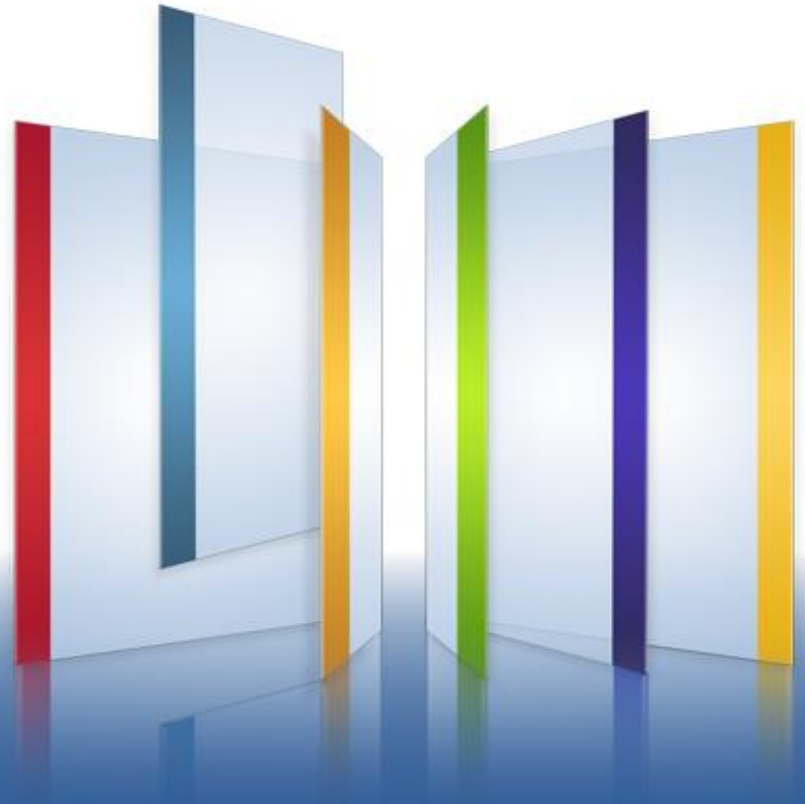
CHECK POINT
INSTITUTE FOR
INFORMATION SECURITY

בית הספר למדעי המחשב על שם בלבטניק
The Blavatnik School of Computer Science

Check Point
SOFTWARE TECHNOLOGIES LTD.



Lecture 4: Stateful Inspection, Advanced Protocols

Agenda

1. Advanced protection techniques
2. File Transfer Protocol - FTP
3. HyperText Transfer Protocol - HTTP
4. About next Assignment

Agenda

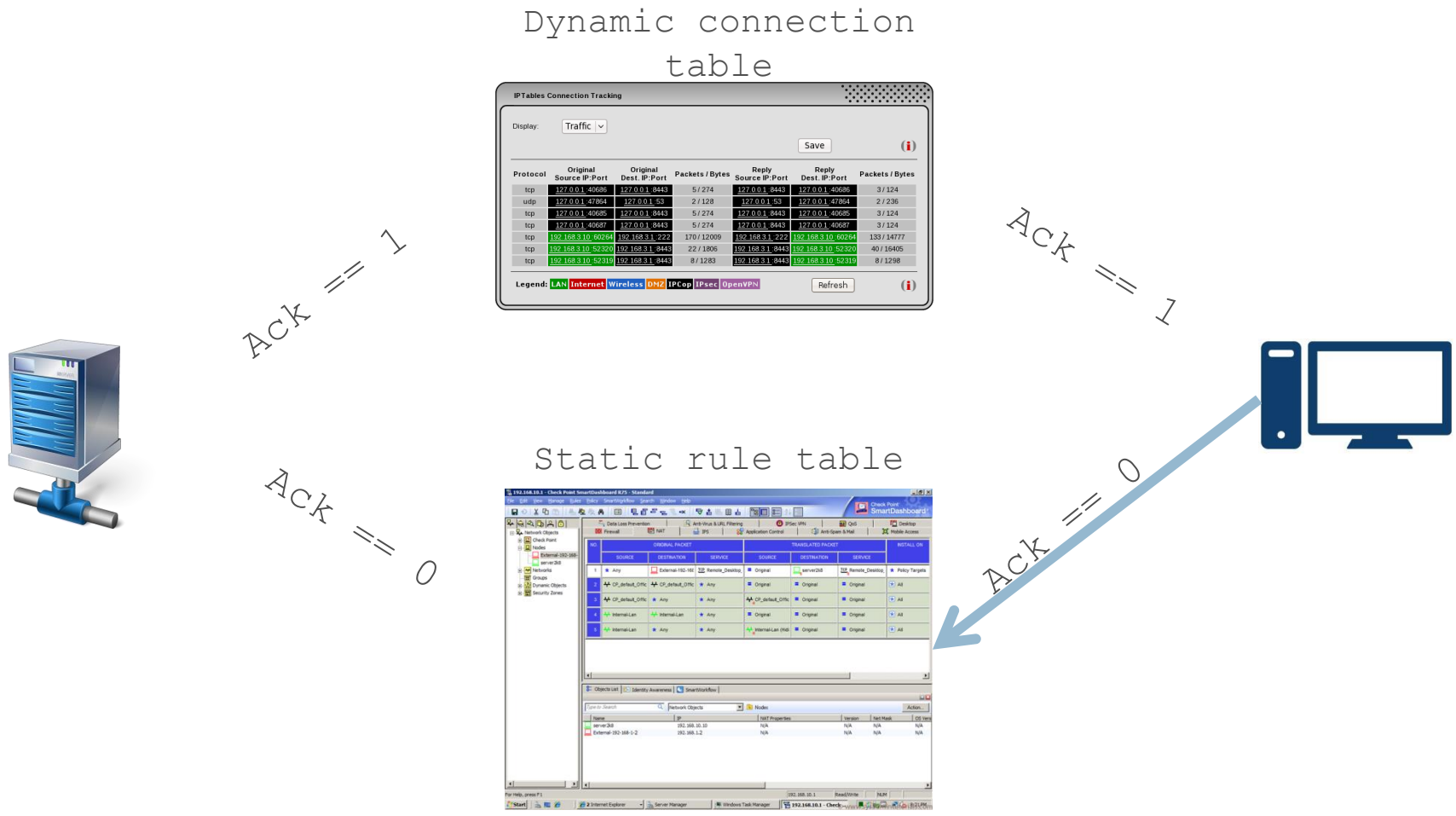
1. **Advanced protection techniques**
2. File Transfer Protocol - FTP
3. HyperText Transfer Protocol - HTTP
4. About next Assignment

Stateful connection tracking

- Advanced firewalls intelligently associating new packet requests with existing legitimate connections.
- A **connection tables** tracks existing TCP connections
- If an incoming TCP packet has ACK=0 then it's a new attempted connection
  - consult the static **rule table** and (if accepted) record a new connection in the **connection table**
- If ACK=1, check the packet against the **connection table** (but not the static **rule table**)
  - If connection present and packet is valid according to protocol state machine, accept and update the **connection table** record
  - Otherwise reject

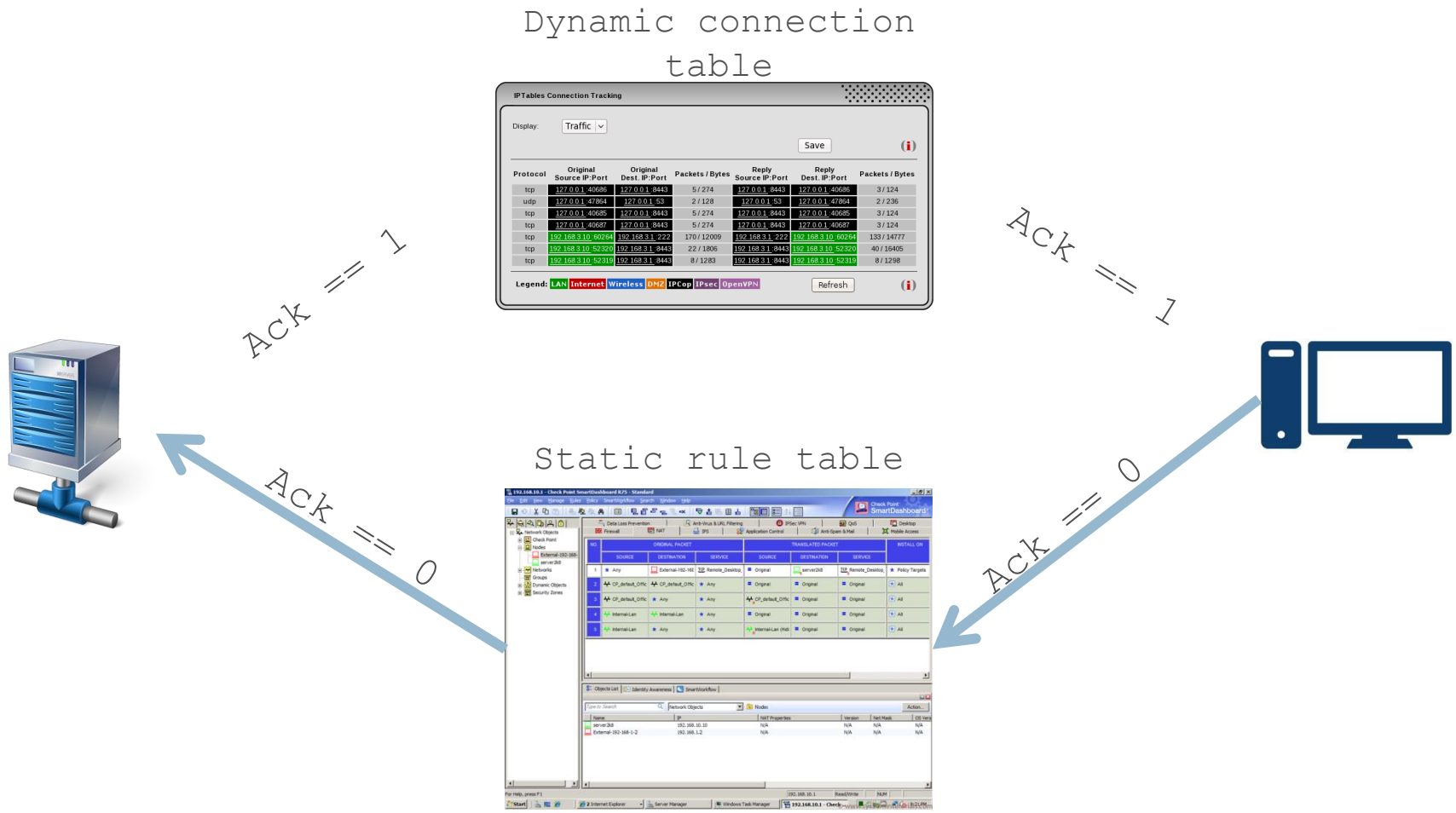
Connection table

- Step 1: SYN



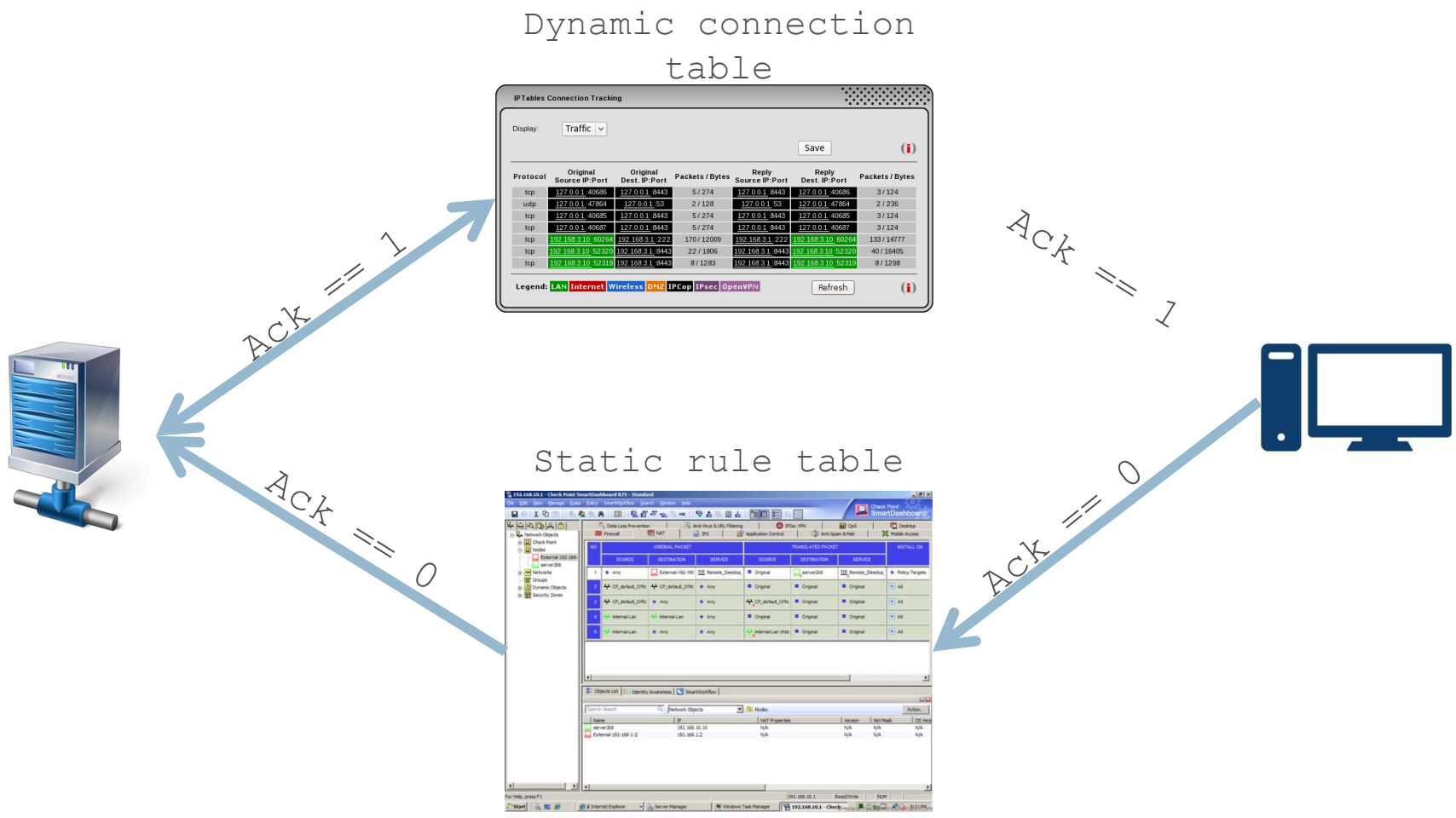
Connection table

- Step 1 (cont.): Check the SYN packet and pass it to the server



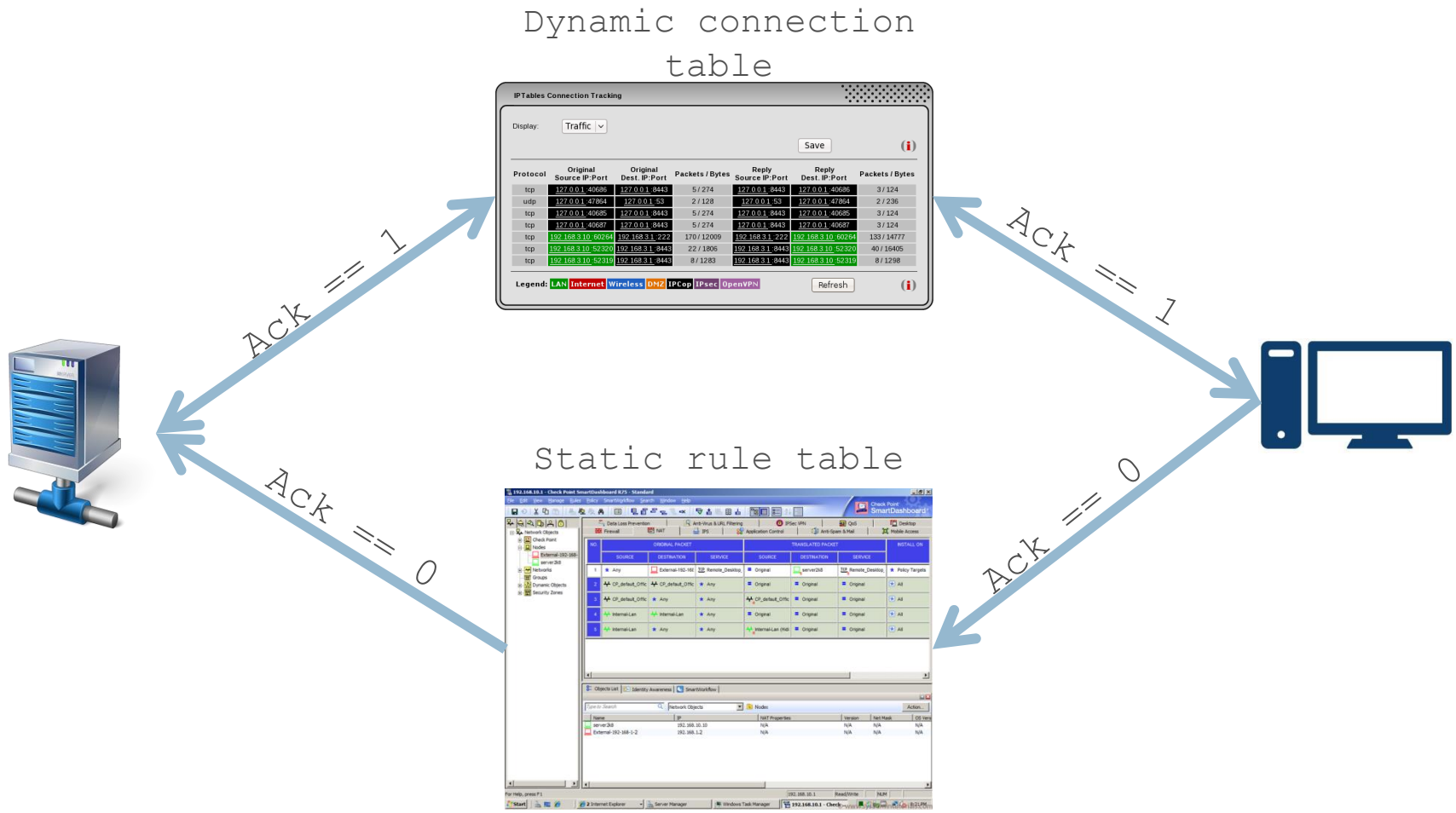
Connection table

- Step 2: SYN-ACK



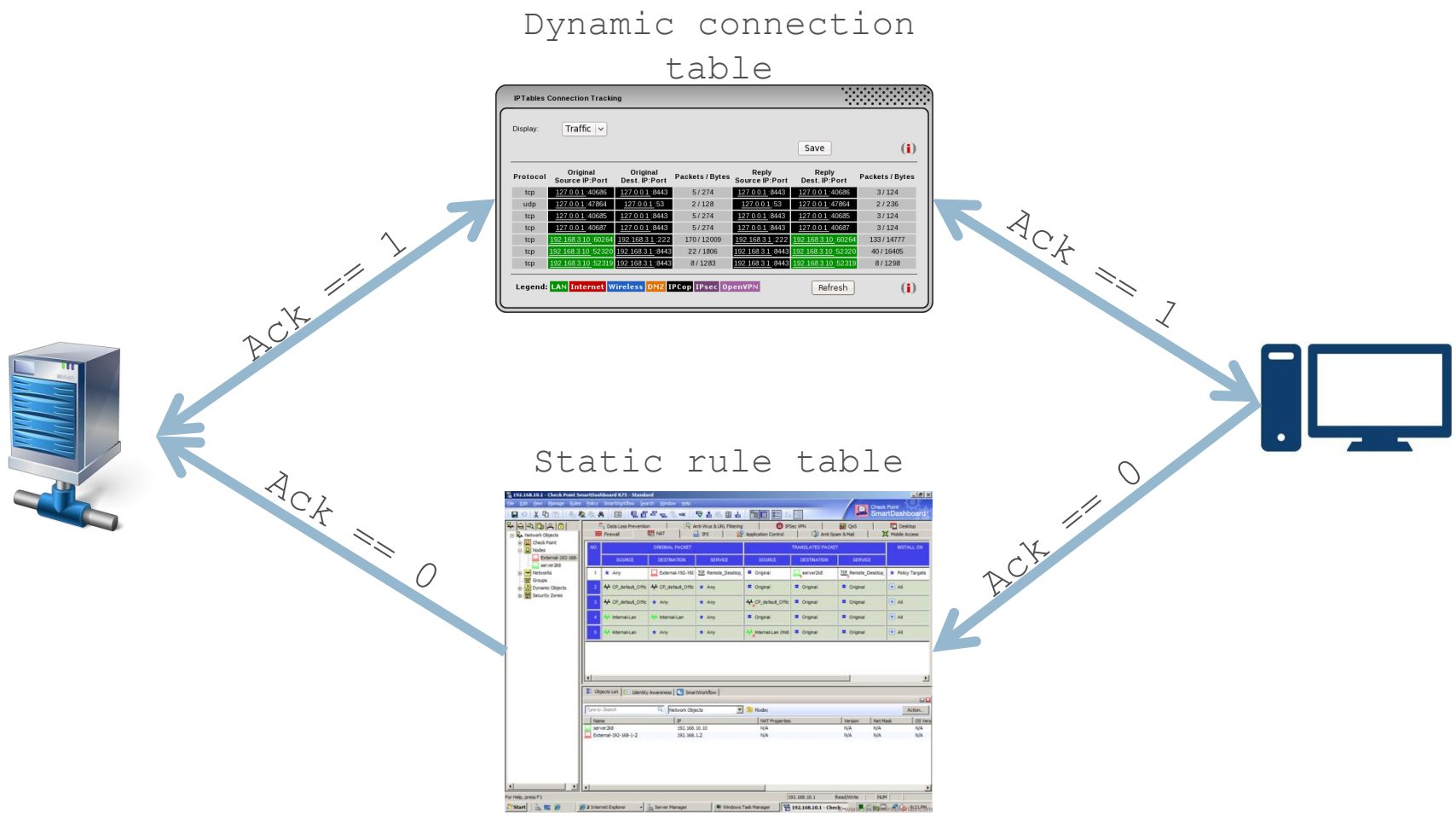
Connection table

- Step 2 (cont.): read/write the session and pass it forward



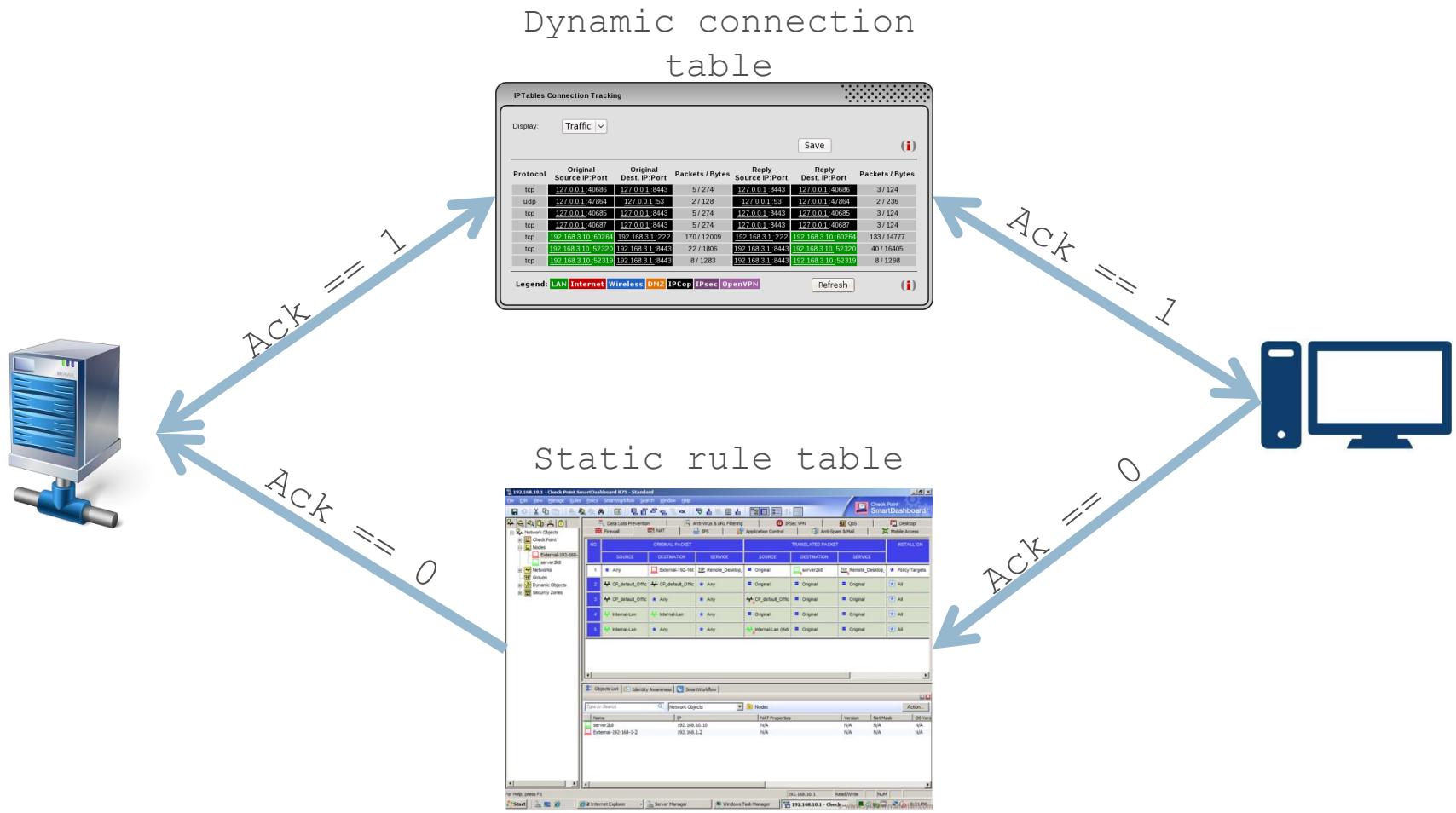
Connection table

- Step 3: ACK



Connection table

- Step 3(cont.): check in the connection table and pass it



Connection table – a closer look

Source IP	Dest. IP	Source Port	Dest. Port	State
192.168.1.15	212.69.12.131	5319	44431	Wait for Syn-ack
212.69.12.131	192.168.1.15	44431	5319	Syn-ack sent
192.168.1.31	10.0.1.1	8841	21	ftp established
10.0.1.1	192.168.1.31	21	8841	ftp established
192.168.1.31	10.0.1.1	8988	20	ftp data session
10.0.1.1	192.168.1.31	20	8988	ftp data session
192.168.1.31	10.0.1.1	10156	80	Syn sent

- Each 2-way session is represented by two rows in the connection table, one for each direction
- The last row is a SYN packet which passed the static rule table, and now we wait for the SYN-ACK packet with ACK == 1

Connection table

- Each approved connection will be saved in a dynamic table
- Each row will contain data about the saved connection
  - Source IP address
  - Source port
  - Destination IP address
  - Destination port
  - State
- Protocols can have several states and we always need to know the current state of the connection. For example:
  - 3-way handshake and waiting for “syn ack” from server
  - Ftp connection who wait to receive “ready” status from server
  - Etc.

Alternative approach (Linux's iptables)

- Alternative: support stateful rules in the main table.
- For example, iptables pass **every** packet (even those of established connection) through the same rule table
- The table has the ability to invoke modules, loaded into the Linux kernel.
- Modules can work on all layers, and can access stateful data structures:
  - inspect IP address, ports, or even GEOIP lookup of IP address
  - inspect specific bytes, or doing a string search, in the raw packet
  - inspect MAC address
  - inspect flags of TCP/UDP/ICMP
  - **stateful connection tracking: look up a dynamic connection table**
  - And many more

Stateful inspection

- Some protocols required connection tracking to establish a secured connection
  - We can't just allow traffic from numerous ports
- Both in TCP and UDP there are protocols as such:
  - TFTP
    - Both client and server open random ports and connect with each other
  - VoIP protocols
    - SIP
    - H.323 protocols
  - Basically anything on top TCP (and some UDP also)

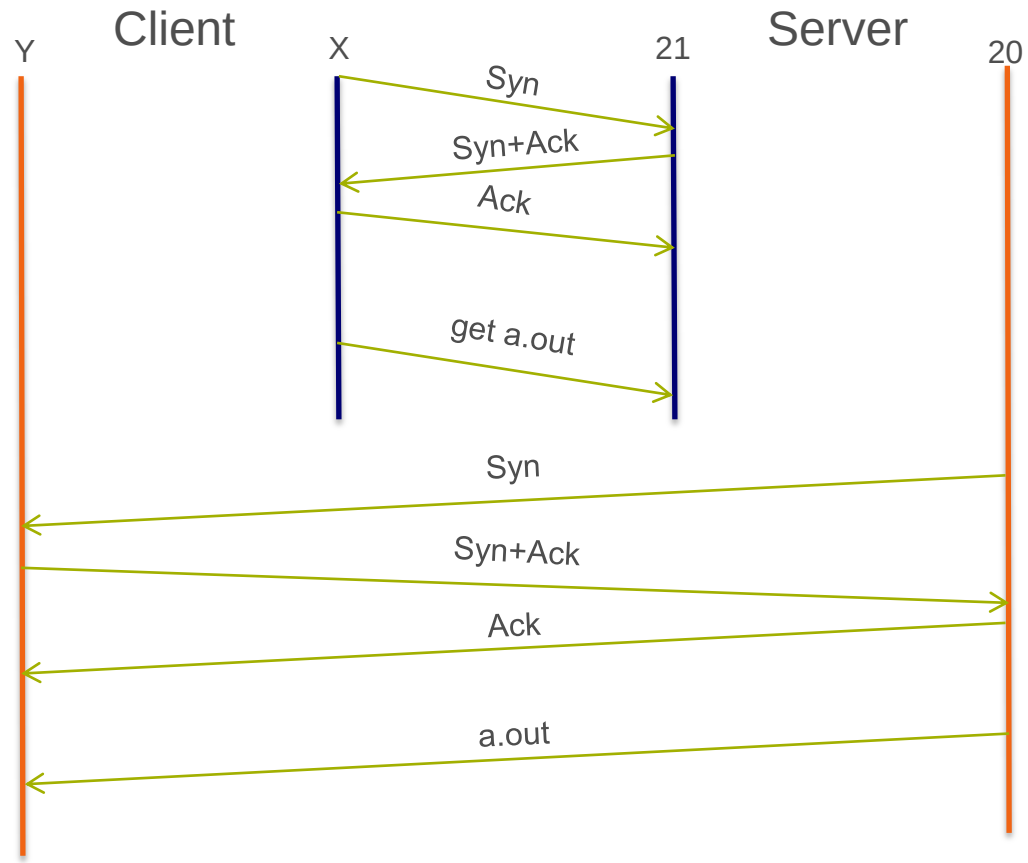
Agenda

1. Advanced protection techniques
2. **File Transfer Protocol - FTP**
3. HyperText Transfer Protocol - HTTP
4. About next Assignment

FTP (File Transfer Protocol)

- The client send the server the port it’s open for data connection
- The firewall need to be able to open connections to arbitrary high ports for it to function properly.
- Thus, the firewall needs to read the payload (the data itself) of the packet to realize we are in FTP connection and identify the FTP command that specifies the receive port number to open.
- The command is inside a TCP stream
  - May be, for example, fragmented across several TCP packets, so inspecting packets individually does not suffice

File Transfer Protocol - FTP



File Transfer Protocol - FTP

- Can be enforced only in stateful inspection.
- Receive port number to open in the packet's payload
- After connection is established, the client send to the server a packet with a special request, called PORT
- A PORT request asks the server to use a different port to the data connection: the server makes a TCP connection to the client though this port.
- The PORT request has a parameter in the form:
  - *h1*,*h2*,*h3*,*h4*,*p1*,*p2*
  - the client is listening for connections on TCP port *p1**256+*p2* at IP address *h1*.*h2*.*h3*.*h4*.

File Transfer Protocol - FTP

- For example, after a connection from 10.0.1.1 to FTP server 10.0.2.2, the client send the following textual command on the TCP connection:

```
PORT 10,0,1,1,165,126
```

- The server understand that client at IP address 10.0.1.1 has opened the port 165*256+126=42366 for transferring a file
- Our firewall now should keep track of packets who reach to this port and IP address, and not automatically drop them, but to inspect them and see if they're related to the ftp connection

Agenda

1. Advanced protection techniques
2. File Transfer Protocol - FTP
3. **HyperText Transfer Protocol - HTTP**
4. About next Assignment

HTTP (HyperText Transfer Protocol)

- Another example to a protocol with lots of information in the payload
- In contrast to FTP, HTTP was designed to operate over a single TCP port and a single TCP connection, to avoid the difficulties we've seen for FTP. However, there is still a lot of state to be kept **between** packets in that single TCP connection. We need to listen and search for http request
- Once we found the client sent GET request, we need to inspect the following packets to see what response we'll get
- Examples

HyperText Transfer Protocol - HTTP

- Regular 200 OK response:

```
GET /secws16/ HTTP/1.1
Host: course.cs.tau.ac.il
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,he;q=0.6
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/41.0.2272.118 Safari/537.36

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0
Connection: Keep-Alive
Content-Length: 4503
Content-Type: text/html; charset=utf-8
Server: Apache/2.2.14 (Ubuntu)
Vary: Accept-Encoding
```

HTTP stream: data in the same connection

- We then ask from the server and receive the data on the same connection (in contrast to FTP)

```
GET http://www.walla.co.il/ HTTP/1.1
Host: www.walla.co.il
Proxy-Connection: keep-alive
Accept:…

HTTP/1.1 200 OK
Content-Type: …
…
<html lang="he-IL" xml:lang="he-IL">
    <head>
            <meta charset="UTF-8" />
            <title> !וואלהNEWS</title>
            <!--[if lt IE 9]>
            <style>
            …
```

Agenda

1. Advanced protection techniques
2. File Transfer Protocol - FTP
3. HyperText Transfer Protocol - HTTP
4. **About next Assignment**