إعداد: محمد سلامه إشراف: م محمد بشار دسوقي

تم تنجيز المشروع الأول (Port Scanner):

من أجل بناء التطبيق بحيث يستخدم الـ Threads بأمثلية، يمكن اتباع عدة مناهج في الحل تختلف بحسب الرؤيا وإمكانية الجهاز.

الحالة الأولى:

حيث يمكن استخدام ExecutorService وحيد بحيث يمتلك عدد نياسب موافق لعدد IP المدخلة مضروباً بعدد المنافذ:

```
ExecutorService es = Executors.newFixedThreadPool(IPnum * Portsnum)
```

ومن ثم توزيع كل ثناية (IP:Port) إلى نيسب لتنفيذها وننتظر النتيجة.

هنا يكون الأمر سيئاً من ناحية تحصيل النتائج لأنها تكون مبعثرة وبحاجة إلى إعادة تجميع وإجراء بعض الحسابات الإحصائية (عدد المنافذ المفتوحة والمغلقة وغيره)

سبب آخر لعدم استخدامها هو في حال وجود عنوان IP لا يمكن الاتصال به لسبب ما، عندها دائما سوف نحصل على خطأ (Exception) ونعالجه (try – catch) وهذا يعد مكلفاً في الزمن الحقيقي وسوف يحصل هذا الأمر بعدد العناوين التي لا يمكن الاتصال بها مضروباً بعدد المنافذ المراد فحصها. (عدد كبير في حالات كثيرة).

الحالة الثانية:

يتم المرور عبر عناوين الـ IP المطلوبة بشكل تسلسلي، ومن أجل كل عنوان يتم معالجته بشكل خاص، حيث يتم إنشاء ExecutorService يحوي عدد نياسب مساو لعدد النياسب الممكن للنظام تقديمه ويتم فحص مجال المنافذ وإرجاع النتيجة.

في هذه الحالة عند وجود خطأ مع عنوان معين لا يتكرر ويحدث exception وحيد لكل عنوان لا يمكن الاتصال به وننتقل للعنوان التالي.

وأيضاً نضمن عمل البرنامج ضمن موارد نظام مثلى حيث لا نحجز عدد نياسب أكثر من الذي يمكن استخدامه على التوازي فعلياً وإرجاع النتائج يكون واضحاً ومباشراً دون الحاجة لتعديلات.

طبعاً يمكن حجز عدد نياسب أكبر نظراً لأن المهمة المنسوبة لكل نيسب هي بسيطة ولا تحتاج زمناً طويلاً وبالتالي يمكن تشغيل عدة نيساب بشكل concurrent دون حصول مشكلة، ولكن الأمر أيضاً يعود لمحدودية كرت الشبكة في معالجة الاتصالات لأن كل مهمة تقتضي محاولة فتح اتصال.