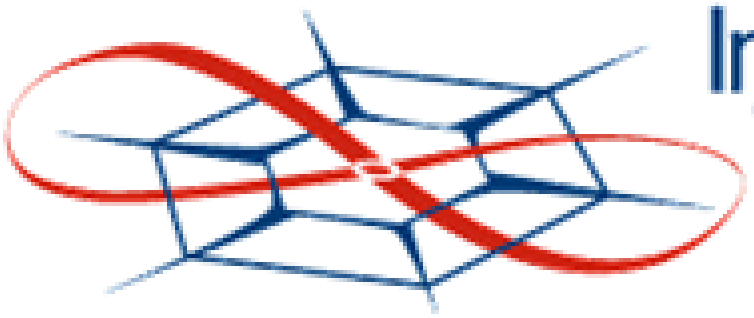


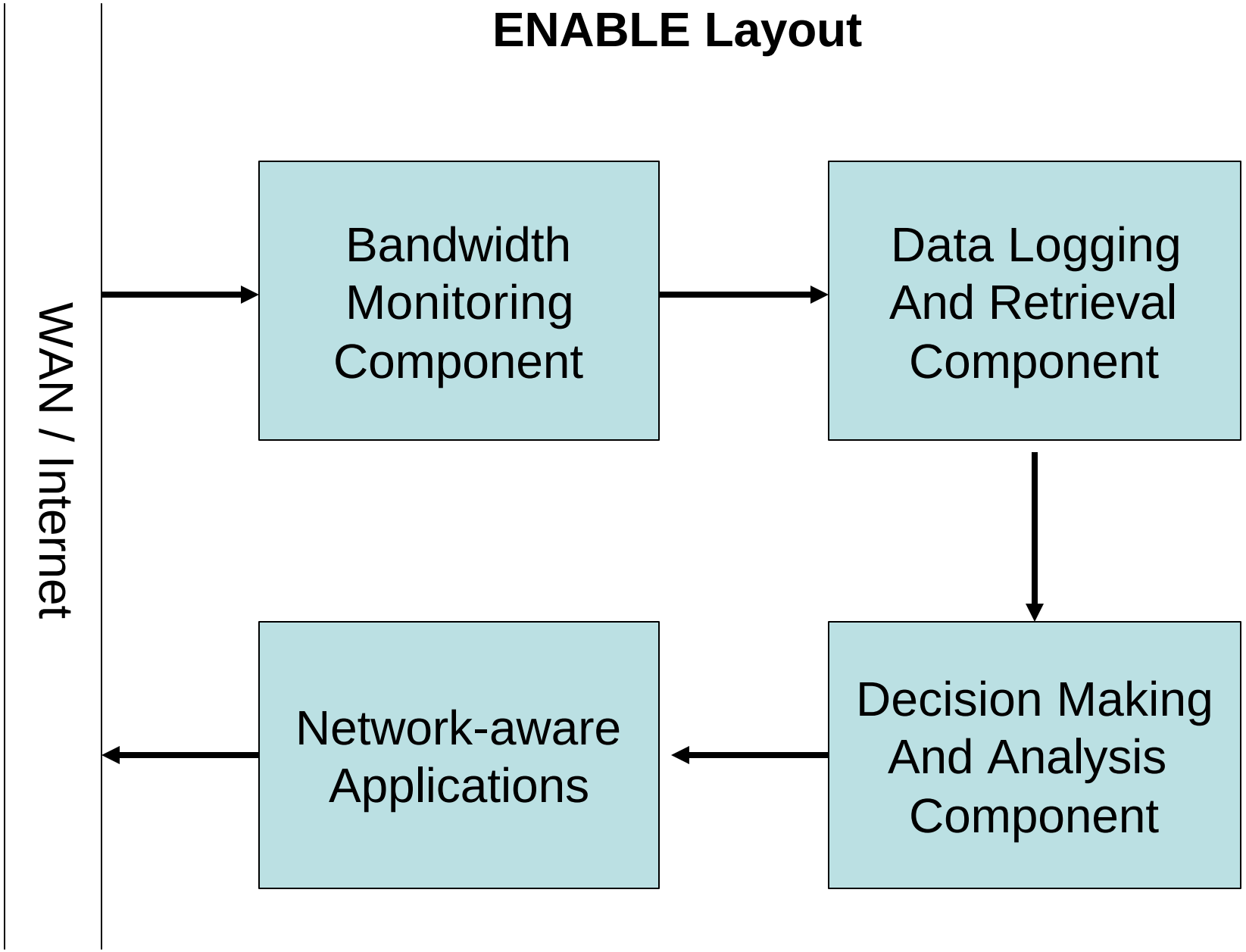
Information and Telecommunication Technology Center

The Development of a FTP Bandwidth Monitoring Tool for the ENABLE Project

A Project Funded by DOE

What is ENABLE?

- Enhancing of Network-aware Applications and Bottleneck Elimination
- Provide services to guarantee maximum efficient use of network resources
- Provide current network characteristics to Network-aware applications in order to effectively adept to current network conditions


ENABLE Layout
WAN / Internet
Bandwidth Monitoring Component
Data Logging And Retrieval Component
Decision Making And Analysis Component
Network-aware Applications

Bandwidth Monitoring Component

- Composed of 3 Components:
  - S.N.O.R.T. IDS
  - MySQL database
  - Perl Script
- Why do we choose these components?

S.N.O.R.T. Capabilities

- Perform real-time traffic analysis
- Perform packet logging
- Perform protocol analysis
- Perform content searching/matching
- Uses flexible rules language
- Uses modular plug-in architecture
- Real-time alert
- MySQL support

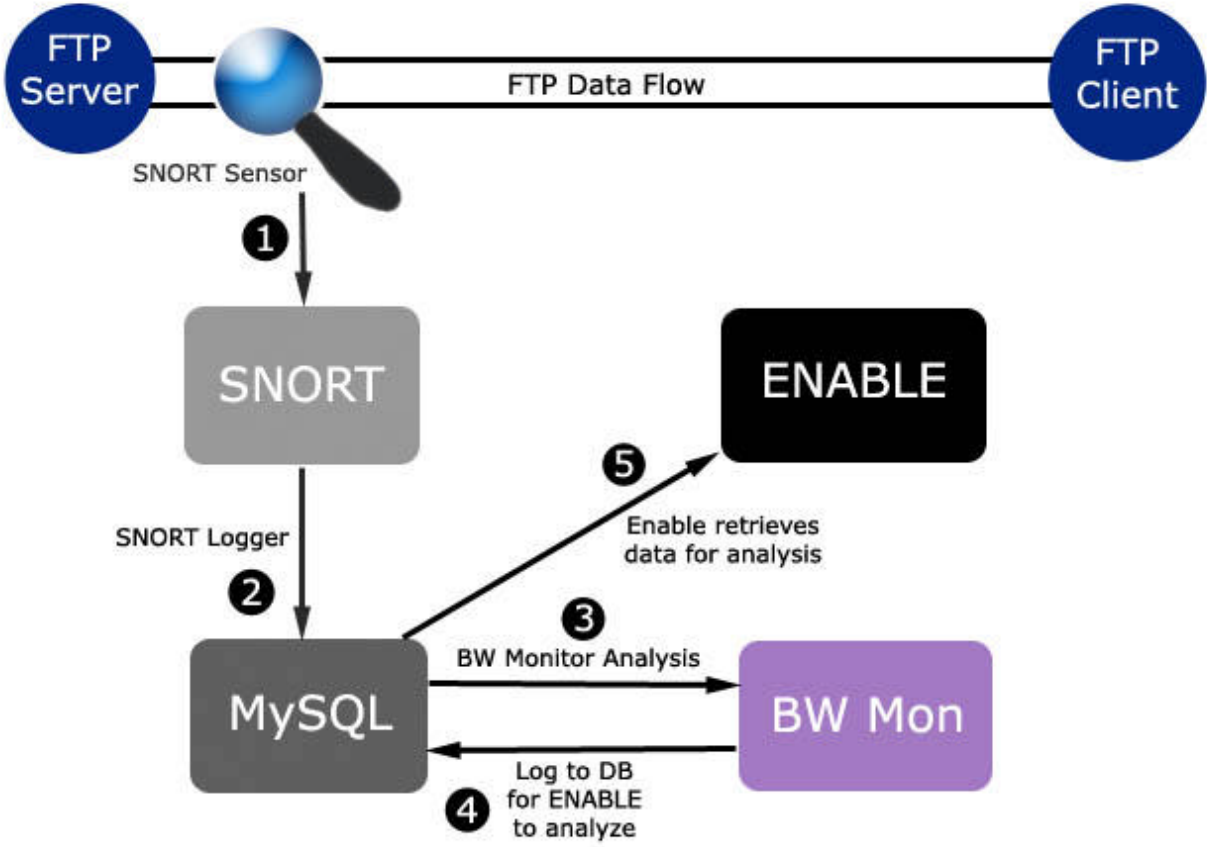
MySQL database

- Fast performance
- Easy to customize
- Operation stability
- Easy deployment
- Easy data management

PERL

- High portability
- Operation Stability
- Easy Availability
- Modular support for MySQL

Bandwidth Monitor Layout



TCP/IP headers

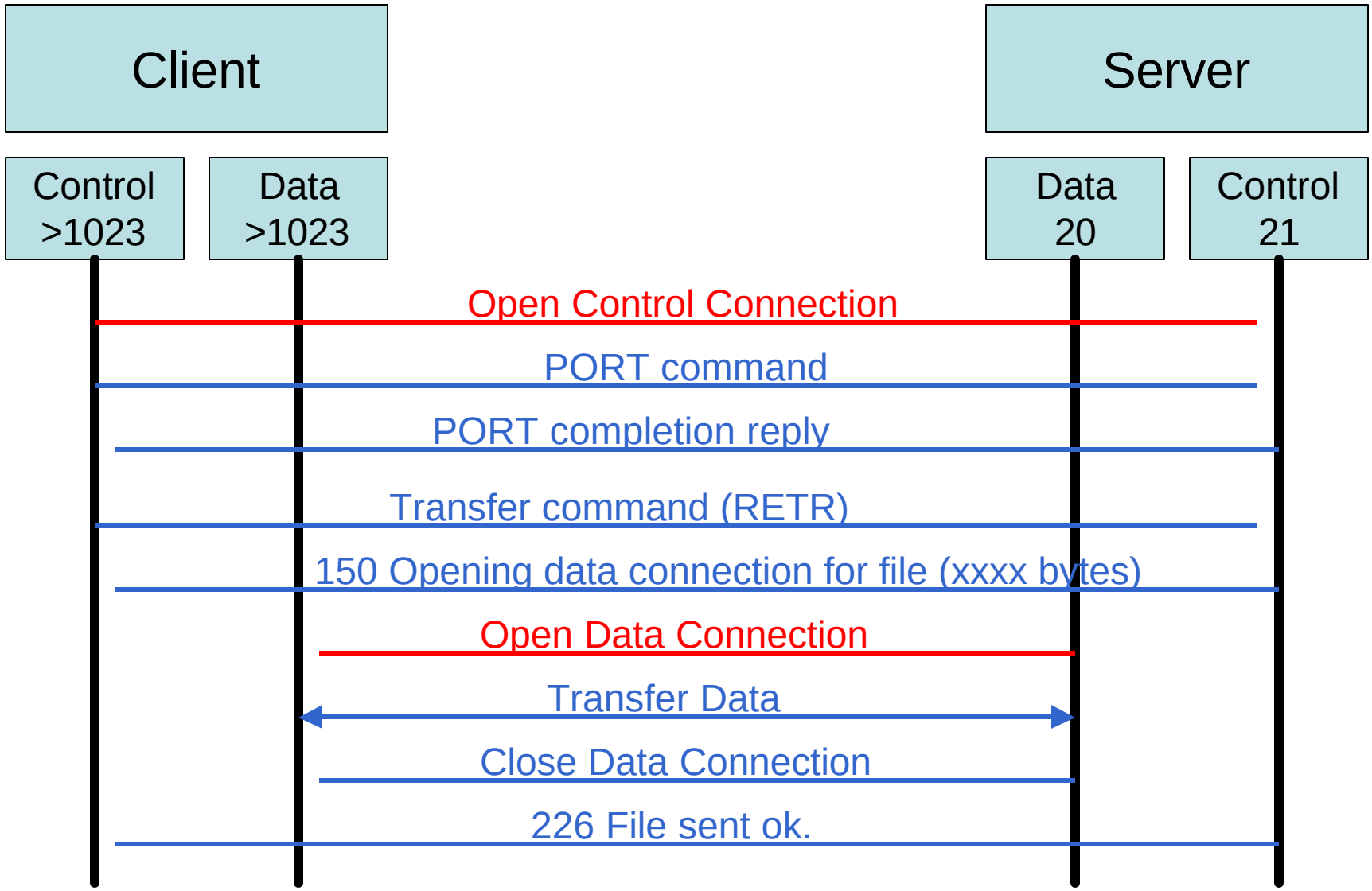
- In order to parse TCP/IP data stream correctly, we need to filter the following fields from the tcp stream:
  - Source and Destination IP address
  - Source and Destination Port numbers
  - TCP Flags
  - Sequence numbers and Acknowledgement numbers

FTP Protocol

- We need to understand how FTP works before we can analyze and parse its stream

  - FTP Control Session (Port 21)
  - FTP Data Session (Port 20)
  - FTP Modes (Active Mode)

FTP Protocol



Theoretical Architecture

- This architecture is the initial / ideal architecture
- This architecture is NOT implemented due to problems with pcap libraries
- This architecture consists of these components:
  - S.N.O.R.T.
  - MySQL database
  - Perl Script (using Net::RawIP module)

Theoretical Architecture

- S.N.O.R.T. functions as an IDS, with rules set up to detect and log FTP transactions
- Perl script utilizes Net::RawIP (based on pcaplib) to monitor FTP transactions in promiscuous mode
- All analyzed and calculated data will be logged to ENABLE database

Theoretical Architecture

- How it works:
  - S.N.O.R.T. will detect incoming FTP logins and log alerts into MySQL database. (SYN on port 21)
  - The PERL script will be run at a timely interval (i.e. 5 minutes) to parse the database for new connections.
  - Once new connections are detected, it will continue to monitor the FTP transactions using pcaplib. (SYN on port 20)
  - Data transferred will be calculated based on the initial sequence number, and the terminating sequence number. (FIN on port 20)
  - Packet losses can also be calculated based on repeated sequence numbers.

Implemented Architecture

- This architecture is a lot more dependent on S.N.O.R.T.
- Contains the same components as the Theoretical Architecture:
  - S.N.O.R.T.
  - MySQL database
  - Perl Script

Implementation Details : Layout

FTP Client

Router

FTP Server

Port 1234	FTP User Login Request	Port 21
Port 1234	FTP User Login Accepted	Port 21
Port 1234	FTP Data Request: RETR or STOR	Port 21
Port 3456	FTP Data Initiation: SYN	Port 20
Port 1234	150 Opening data connection for filename (1234 bytes)	Port 21
	⋮	
Port 3456	FTP Data Completion: FIN	Port 20
Port 1234	226 File Sent ok.	Port 21

Implementation Details: S.N.O.R.T.

- S.N.O.R.T. functions as an IDS, with rules set up to detect FTP transactions
- New rule types are created to detect FTP transfer contents
- The file size transferred can be obtained in the detection by using these rule types
- The bandwidth consumed can be estimated

Implementation Details: S.N.O.R.T.

- Creating new ruletypes in S.N.O.R.T.
  - This ruletype classifies the FTP transactions we will detect as alerts, and logs them to MySQL database

```
ruletype BWMon
{
  type alert
  output database: alert, mysql, user=snort password=snp3rt dbname=snort
    host=localhost
}
```

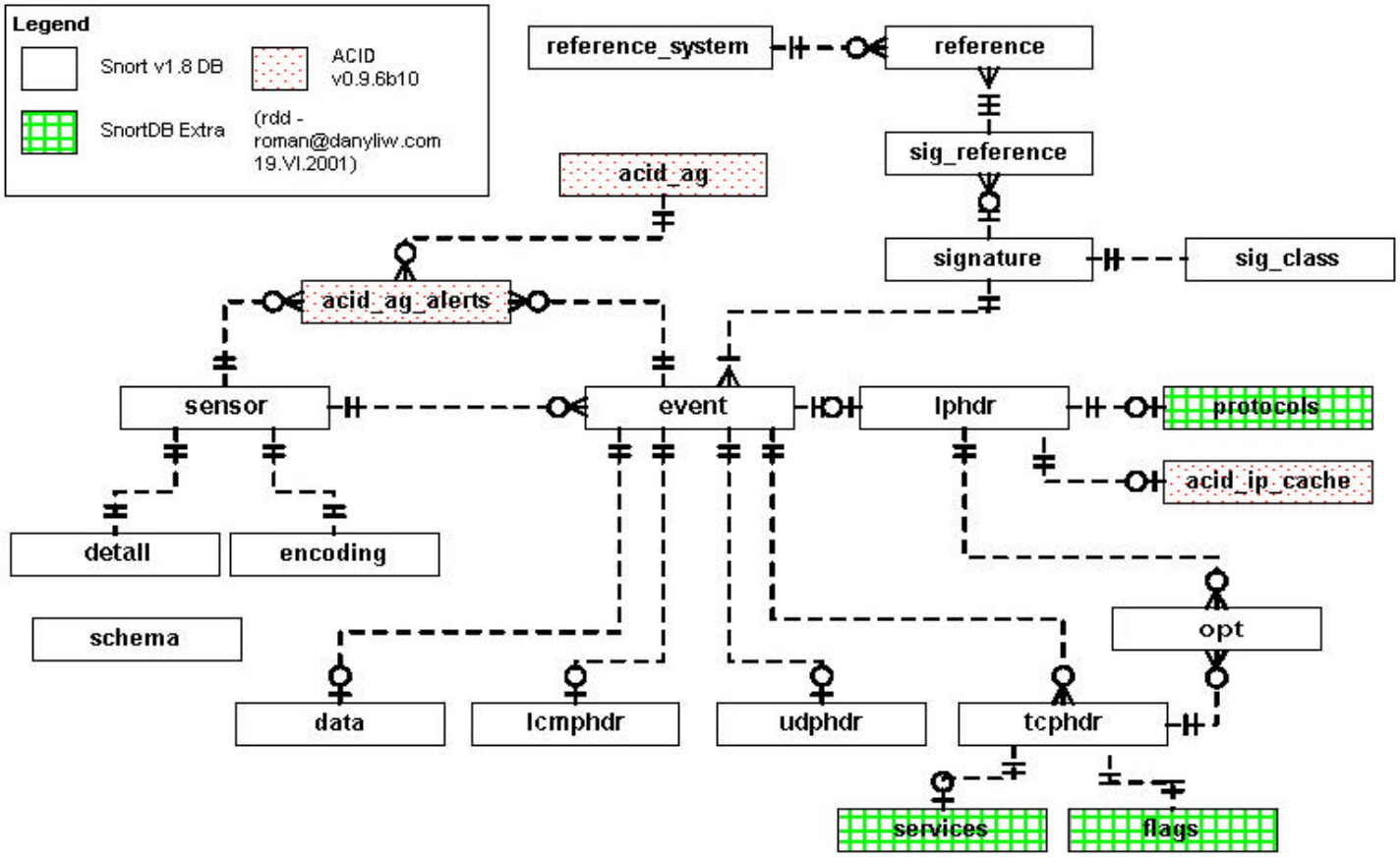
Implementation Details: S.N.O.R.T.

- Creating new rules for S.N.O.R.T.
  - These rules captures important information from FTP transactions in order to calculate file size transferred and bandwidth consumption

```
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 21 (flags: S; msg: "Outgoing FTP CTRL"; classtype:BW-Mon;)
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 20 (flags: S; msg: "Outgoing FTP DATA"; classtype:BW-Mon;)
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 21 (msg: "Outgoing FTP DATA Download"; content:"RETR"; classtype:BW-Mon;)
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 21 (msg: "Outgoing FTP DATA Upload"; content:"STOR"; classtype:BW-Mon;)
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 21 (msg: "Outgoing FTP Xfer Filesize"; content:"150 Opening"; regexp; classtype:BW-Mon;)
BWMon tcp $HOME_NET any <> $EXTERNAL_NET 21 (flags: AP; msg: "Outgoing FTP Xfer Completed"; content:"226 File"; regexp; classtype:BW-Mon;)
BWMon tcp $EXTERNAL_NET 20 -> $HOME_NET any (flags: AF; msg: "Outgoing FTP Xfer Terminated"; classtype:BW-Mon;)
```

Implementation Details: MySQL

- S.N.O.R.T. Table Structure



Implementation Details: MySQL

- SIGNATURE table in S.N.O.R.T.

Field	Type	Null	Key	Default	Extra
Sig_id	int(10) unsigned		PRI	NULL	auto_increment
Sig_name	varchar(255)		MUL		
Sig_class_id	int(10) unsigned		MUL	0	
Sig_priority	int(10) unsigned	YES		NULL	
Sig_rev	int(10) unsigned	YES		NULL	
Sig_sid	int(10) unsigned	YES		NULL	

Implementation Details: MySQL

- EVENT table in S.N.O.R.T.

Field	Type	Null	Key	Default	Extra
Sid	int(10) unsigned		PRI	0	
cid	int(10) unsigned		PRI	0	
signature	int(10) unsigned		MUL	0	
timestamp	datetime		MUL	0000-00-00 00:00:00	

Implementation Details: MySQL

- IPHDR table in S.N.O.R.T.

Field	Type	Null	Key	Default	Extra
Sid	int(10) unsigned		PRI	0	
cid	int(10) unsigned		PRI	0	
ip_src	int(10) unsigned		MUL	0	
ip_dst	int(10) unsigned		MUL	0	
ip_ver	Tinyint(3) unsigned	YES		NULL	
ip_hlen	Tinyint(3) unsigned	YES		NULL	
ip_tos	Tinyint(3) unsigned	YES		NULL	
ip_len	smallint(5) unsigned	YES		NULL	
ip_id	smallint(5) unsigned	YES		NULL	
ip_flags	Tinyint(3) unsigned	YES		NULL	
ip_off	smallint(5) unsigned	YES		NULL	
ip_ttl	Tinyint(3) unsigned	YES		NULL	
ip_proto	Tinyint(3) unsigned			0	
ip_csum	smallint(5) unsigned	YES		NULL	

Implementation Details: MySQL

- TCPHDR table in S.N.O.R.T.

Field	Type	Null	Key	Default	Extra
Sid	int(10) unsigned		PRI	0	
cid	int(10) unsigned		PRI	0	
Tcp_sport	smallint(5) unsigned		MUL	0	
Tcp_dport	smallint(5) unsigned		MUL	0	
Tcp_seq	int(10) unsigned	YES		NULL	
Tcp_ack	int(10) unsigned	YES		NULL	
Tcp_off	Tinyint(3) unsigned	YES		NULL	
Tcp_res	Tinyint(3) unsigned	YES		NULL	
Tcp_flags	Tinyint(3) unsigned		MUL	0	
Tcp_win	smallint(5) unsigned	YES		NULL	
Tcp_csum	smallint(5) unsigned	YES		NULL	
Tcp_urp	smallint(5) unsigned	YES		NULL	

Implementation Details: MySQL

- DATA table in S.N.O.R.T.

Field	Type	Null	Key	Default	Extra
Sid	int(10) unsigned		PRI	0	
cid	int(10) unsigned		PRI	0	
data_payload	text	YES		NULL	

Implementation Details: MySQL

- FTPDATA table in ENABLE

Field	Type	Null	Key	Default	Extra
id	int(10)		PRI	NULL	auto_increment
cid	int(10)			0	
starttime	datetime			0000-00-00 00:00:00	
endtime	datetime			0000-00-00 00:00:00	
Src_host	varchar(40)	YES		NULL	
Src_port	int(10)	YES		NULL	
Dst_host	varchar(40)	YES		NULL	
Dst_port	int(10)	YES		NULL	
filesize	int(20)	YES		NULL	
Bw	int(20)	YES		NULL	
rexmits	int(20)	YES		NULL	

Implementation Details: PERL Script

- The following parameters needs to be configured:

```
# SNORT database parameters
# Change these parameters to fit yours
my $dbuser = "snort";
my $dbpass = "snp3rt";
my $db = "snort";
my $dsn = "DBI:mysql:$db";

# Do NOT change these tables, unless they changes in newer version of SNORT
my $table1 = "event";
my $table2 = "iphdr";
my $table3 = "tcphdr";
my $table4 = "signature";
my $table5 = "data";
```

Implementation Details: PERL Script

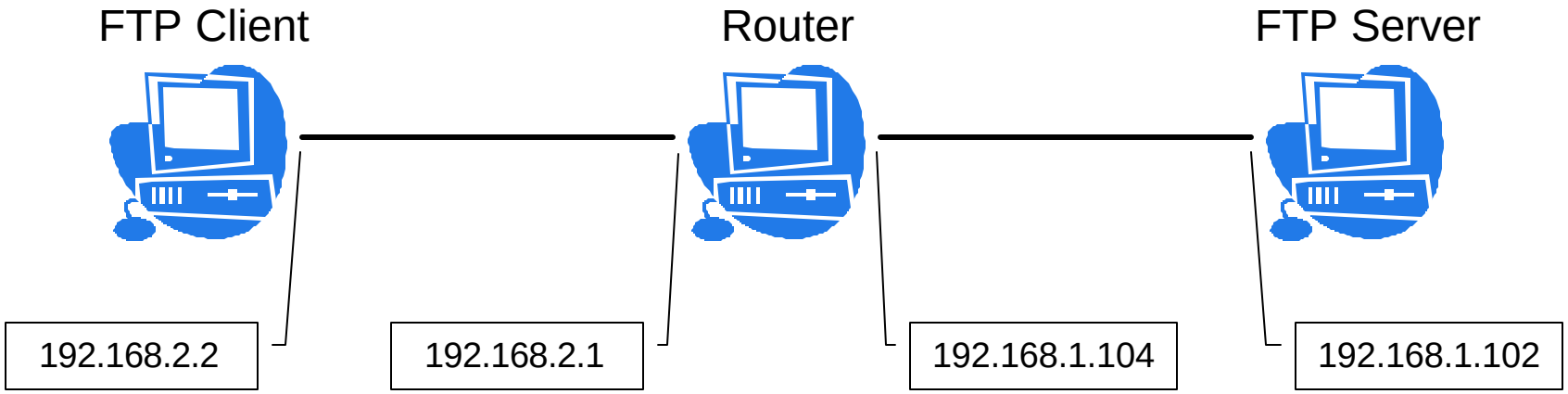
```
# ENABLE database parameters
# Change these accordingly
my $dbuser2 = "enable";
my $dbpass2 = "eNa~";
my $db2 = "enable_data";
my $dsn2 = "DBI:mysql:$db2";
my $table = "ftpdata";

# Replace this with the sig_id from signature table
my $sig_id1 = "47";     # Snort sig_id for Outgoing FTP Data (20)
my $sig_id2 = "52";     # Snort sig_id for Outgoing FTP Filesize (21)
my $sig_idt = "50";     # Snort sig_id for Outgoing FTP Completed (21)
my $sensor_id = "2";    # Snort sensor id
my $datefile = "/etc/snort/previousdate";
```

Implementation Details: PERL Script

- Algorithm Concepts : how do we differentiate between simultaneous data transfer?
  - Socket information is unique to each session
  - CID of the SIG_ID for “Outgoing FTP Xfer Completed” must be greater than CID of the SIG_ID for “Outgoing FTP DATA Download” or “Outgoing FTP DATA Upload”

Testing and Results : Layout



Testing and Results : Test 1

- Single FTP data transfer
- File size of 20kb, 2.3Mb, 14.5Mb, and 585Mb
- Active FTP transfer mode

Testing and Results : Test 1

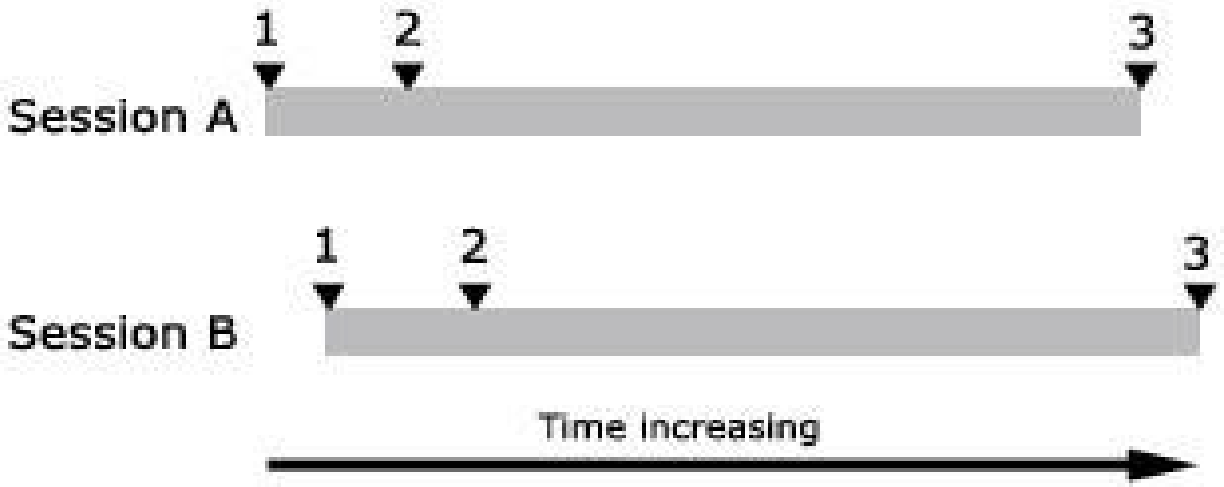
- A portion of the results:

sid	cid	signature	sig_name	timestamp
2	657	46	Outgoing FTP CTRL	2003-06-05 04:58:18
2	658	48	Outgoing FTP DATA Download	2003-06-05 04:58:30
2	659	47	Outgoing FTP DATA	2003-06-05 04:58:30
2	660	52	Outgoing FTP Xfer Filesize	2003-06-05 04:58:30
2	661	50	Outgoing FTP Xfer Completed	2003-06-05 04:58:31
2	662	48	Outgoing FTP DATA Download	2003-06-05 04:58:41
2	663	47	Outgoing FTP DATA	2003-06-05 04:58:41
2	664	52	Outgoing FTP Xfer Filesize	2003-06-05 04:58:41
2	665	50	Outgoing FTP Xfer Completed	2003-06-05 04:58:41

Testing and Results : Test 2

- Multiple Cascading FTP data transfer
- Filesize 14.5 Mb
- Active FTP transfer mode

Testing and Results : Test 2



Test 2 Illustration

Testing and Results : Test 2

sid	cid	signature	timestamp
2	730	46	2003-06-08 20:48:11
2	731	46	2003-06-08 20:48:25
2	732	46	2003-06-08 20:50:42
2	733	46	2003-06-08 20:51:55
2	734	47	2003-06-08 20:52:12
2	735	52	2003-06-08 20:52:12
2	736	51	2003-06-08 20:52:12
2	737	50	2003-06-08 20:52:12
2	738	47	2003-06-08 20:53:09
2	739	52	2003-06-08 20:53:09
2	740	51	2003-06-08 20:53:09
2	741	50	2003-06-08 20:53:09
2	742	48	2003-06-08 20:53:37
2	743	47	2003-06-08 20:53:37
2	744	52	2003-06-08 20:53:37
2	745	48	2003-06-08 20:53:39
2	746	47	2003-06-08 20:53:39
2	747	52	2003-06-08 20:53:39
2	748	51	2003-06-08 20:53:41
2	749	50	2003-06-08 20:53:41
2	750	50	2003-06-08 20:53:42

Testing and Results : Test 2

- This is the result stored in FTPDATA for ENABLE

id	13	14
cid	744	747
starttime	2003-06-08 20:53:37	2003-06-08 20:53:39
endtime	2003-06-08 20:53:41	2003-06-08 20:53:41
src_host	192.168.1.102	192.168.1.102
src_port	21	21
dst_host	192.168.2.2	192.168.2.2
dst_port	1292	1292
filesize	14936458	14936458
Bw	**3734114**	**7468229**
rexmits	0	0

Testing and Results : Test 2

- Session A reveals:

```
ftp> get test.wmv
local: test.wmv remote: test.wmv
200 Port command successful.
150 Opening data connection for test.wmv (14936458 bytes).
226 File sent ok.
14936458 bytes received in 4.96 secs (2.9e+03 Kbytes/sec)
```

- Session B shows:

```
ftp> get test.wmv
local: test.wmv remote: test.wmv
200 Port command successful.
150 Opening data connection for test.wmv (14936458 bytes).
226 File sent ok.
14936458 bytes received in 2.25 secs (6.5e+03 Kbytes/sec)
```

Conclusion

- This project is successful based on 3 assumptions:
  - All FTP downloads are completed successfully
  - Small files that could be transmitted in less than a second is considered negligible.
  - The retransmission rate is considered negligible compared to the size of the files being transferred.

Thank you for coming!