Fixing HTTP/2 and preparing for HTTP/3 over QUIC

Robin Marx  @programmingart
Robin Marx
@programmingart
PhD Student
Web Performance
Internet Protocols
Contributor to QUIC and HTTP/3
I list my references
01 Fixing HTTP/2

- Denialing of Services
- Pushing the wrong things
- Buggy prioritization
1. Flooding HTTP/2

1. CVE-2019-9511 “Data Dribble”
2. CVE-2019-9512 “Ping Flood”
3. CVE-2019-9513 “Resource Loop”
4. CVE-2019-9514 “Reset Flood”
5. CVE-2019-9515 “Settings Flood”
6. CVE-2019-9516 “0-Length Headers Leak”
7. CVE-2019-9517 “Internal Data Buffering”
8. CVE-2019-9518 “Empty Frames Flood”

- Tomcat
- macOS X
- nginx
- Go
- H2O
- LiteSpeed
- Windows
- NodeJS
2. Push in theory

index.html
- style.css
- font.woff2

https://calendar.perfplanet.com/2016/http2-push-the-details/
2. Push in practice

index.html
- style.css
- font.woff2

 Push in practice

browser

server

GET /index.html
200 OK /index.html

GET /style.css
200 OK /style.css

GET /font.woff2
200 OK /font.woff2

no push

push everything

14 KB

https://calendar.perfplanet.com/2016/http2-push-the-details/
2. Congestion Control

14 KB

Packet(s) lost
2. Push in practice

Index.html
- Style.css
- Font.woff2

GET /index.html
200 OK /index.html
GET /style.css
200 OK /style.css
GET /font.woff2
200 OK /font.woff2

14 KB

Browser
Server

GET /index.html
200 OK /index.html
GET /style.css
200 OK /style.css
GET /font.woff2
200 OK /font.woff2

Browser
Server

GET /index.html
200 OK /index.html
GET /style.css
200 OK /style.css
GET /font.woff2
200 OK /font.woff2

10kB /index.html
PUSH 4kB /style.css
ACKS
PUSH 16kB /style.css

10kB.html
20kB.css
with push

https://calendar.perfplanet.com/2016/http2-push-the-details/
Other issues

Cache Digests are dead
- No standard way to know what the client has cached
- Can work around that with cookies, service workers or other heuristics

Quirks in browsers and servers
- e.g., no push for fetch() or XHR, <iframe>

Push is per-connection
- could end up sending things twice

Used by 0.5% of sites
CDNs

“warm up” the connection from client to CDN

14 KB

112 KB

https://developer.akamai.com/ion/adaptive-acceleration
https://tools.ietf.org/html/draft-bishop-httpbis-push-cases-00
https://www.fastly.com/blog/optimizing-http2-server-push-fastly
Beyond the initial page load!

Speed up API calls

```json
{
    "id": 666,
    "images": [
        "https://example.com/big_image1.jpg",
        "https://example.com/big_image2.jpg",
        "https://example.com/big_image3.jpg"
    ],
    "details": "https://example.com/details/666"
}
```

See also: "edge workers"
3. Multiplexing and Prioritization

HTTP/1.1
- main.js
- style.css
- image1.jpg
- image2.jpg

HTTP/2
- multiplexed
3. Multiplexing and Prioritization

HTTP/1.1
- main.js
- style.css
- image1.jpg
- image2.jpg

HTTP/2
- incremental
- sequential
- mixed
Welcome to the jungle
HTTP/2 Prioritization in Browsers
HTTP/2 Prioritization in Browsers

``sequential``
everything

``incremental``
everything
HTTP/2 Prioritization in Browsers

Up to 50% slower than Chrome
## HTTP/2 Prioritization in Servers

### deployments pass

<table>
<thead>
<tr>
<th>CDN / Hosting</th>
<th>Status</th>
<th>Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akamai</td>
<td>Pass</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Amazon CloudFront</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Pass</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
<tr>
<td>Akamai</td>
<td>Fail</td>
<td>Dec 22, 2018</td>
</tr>
</tbody>
</table>

https://www.youtube.com/watch?v=ct5MvtmL1NM
https://github.com/andydavies/http2-prioritization-issues
https://www.slideshare.net/patrickmeenan/http2-in-practice
Bufferbloat

Expected:
Types of Headaches

Migraine

Hypertension

Stress

Prioritization
What to do

- **Test** your pages

webpagetest.org
What to do

- **Test** your pages
  [webpagetest.org](https://webpagetest.org)

- **Pick a different server / CDN**
- Reduce buffer sizes
- Use BBR congestion controller
- Override browser-sent priorities on the server
- Be careful with `<link rel="preload" />`

[https://www.youtube.com/watch?v=ct5MvtmL1NM](https://www.youtube.com/watch?v=ct5MvtmL1NM)
[https://www.slideshare.net/patrickmeenan/http2-in-practice](https://www.slideshare.net/patrickmeenan/http2-in-practice)
[https://twitter.com/csswizardry/status/1180810824862064646?s=20](https://twitter.com/csswizardry/status/1180810824862064646?s=20)
[https://andydavies.me/blog/2019/02/12/preloading-fonts-and-the-puzzle-of-priorities/](https://andydavies.me/blog/2019/02/12/preloading-fonts-and-the-puzzle-of-priorities/)
HTTP/3 prioritization proposal

Header-based, simpler

```
:method = GET
:scheme = https
:path = /image.jpg
priority = urgency=3, progressive=1
```

<table>
<thead>
<tr>
<th>Urgency Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 prerequisite (Section 2.1.1)</td>
</tr>
<tr>
<td>0 default (Section 2.1.2)</td>
</tr>
<tr>
<td>between 1 and 5 supplementary (Section 2.1.3)</td>
</tr>
<tr>
<td>6 background (Section 2.1.4)</td>
</tr>
</tbody>
</table>

https://github.com/kazuho/draft-kazuho-httpbis-priority
02
HTTP/3 over QUIC

- Encrypted UDP
- Load balancing and routing
- 0-RTT setup
- QUIC and HTTP/3 discovery
- Debugging and tooling
QUIC is TCP 2.0
HTTP/3 is HTTP/2 for QUIC

2015
- Single connection
- Multiplexing
- Header Compression
- Server Push
- Prioritization

2019
- Single connection
- Multiplexing
- Header Compression
- Server Push
- ~Prioritization

application
HTTP/2
TLS
TCP

transport
application
HTTP/3
TLS
QUIC
UDP
QUIC encrypts the Transport layer

- Application
  - Secure
    - Flags, connection id
    - Secure
      - Ports
      - Flags, options
      - Flow control, retransmits, packet seq nr, acknowledgements, ports
  - Transport
1. Networks blocking QUIC

3\(^1\)-20\(^2\)% of networks block QUIC/UDP

```
The websites will still work, so you might as well choose security over a tiny increase in performance``

\(^1\) https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf
\(^3\) https://www.fastvue.co/fastvue/blog/googles-quic-protocols-security-and-reporting-implications/
1. Networks blocking QUIC

3\(^1\) - 20\(^2\)% of networks block QUIC/UDP

```
```

```
```

"The websites will still work, so you might as well choose security over a tiny increase in performance"\(^3\)

\(\text{dschinazi} 9:17\ \text{PM}\)

QUIC uses the local store but it fails the handshake if the root CA is not in the default set. This was done by policy to prevent antivirus software from MITM QUIC so we can keep evolving QUIC.

\(^1\) https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf
\(^3\) https://www.fastvue.co/fastvue/blog/googles-quic-protocols-security-and-reporting-implications/
1. Built-in security features

“SYN flood”
- Stateless Retry

UDP reflection / amplification
- Maximum amplification factor of 3

2. Connection ID

- IP addresses
- ports
- flags, \textit{connection id}
- SECURE

0-20 bytes
2. NAT rebinding + connection migration

193.190.10.140:12345
Conn. ID: 0xDEADBEEF

185.194.187.142:23456
Conn. ID: 0xDEADBEEF

<table>
<thead>
<tr>
<th>IP addresses</th>
<th>ports</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

flags, **connection id**
0-20 bytes

SECURE
2. NAT rebinding + connection migration

193.190.10.140:12345  
Conn. ID: 0xDEADBEEF

185.194.187.142:23456  
Conn. ID: 0x8BADFOOD

IP addresses
ports
flags, connection id
SECURE
0-20 bytes
2. NAT rebinding + connection migration

Client

0xDEADBEEF
0x8BADF00D
0xFACEB00C
0xDEADC0DE

Connection #5

Server

0xDEADBEEF
0x8BADF00D
0xFACEB00C
0xDEADC0DE

193.190.10.140:12345
Conn. ID: 0xDEADBEEF

185.194.187.142:23456
Conn. ID: 0x8BADF00D
2. Load balancing and routing breaks

NOR

Connection ID

5-tuple hash:
- Source IP
- Source Port
- Destination IP
- Destination Port
- Protocol

Private IP, port

VM

Private IP, port

VM

Private IP, port

VM
2. **Stateless** load balancing and routing

---

Connection IDs

Client Connection IDs

Server Connection IDs

Server Nr

- 214
- ?

---

https://blog.cloudflare.com/warp-technical-challenges
2. **Stateless** load balancing and routing

Load balancers and servers need to use the same semantics

https://blog.cloudflare.com/warp-technical-challenges
3. Connection setup

TCP + HTTP/2
TLS 1.2

SYN
SYN/ACK
ACK
TLS
TLS
TLS
GET /index.html
<head>

0-RTT
QUIC
3. Connection setup

TCP + HTTP/2
TLS 1.2

QUIC
TLS 1.3
RESUME
0-RTT
3. Connection setup

0-RTT

- Store state on the server
  - Share across server pool...
- Store state in tickets themselves
  - Encrypt tickets separately
  - Share ticket keys...
  - Rotate ticket keys frequently

https://www.rfc-editor.org/rfc/rfc8470.html
3. Connection setup

0-RTT

- Store state on the server
  - Share across server pool...
- Store state in tickets themselves
  - Encrypt tickets separately
  - Share ticket keys...
  - Rotate ticket keys frequently
- Replay attacks: idempotent only

https://www.rfc-editor.org/rfc/rfc8470.html
4. QUIC discovery
4. QUIC discovery

QUIC

TCP

4. QUIC discovery

QUIC

TCP

100 ms delay
4. QUIC discovery

QUIC

TCP

100 ms delay

Happy eyeballs

4. QUIC discovery

QUIC IPv6

TCP IPv6

100 ms

QUIC IPv4

200 ms

TCP IPv4

300 ms

Happy eyeballs

Replay "attacks"...
4. QUIC and HTTP/3 discovery with alt-svc

TCP

UDP

QUIC

Alt-Svc: h3=":49288";quic="1,1abadaba,51303334"

QUIC versions

PORT

TCP

QUIC

UDP

PORT

2 https://github.com/MikeBishop/dns-alt-svc
https://daniel.haxx.se/blog/2019/03/03/alt-svc-in-curl/
5 https://bugs.chromium.org/p/chromium/issues/detail?id=1015101#c1
4. QUIC and HTTP/3 discovery with alt-svc

TCP

UDP

QUIC

Alt-Svc: h3=":49288";quic="1,1abadaba,51303334"

QUIC versions

PORT

- Need alt-svc cache

2 https://github.com/MikeBishop/dns-alt-svc
3 https://daniel.haxx.se/blog/2019/03/03/alt-svc-in-curl/
5 https://bugs.chromium.org/p/chromium/issues/detail?id=1015101#c1

4. QUIC and HTTP/3 discovery with alt-svc

TCP

UDP

QUIC

Alt-Svc: h3=":49288";quic="1,1abada,51303334"

- Need alt-svc cache
- Still needs racing
  - Networks block QUIC

https://github.com/MikeBishop/dns-alt-svc
https://daniel.haxx.se/blog/2019/03/03/alt-svc-in-curl/
https://bugs.chromium.org/p/chromium/issues/detail?id=1015101#c1
4. QUIC and HTTP/3 discovery with alt-svc

- Need alt-svc cache
- Still needs racing
  - Networks block QUIC
  - TCP first
    - Unless we hi-jack DNS!²

Alt-Svc: h3=":49288";quic="1,1abadaba,51303334"

TCP

UDP

QUIC versions

PORT

QUIC

² [https://bugs.chromium.org/p/chromium/issues/detail?id=1015101#c1](https://bugs.chromium.org/p/chromium/issues/detail?id=1015101#c1)

² [https://github.com/MikeBishop/dns-alt-svc](https://github.com/MikeBishop/dns-alt-svc)

² [https://daniel.haxx.se/blog/2019/03/03/alt-svc-in-curl/](https://daniel.haxx.se/blog/2019/03/03/alt-svc-in-curl/)


4. QUIC and HTTP/3 discovery with alt-svc

- Need alt-svc cache
- Still needs racing
  - Networks block QUIC
  - TCP first
    - Unless we hi-jack DNS! \(^2\)
- PITA to test locally

TCP

UDP

QUIC

Alt-Svc: h3=":49288";quic="1,1abadaba,51303334"

QUIC versions

PORT
Types of Headaches

Migraine

Hypertension

Stress

ALT-SVC
5. Debugging QUIC and HTTP/3

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:00:0000</td>
<td>173.194.72.125</td>
<td>192.168.64.1</td>
<td>TLSv1</td>
<td>103</td>
<td>Application Data</td>
</tr>
<tr>
<td>2</td>
<td>00:00:0125</td>
<td>173.194.72.125</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>66</td>
<td>&gt; https [ACK] Seq=1 Ack=38 Win=2239 Len=0 Tlsal=114955745 Tlsec=47235863</td>
</tr>
<tr>
<td>3</td>
<td>00:10:0000</td>
<td>12.7840563</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>74</td>
<td>74 579574 192.168.64 1000 M5S14600 80H E04ACK M5S14600 80H E04ACK</td>
</tr>
<tr>
<td>4</td>
<td>00:12:765363</td>
<td>IntelCor bfr dd:24</td>
<td>Broadcast</td>
<td>ARP</td>
<td>42</td>
<td>Who has 192.168.64.7 Tel=192.168.6.7</td>
</tr>
<tr>
<td>5</td>
<td>00:12:765364</td>
<td>GeneTeke Sc:ee:dd</td>
<td>IntelCor bfr dd:24</td>
<td>ARP</td>
<td>42</td>
<td>192.168.64.15 at 21:20:48:64</td>
</tr>
<tr>
<td>6</td>
<td>00:12:765365</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>60</td>
<td>5501 &gt; 53750 [ACK] Seq=1 Ack=1 Win=14688 Len=0</td>
</tr>
<tr>
<td>7</td>
<td>00:12:765366</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>78</td>
<td>37595 &gt; 5001 [IPSH, ACK] Seq=25 Ack=1 Win=14688 Len=24</td>
</tr>
<tr>
<td>8</td>
<td>00:12:765367</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=1405 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>9</td>
<td>00:12:765368</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=2495 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>10</td>
<td>00:12:765369</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=4485 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>11</td>
<td>00:12:765370</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=4655 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>12</td>
<td>00:12:765371</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=4825 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>13</td>
<td>00:12:765372</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=4995 Ack=1 Win=14600 Len=1460</td>
</tr>
<tr>
<td>14</td>
<td>00:12:765373</td>
<td>192.168.64.7</td>
<td>192.168.64.1</td>
<td>TCP</td>
<td>1514</td>
<td>37595 &gt; 5001 [ACK] Seq=5065 Ack=1 Win=14600 Len=1460</td>
</tr>
</tbody>
</table>
```

**Frame 9:** 1514 bytes on wire (12124 bits), 1514 bytes captured (12124 bits)
- Transmission Control Protocol, Src Port: 35795 (35795), Dst Port: 5001 (5001), Seq: 25, Ack: 1, Len: 1460
- Header length: 20 bytes
- Flags: 8x10 (ACK)
- Window size value: 933

Profile: Default
5. qlog: structured endpoint logging

```json
{
    "event_fields": ["time", "group_id", "category", "event", "data"],
    "events": [
        [1553986553579, 0, "transport", "packet_received", {}],
        [1553986553580, 0, "recovery", "metrics_updated", {"smoothed_rtt": 85}]
    ]
}
```

https://quic.edm.uhasselt.be/
5. qlog: structured endpoint logging

```json
{
    "event_fields": [
        "time", "group_id", "category", "event", "data"
    ],
    "events": [
        [1553986553579, 0, "transport", "packet_received", {}],
        [1553986553580, 0, "recovery", "metrics_updated", {"smoothed_rtt": 85}],
        [1553986553588, 1, "http", "frame_parsed", {"frame_type": "DATA"}],
        [1553986553598, 0, "transport", "packet_sent", {}]
    ]
}
```

Nine QUIC implementations currently output qlog

@rmarx we currently have qlog enabled in prod with similar amounts of events being recorded a day as I quoted before (dozens of billions).

https://quic.edm.uhasselt.be/
5. qvis: tooling and visualization

https://qvis.edm.uhasselt.be

https://quic.edm.uhasselt.be

5. qvis: tooling and visualization

https://qvis.edm.uhasselt.be

https://tools.ietf.org/html/draft-ma...m-qlog-main-schema-01
https://tools.ietf.org/html/draft-ma...m-qlog-event-definitions-quic-h3-01
Benefits from all this complexity?

Some

~8% faster

Up to -26% at p99

Others

Users don’t care

2-9x CPU needed

https://www2.cs.duke.edu/courses/fall18/compsci514/readings/QUIC-sigcomm2017.pdf
Benefits from all this complexity?

Some

~8% faster
Up to -26% at p99

Remember

Google and Facebook deploy QUIC at scale
Huge part of internet traffic today is QUIC

Others

Users don’t care
2-9x CPU needed

https://www2.cs.duke.edu/courses/fall18/compsci514/readings/QUIC-sigcomm2017.pdf
Practical way forward

Experiment now, test in June, deploy end of 2020

https://github.com/quicwg/base-drafts/wiki/Implementations
https://www.zdnet.com/article/cloudflare-google-chrome-and-firefox-add-http3-support/
Practical way forward

Experiment now, test in June, deploy end of 2020

Use a CDN

- Cloudflare, Fastly, Google and Microsoft: heavily invested
- Akamai: does google QUIC already
- Amazon: also working on –something–

https://github.com/quicwg/base-drafts/wiki/Implementations
https://www.zdnet.com/article/cloudflare-google-chrome-and-firefox-add-http3-support/
Practical way forward

Experiment now, test in June, deploy end of 2020

Use a CDN
- Cloudflare, Fastly, Google and Microsoft: heavily invested
- Akamai: does google QUIC already
- Amazon: also working on something-

Server support is coming
- nginx announced (for now: cloudflare patch)
- LiteSpeed, NodeJS, aioquic open source

Browser support is coming
- Chrome, Edge, Firefox in ‘Canary’
- Safari in progress

https://github.com/quicwg/base-drafts/wiki/Implementations
https://www.zdnet.com/article/cloudflare-google-chrome-and-firefox-add-http3-support/
GO FORTH and REPRODUCE!
Cyberconflict: A new era of war, sabotage, and fear

Sarah Connor
9:55am-10:10am Wednesday, March 27, 2019
Location: Balcony

We're living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you're often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

Sarah Connor explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we're uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

Sarah Connor
The New York Times

Sarah Connor is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.

Please provide feedback
@programmingart
Image sources

- https://www.wallpaperflare.com/terminator-genisys-t-800-copy-space-black-background-close-up-wallpaper-qyrlz
- https://twitter.com/AndyDavies/status/1065916677408346112
- https://www.reddit.com/r/Terminator/comments/9bb3kf/happy_judgment_day/
- https://tenor.com/search/terminator-gatling-gun-gifs
- https://i.ytimg.com/vi/rmmF11TKueA/maxresdefault.jpg
03

Extra slides

- Too much content, too little time
Prioritization matters

- main.js fully downloaded
- style.css fully downloaded
- main.js fully downloaded
- style.css fully downloaded
Prioritization matters

Image 1 fully downloaded

Image 2 fully downloaded

Image 1 fully downloaded

Image 2 fully downloaded

https://h3.edm.uhasselt.be
Other HTTP/2 (implementation) issues

- Mixed content (HTTP to HTTPS)
- Coalescing connections can be flaky
- Support from intermediaries/libraries (e.g., proxies, caches, native libraries)
- ...

https://jakearchibald.com/2017/h2-push-tougher-than-i-thought/
hhttps://daniel.haxx.se/blog/2016/08/18/http2-connection-coalescing/
https://tweakers.net/reviews/4555/3/tweakers-stapt-over-op-https-mixed-content.html
https://medium.com/bbc-design-engineering/http-2-is-easy-just-turn-it-on-34baad2d1fb1
Google QUIC ≠ QUIC + HTTP/3

- Been around for years
- Mainly Google and Akamai

- Being standardized by the IETF
- Same concepts, very different execution
A free PowerPoint Template made by Slidor.