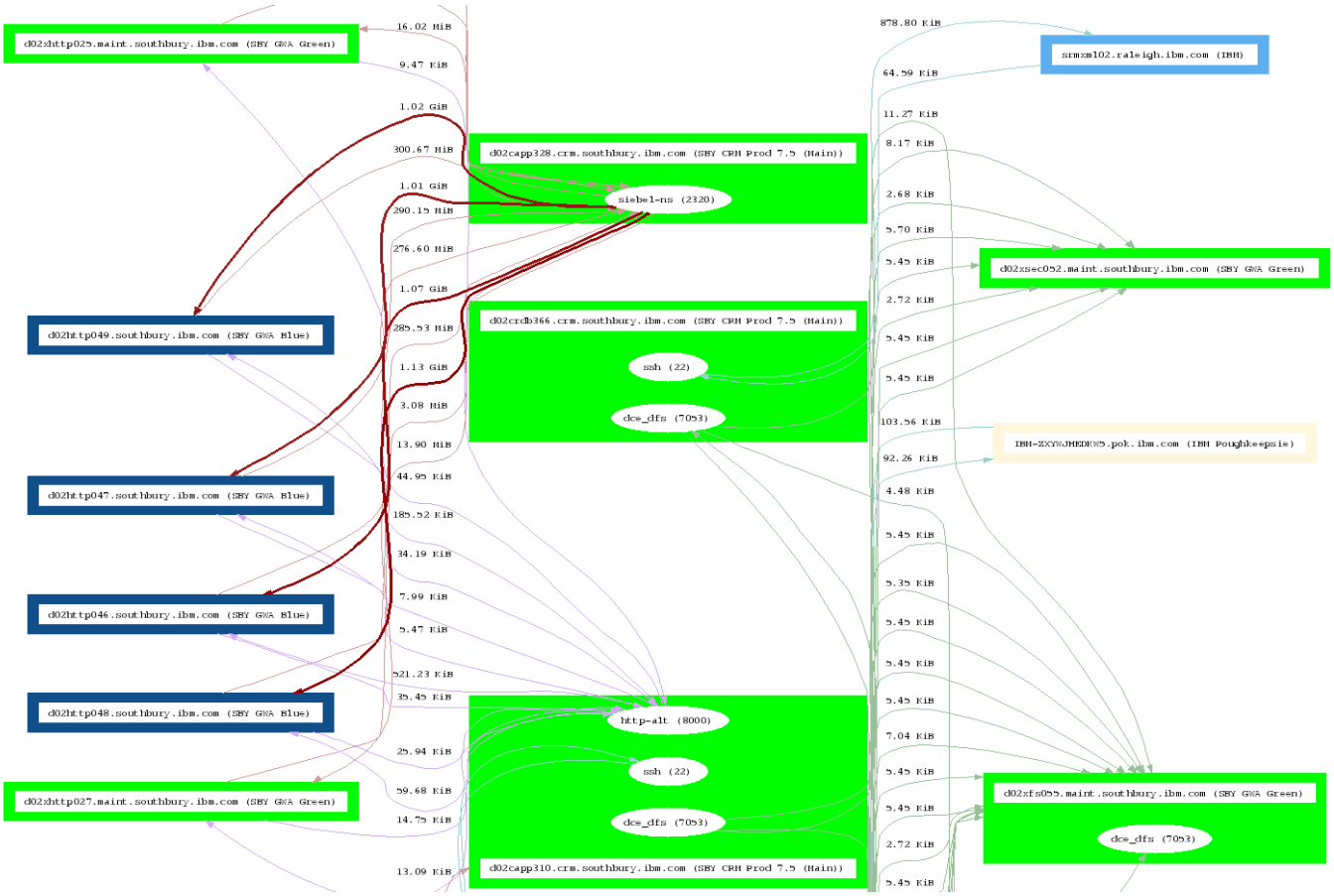

Fig. 6. Relationships in CRM environment.

ACKNOWLEDGMENT

The authors would like to thank Susan Meade, Peggy Kline, David Nayir, and Scott Gronbach for their support in testing the approach in the CRM environment.

REFERENCES

[1] Vijay Machiraju, Claudio Bartolini, and Fabio Casati, "Technologies for business-driven IT management," in *Extending Web Services Technologies: The Use of Multi-Agent Approaches*, Larence Cavedon, Zakaria Maamar, David Martin, and Boualem Benatallah, Eds. Kluwer Academic Publishers, Mar. 2004.

[2] Claudio Bartolini, Mathias Sallé, and David Trastour, "IT service management driven by business objectives – An application to incident management," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, Apr. 2006

.

[3] Cisco Systems, "Cisco IOS NetFlow, Introduction," http://www.cisco.com/warp/public/732/Tech/nmp/netflow, 2005.

[4] Benoit Claise (Ed.), "IPFIX Protocol Specifications," http://www.ietf.org/html.charters/ipfix-charter.html, IETF, Sept. 2005.

[5] IBM Research, "AURORA – A Flow-Based Network Profiling System," http://www.zurich.ibm.com/aurora, 2005.

[6] Andreas Kind, Paul Hurley, and Jeroen Massar, "A Light-Weight and Scalable Network Profiling System," *ERCIM News*, , no. 60, pp. 67–68, Jan. 2005.

[7] Dieter Gantenbein, "Tracking Assets and Vulnerabilities in Corporate Networks," *ERCIM News*, vol. 56, Jan. 2004.

[8] Hisashi Kashima, Tadashi Tsumura, Tsuyoshi Idé, Takahide Nogayama, Ryo Hirade, Hiroaki Etoh, and Takeshi Fukuda, "Network-Based Problem Detection for Distributed Systems," in *ICDE*, 2005, pp. 978–989.

[9] IANA, "Internet Assigned Port Numbers," http://www.iana.org/assignments/port-numbers, 2005.