WTA (IT302)

Web Protocols for Real-time Communication

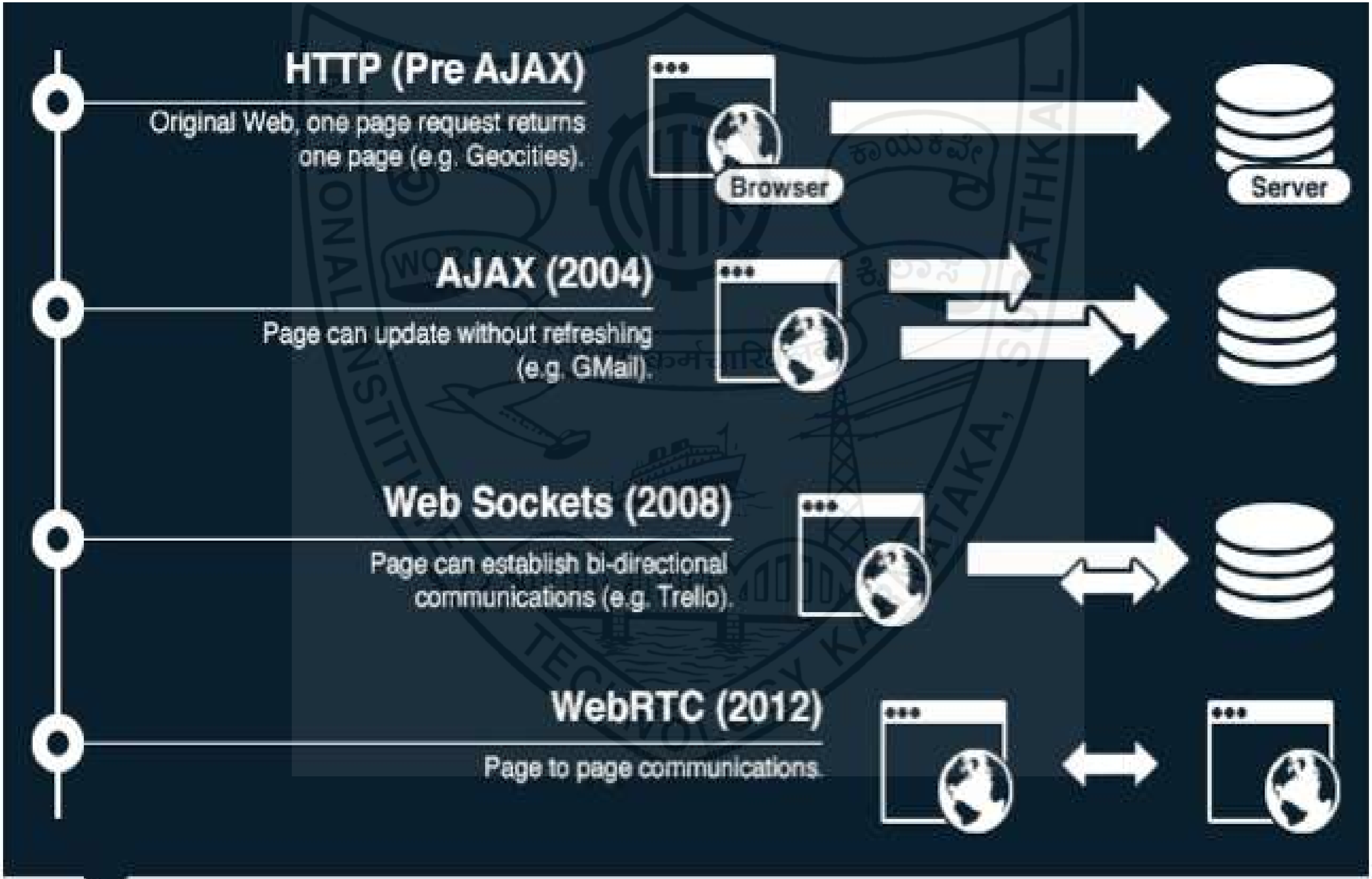
HTML5 WebSocket Protocol; WebRTC

New-age Web

- Evolution of web apps
  - Dynamic and real-time application
  - Webmail, Chat, word processing, etc.

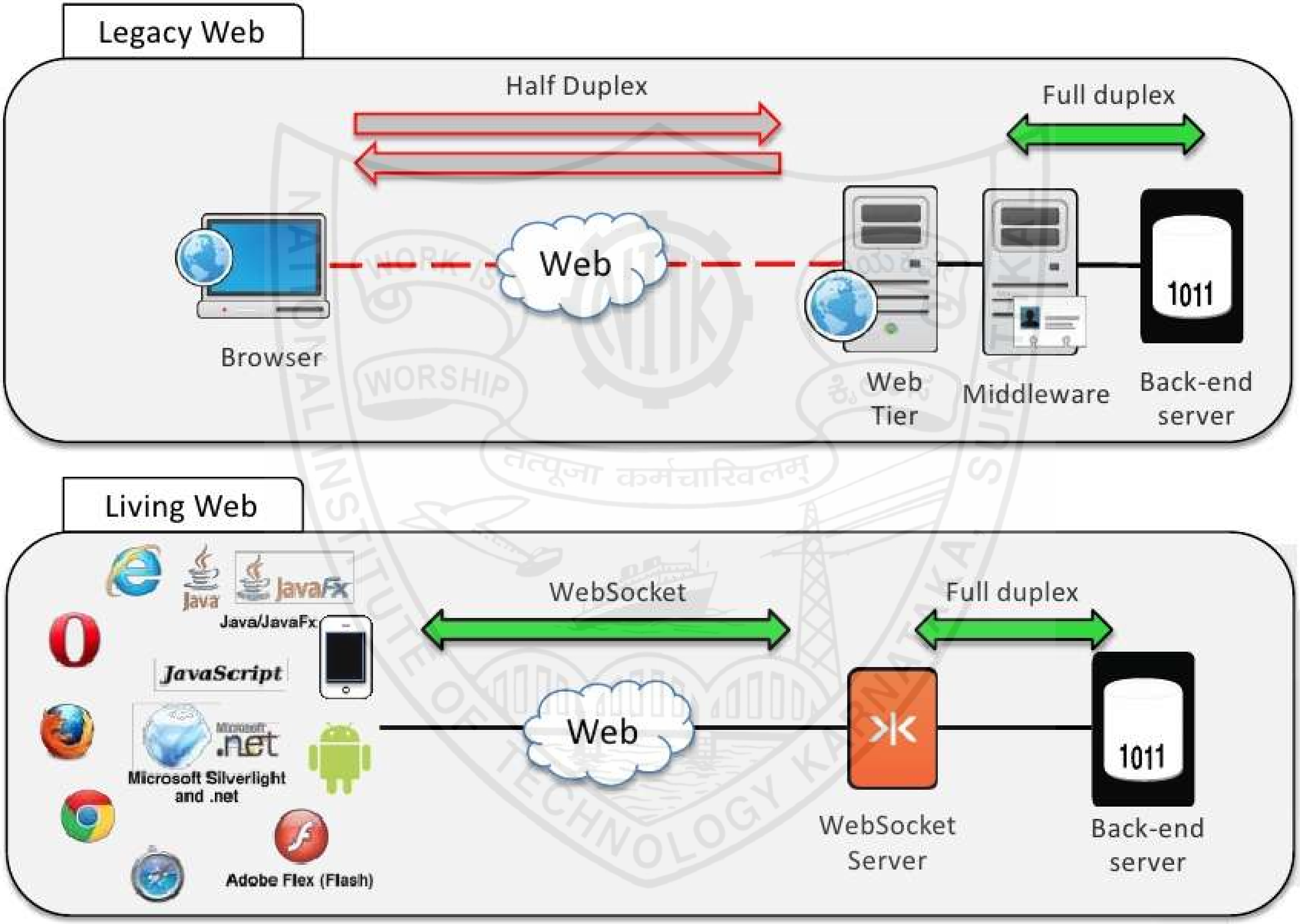
- HTTP is not designed for web apps
  - Large overhead
  - Hanging-GET is necessary for real-time server push

How did we get here?



WebSockets

- New protocol over TCP
  - Opening handshake
- HTTP-esque request and response
  - Newly defined WebSocket frame
- New API for JavaScript


Legacy Web
Half Duplex
Full duplex
Web
Browser
Web Tier
Middleware
Back-end server
1011
Living Web
WebSocket
Full duplex
Java
JavaFx
Java/JavaFx
JavaScript
Microsoft .net
Microsoft Silverlight and .net
Web
WebSocket Server
Back-end server
1011
Adobe Flex (Flash)

WebSockets

- Intended to replace hanging-GET based bidirectional channel
  - Two XMLHttpRequest → One WebSocket
- Full duplex
- Smaller overhead → Fewer TCP connection
- Simpler API

WebSockets - Requirements

- Coexist with HTTP on the same port
  - Use 80/443 which are rarely blocked
- Work with HTTP infrastructure
  - Proxy and firewall
- Allow cross origin connection
  - http://example.com/foo.js establishes WebSocket to ws://example.org/chat

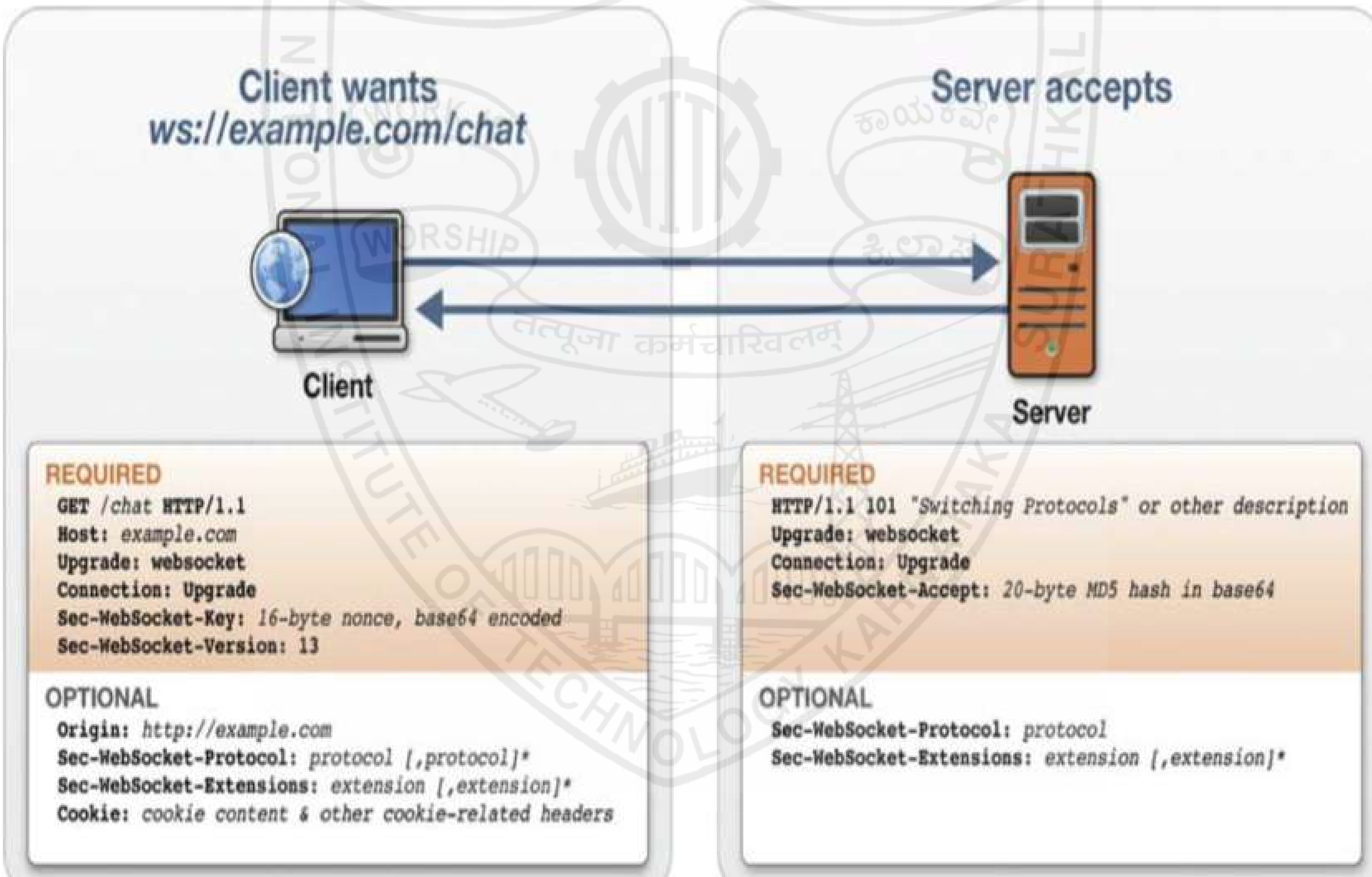
HTML5 WebSocket specification

- TCP based, bi-directional, full-duplex messaging
- Establish connection (Single TCP connection)
  - Send messages in both direction (Bi-directional)
  - Send message independent of each other (Full Duplex)
- End connection

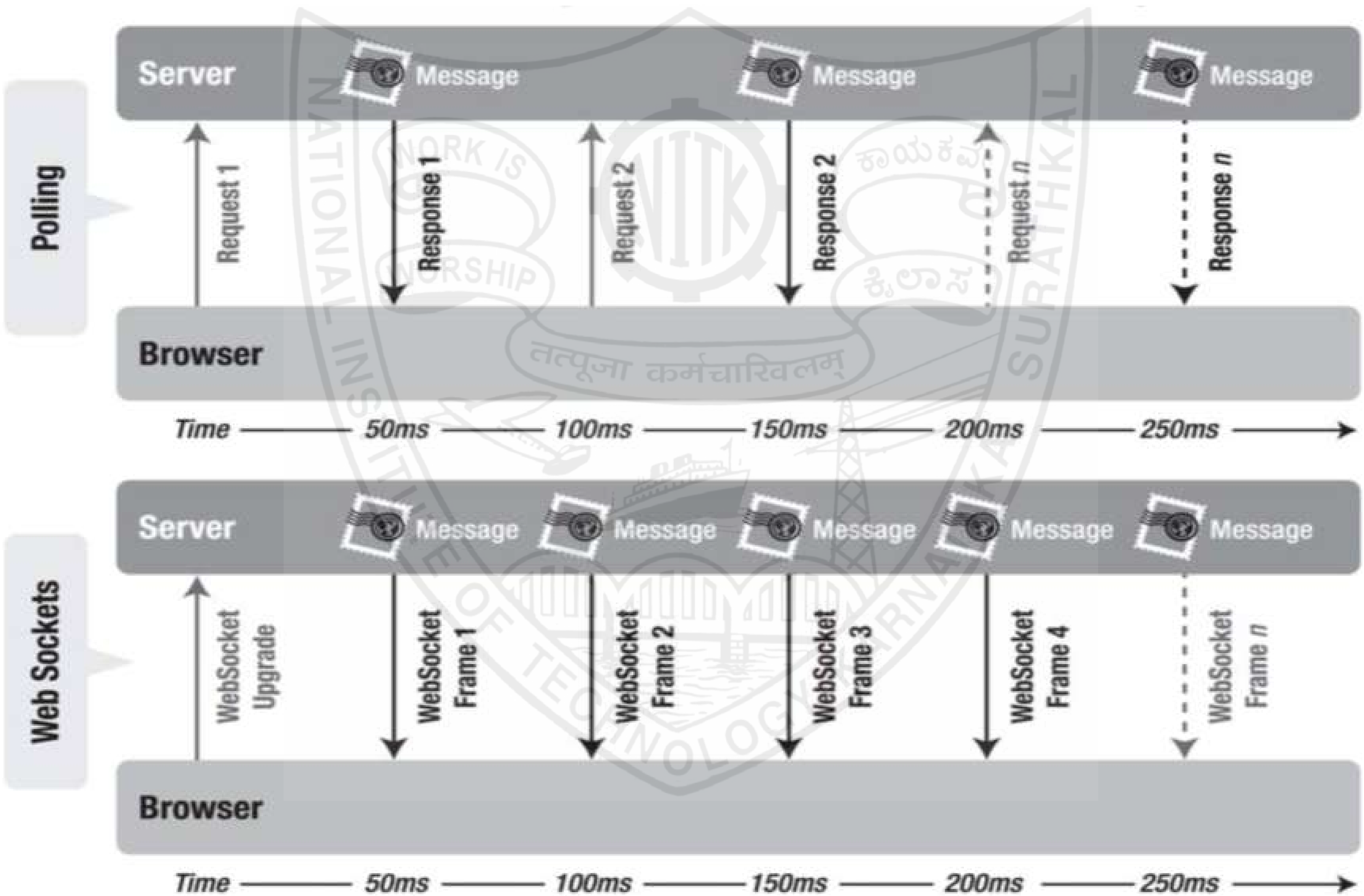
HTML5 WebSocket specification

- Useful for building true real-time functionality into Web applications.
  - E.g.
    - Online Chat
    - collaborative document editing
    - massively multiplayer online (MMO) games
    - stock trading applications etc.

The WebSocket Handshake



Latency comparison between the polling and WebSocket applications



Websocket small header size → efficiency

- Assume HTTP header is 871 bytes (some are 2000bytes)

Regular HTTP	HTTP with websocket
•Use case A: 1,000 clients polling every second: Network traffic is (871 · 1,000) = 871,000 bytes = 6,968,000 bits per second (6.6 Mbps)	•Use case A: 1,000 clients receive 1 message per second: Network traffic is (2 · 1,000) = 2,000 bytes = 16,000 bits per second (0.015 Mbps)
•Use case B: 10,000 clients polling every second: Network traffic is (871 · 10,000) = 8,710,000 bytes = 69,680,000 bits per second (66 Mbps)	•Use case B: 10,000 clients receive 1 message per second: Network traffic is (2 · 10,000) = 20,000 bytes = 160,000 bits per second (0.153 Mbps)
•Use case C: 100,000 clients polling every 1 second: Network traffic is (871 · 100,000) = 87,100,000 bytes = 696,800,000 bits per second (665 Mbps)	•Use case C: 100,000 clients receive 1 message per second: Network traffic is (2 · 100,000) = 200,000 bytes = 1,600,000 bits per second (1.526 Mbps)

WebSocket Protocol – Status codes

- Uses 4-digit codes in contrast to HTTP's 3-digit codes.

- Predefined codes
  - 1000 Normal closure
  - 1001 Peer is going away
  - 1002 Protocol error
  - 1003 Received unacceptable data
  - 1004 Too large message

More about WebSockets

- RFC 6455 - The WebSocket Protocol - IETF Tools
  - https://tools.ietf.org/html/rfc6455
- Introducing WebSockets: Bringing Sockets to the Web
  - http://www.html5rocks.com/en/tutorials/websockets/basics/
- WebSocket.org
  - https://www.websocket.org/
  - https://www.websocket.org/demos.html

Earlier Efforts

- Many web services already use concepts of RTC, but need downloads, native apps or plugins.
  - E.g. : Skype, Facebook (uses Skype) and Google Hangouts (uses Google Talk plugin).

- Issues -
  - Downloading, installing and updating plugins can be complex, error prone and annoying.
  - Plugins can be difficult to deploy, debug, troubleshoot, test and maintain—and may require licensing and integration with complex, expensive technology.

WebRTC

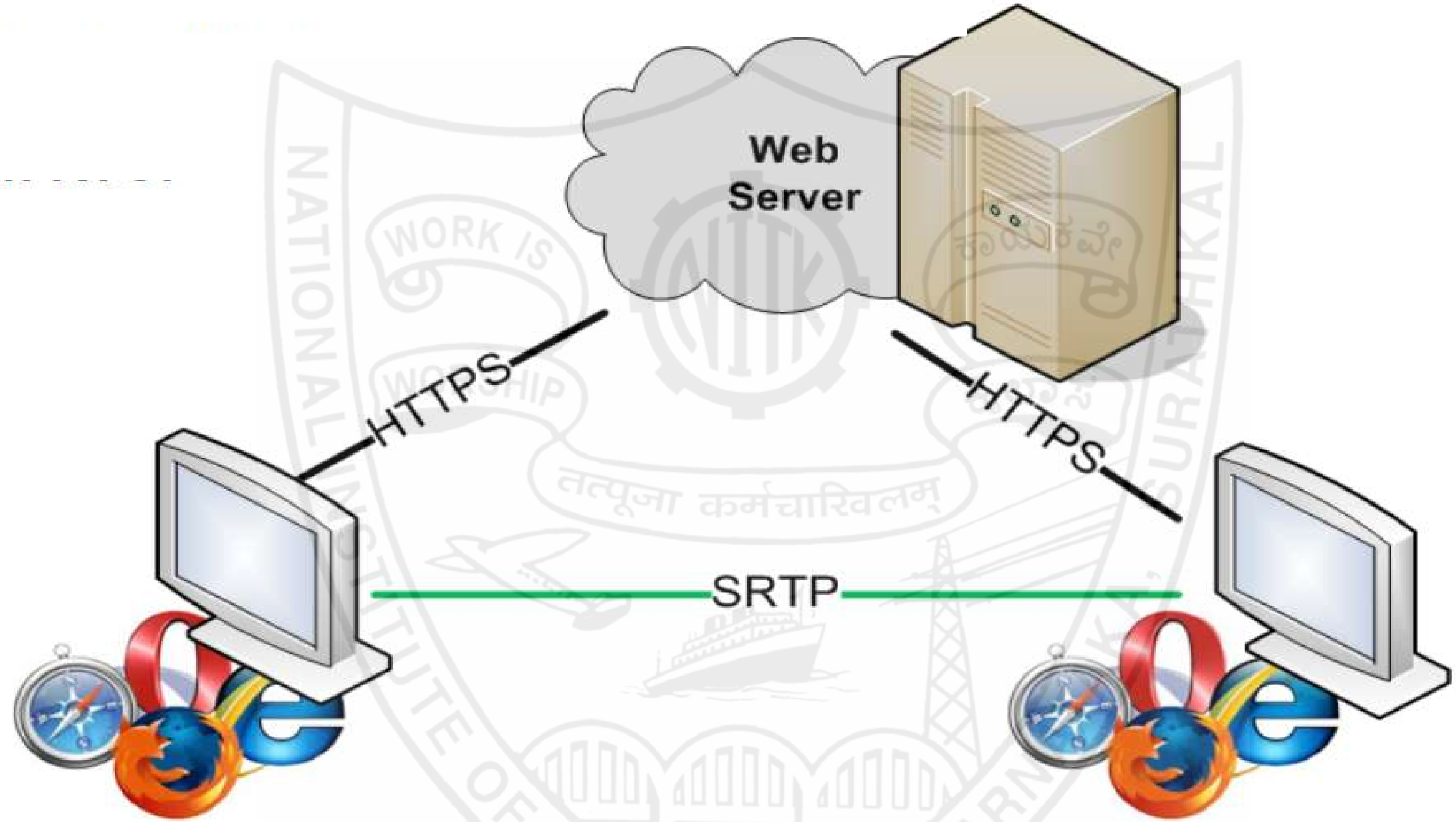
- Standards to enable browser based sessions (voice, video, Collab, …) without the need of custom clients or plugins
- Builds on HTLM5 capabilities with JavaScript
- Standardized by W3C and IETF
  - IETF RTCWeb WG (Internet world, IP protocols)
  - W3C WebRTC WG (web world, Browsers etc.)
- Intended for all browsers to support

WebRTC (contd.)

- Web Browsers with Real-Time-Communication
  - Audio/Video Chat on the web.
  - Integration via simple, standardized Web APIs.
  - Does not require plugins, downloads or installs.
  - No licenses or other fees.
  - Multiple browsers, multiple platforms.
  - No Security issues.
  - …
  - Just surf to the right address!

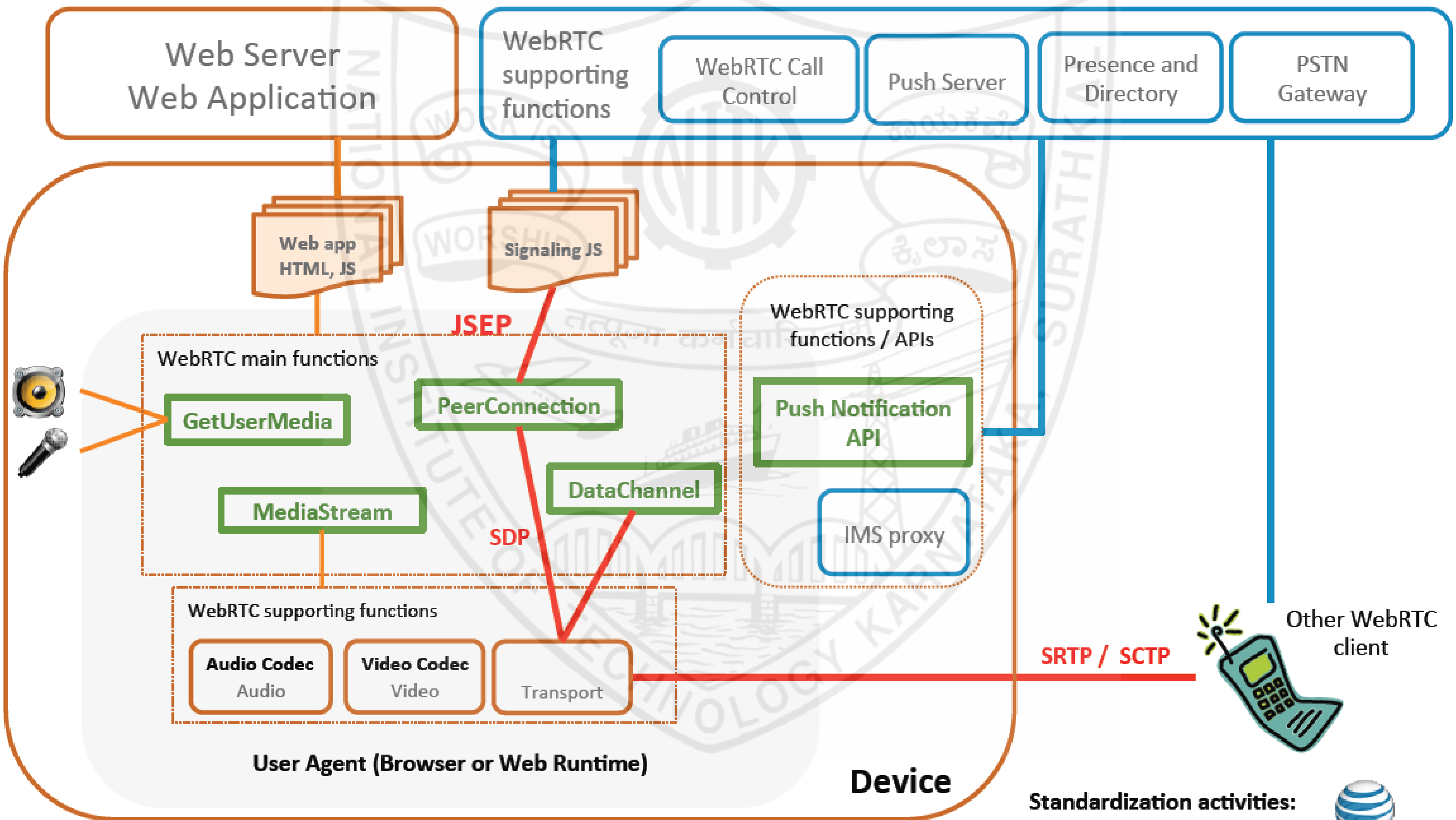
http://www.webrtc.org/faq

High Level Model



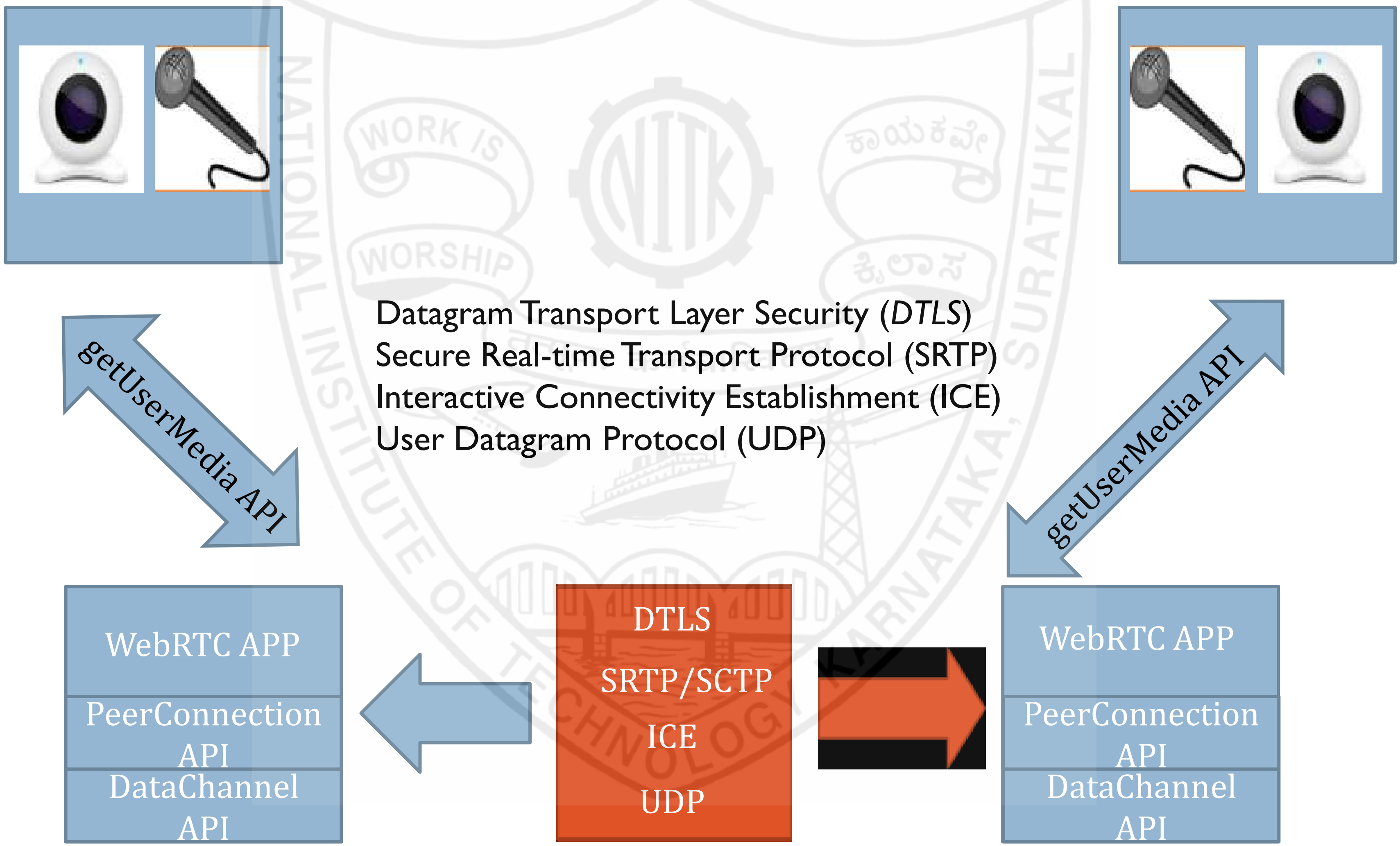
- Web Server/service based signaling brokering
  - Offer/Answer model
- Direct media flow, sometimes relayed due to NAT/NAPT
  - SRTP/RTCP

WebRTC Protocol Specification



Standardization activities:
W3C – Web API
IETF – Protocol

WebRTC API Stack View



Security in WebRTC

- Can compromise security requirements.
- For example:
  - Unencrypted media or data might be intercepted
    - browser ↔ browser, browser ↔ server.
  - Application might record and distribute video or audio without the user knowing.
  - Malware or viruses might be installed alongside an apparently innocuous plugin or application.

Security in WebRTC (contd.)

- WebRTC has several features to avoid these problems:
  - Use of secure protocols like DTLS and SRTP.
  - Encryption is mandatory for all WebRTC components.
  - WebRTC components run in the browser sandbox and not in a separate process
    - components do not require separate installation, and are updated whenever the browser is updated.
  - Camera and microphone access must be granted explicitly by user.
    - when the camera or microphone are running, this is clearly shown on the user interface.

Current Limitations

- Cloud Infrastructure
  - A server is required by WebRTC to complete all the required tasks: User discovery, Signalling and NAT/firewall traversal.
- Native Application support
  - not a software development kit that can be used in native iOS or Android applications or in native desktop applications.
- Multiparty Conferencing
  - inefficient when setting up communications between more than two end users.
- Recording
  - WebRTC does not support recording as of now.

More information…

- Salvatore Loreto, Simon Pietro Romano (2012) 'Real-Time Communications in the Web', IEEE paper October, 2012, IETF.org
- WebRTC book by Alan B. Johnston and Daniel C. Burnett : webrtcbook.com .
- Video of Justin Uberti's WebRTC session at Google I/O, 27 June 2012. webrtc.org
- Demos -
  - https://webrtc.github.io/samples/
  - https://www.webrtc-experiment.com/