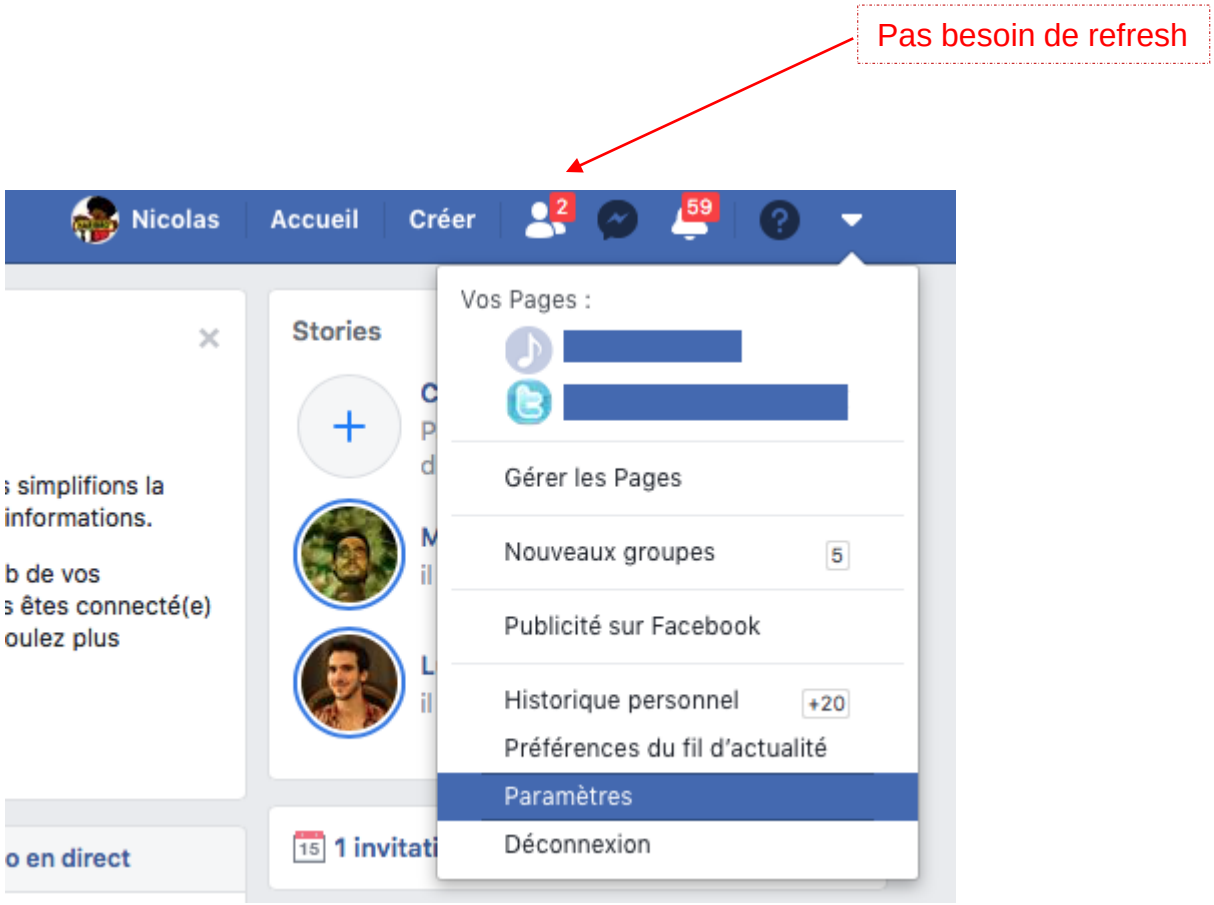
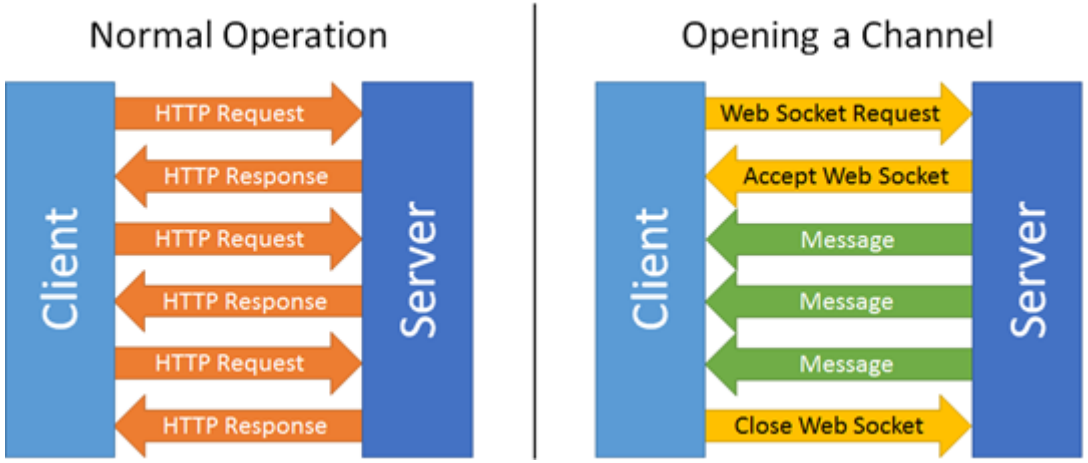
Une application temps réel avec Spring + Stomp + Websocket

Max Devulder

Temp réel ?

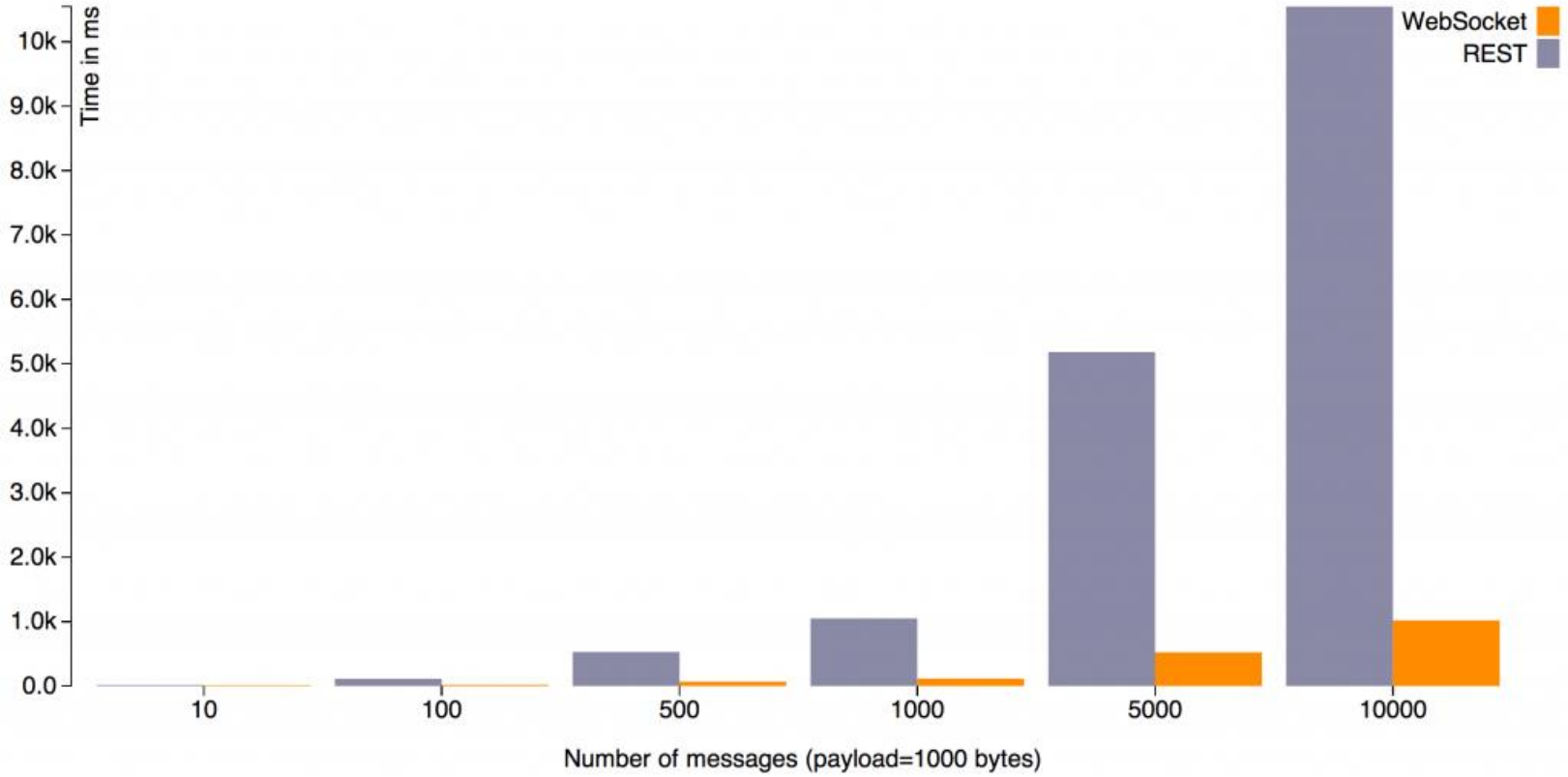

Pas besoin de refresh
Nicolas
Accueil
Créer
2
59
Stories
Vos Pages :
Gérer les Pages
Nouveaux groupes
5
Publicité sur Facebook
Historique personnel
+20
Préférences du fil d'actualité
Paramètres
Déconnexion
s simplifions la
informations.
b de vos
s êtes connecté(e)
oulez plus
o en direct
15
1 invitati

- **Protocole** ws://… (ou wss):
- Asynchrone, bidirectionnelle en mode full duplex
- Envoi de petits messages
- Websocket vs HTTP

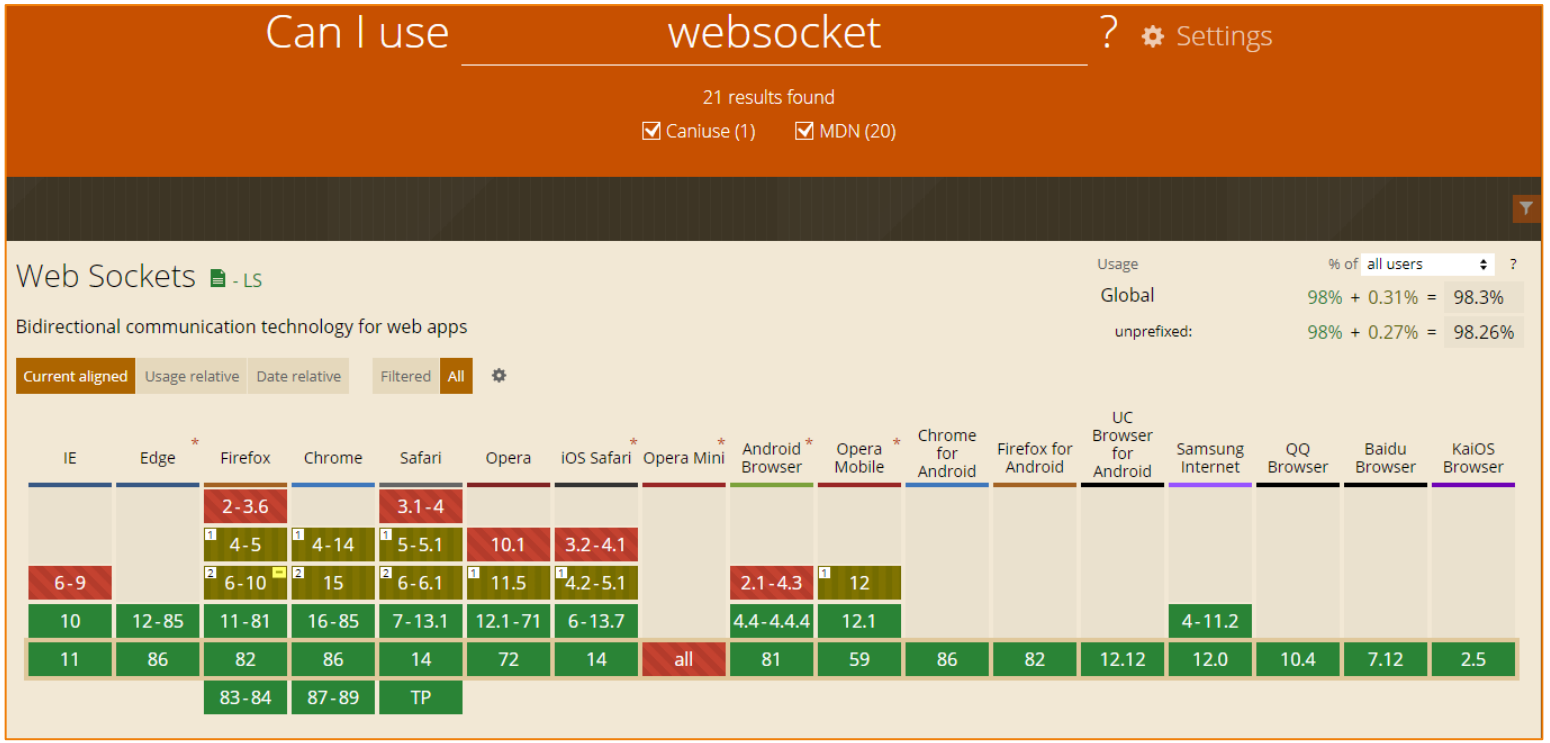


Performance

Source : https://browsee.io/blog/websocket-vs-http-calls-performance-study/



- Disponible nativement depuis 2011



- Installation avec spring-boot:

Ajouter la dépendance « spring-boot-starter-websocket » dans le **pom.xml**

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

Coté back, configuration Spring

Classe Spring

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

  public static final String ENDPOINT = "dlabs";
  private static final String APP = "/app";
  public static final String CONTEXT_MESSAGE = "/message";

  @Override
  public void configureMessageBroker(final MessageBrokerRegistry config) {
    config.enableSimpleBroker(CONTEXT_MESSAGE);
    config.setApplicationDestinationPrefixes(APP);
  }

  @Override
  public void registerStompEndpoints(final StompEndpointRegistry registry) {
    registry.addEndpoint(ENDPOINT).setAllowedOrigins("*");
    registry.addEndpoint(ENDPOINT).setAllowedOrigins("*").withSockJS();
  }
}
```

Coté back, envoi d'un message

Classe Spring injectée

```
@Service
public class WebSocketBO {

  @Autowired
  private SimpMessagingTemplate template;

  /**
   * Envoi à tous les utilisateurs front
   * @param message
   * @throws Exception
   */
  public void send(final WebSocketMessageDTO message) throws Exception {
    template.convertAndSend(WebSocketConfig.CONTEXT_MESSAGE + "/" + message.getType(), message);
  }

}
```

Coté front, réception d'un message

Exemple complet : https://spring.io/guides/gs/messaging-stomp-websocket

```
function connect() {
    var socket = new SockJS('/dlabs');
    stompClient = Stomp.over(socket);
    stompClient.connect({}, function (frame) {
        setConnected(true);
        console.log('Connected: ' + frame);
        stompClient.subscribe('/messages', function (greeting) {
            showGreeting(JSON.parse(greeting.body).content);
        });
    });
}
```

Le endpoint

Le context message,
exécuté à la réception d'un message