چکیده

در این پروژه به بررسی فرآیند پخش شایعه در شبکه‌های پیچیده پرداخته شده است. مدل مورد استفاده برای شبیه‌سازی پخش شایعه، مدل SIR است. شبکه‌های مورد آزمایش، شبکه‌های مقیاس آزاد، مقیاس آزاد و شبکه‌ای نمونه برداری شده از توییتر هستند. فرآیند شبیه‌سازی، در حالت‌های مختلفی انجام شده است. حالت اول بدون واکسینه کردن افراد و حالت‌های بعدی با واکسینه کردن افراد در شروع فرایند پخش شایعه شبیه‌سازی شدند. واکسینه کردن در دو حالت کلی فعال و غیرفعال انجام شده است. در حالت فعال، راس‌های واکسینه شده به طور فعال در واکسینه کردن بقیه راس‌ها نقش دارند، و در حالت غیرفعال راس‌های واکسینه شده، تعاملی با بقیه راس‌ها ندارند. برای واکسینه کردن راس‌ها در حالت غیرفعال، دو روش تصادفی، و مبتنی بر درجه در نظر گرفته شده است. طبق نتایج به دست آمده، واکسینه کردن راس‌ها با روش‌های مختلف، سرعت پخش شایعه را کندتر کرده، و درصد حداکثر افراد آلوده به شایعه را کاهش می‌دهد.

واژه‌های کلیدی: شبکه‌های پیچیده، پخش شایعه، مدل SIR، واکسینه کردن تصادفی، واکسینه کردن مبتنی بر درجه

فهرست عناوین صفحه

فهرست اشکال **صفحه**

۱. فصل اول

مقدمه

مقدمه

در دنیایی که زندگی می‌کنیم، شبکه‌ها همه جا حضور دارند: شبکه عصبی در داخل بدن ما، شبکه اینترنت، شبکه اجتماعی، تنها چند نمونه از شبکه‌هایی هستند که در دنیای اطراف ما وجود دارند. این شبکه‌ها به طور عمده ویژگی‌های خاصی دارند. از جمله این ویژگی‌ها این است که توزیع درجه در این شبکه‌ها به صورت توانی است. همچنین، در دسته‌ای از این شبکه‌ها، بین تمامی راس‌های شبکه، مسیر نسبتاً کوتاهی وجود دارد. به این نوع شبکه‌ها، شبکه‌های پیچیده [1]می‌گویند. از طرفی شبکه‌های اجتماعی اطراف ما نیز در این دسته‌ها از شبکه‌ها جای می‌گیرند. به همین دلیل، برای شبیه‌سازی از این شبکه‌ها استفاده کردیم. در فصل دوم، به طور مفصل در مورد شبکه‌های پیچیده توضیح داده می‌شود.

یکی از پدیده‌هایی که در شبکه‌های اجتماعی شکل می‌گیرد، پخش شایعه در این شبکه‌هاست. ابتدا چند فرد در شبکه در مورد شایعه‌ای صحبت می‌کنند. سپس نزدیکان آن‌ها نیز این شایعه‌ها را پخش می‌کنند، و به همین ترتیب این شایعه‌ها در مقیاس وسیعی پخش می‌شود. برای شبیه‌سازی این پدیده پخش شایعه، ما از مدل SIR که از جمله مدل‌های مبتنی بر پخش شایعه[2] است، استفاده کردیم. در این مدل افراد در سه حالت S، I و R قرار می‌گیرند. افرادی که در حالت S یا حساس[3] هستند، افرادی در شبکه هستند که در مورد شایعه چیزی نشنیده‌اند.

[1] Complex networks
[2] Gossip-based Protocols
[3] Suceptible

افرادی که در حالت I یا آلوده[4] هستند، در دسته‌ای قرار می‌گیرند، که شایعه را در جامعه پخش می‌کنند، و افرادی که در دسته R، یا بهبودیافته[5] قرار می‌گیرند، افرادی هستند که نسبت به شایعه مصون هستند، و در مورد کذب بودن آن خبر دارند. در مورد این مدل به تفصیل در فصل ۲ توضیح داده است.

برای کم کردن سرعت پخش شایعه و کم شدن تاثیر آن، می‌توانیم راس‌هایی را در ابتدای کار، افرادی را در حالت بهبودیافته قرار می‌دهیم و در واقع آن‌ها را مصون می‌کنیم. چنین افرادی را می‌توان در شبکه اجتماعی، افرادی در نظر گرفت که نسبت به غلط بودن شایعه آگاهی دارند، و آن را در شبکه پخش نمی‌کنند. در واکسینه کردن افراد نیز دو روش تصادفی، و مبتنی بر درجه در نظر گرفته شده است. در روش تصادفی راس‌ها برای واکسینه شدن به طور تصادفی انتخاب می‌شوند، و در روش مبتنی بر درجه، انتخاب راس‌های با درجه بیشتر برای واکسینه کردن، اولویت دارد. در فصل در فصل ۳ با جزئیات بیشتری در مورد مسئله مورد بررسی توضیح داده شده است. برای پیاده‌سازی مدل SIR در شبکه‌های پیچیده از شبیه‌ساز PeerSim کمک گرفته شده است. این شبیه‌ساز مبتنی بر جاوا، رابط[6]هایی را در اختیار ما می‌گذارد که با استفاده از آن‌ها شبیه‌سازی مدل‌های پخش شایعه در شبکه‌هایی با مقیاس بزرگ، قابل اجراست. برای پیاده‌سازی رابط کاربری نیز از زبان جاوا استفاده شده است. جزئیات پیاده‌سازی در فصل ۴ آمده است.

در فصل ۵ نتایج حاصل از آزمایش‌های انجام شده بررسی شده‌اند و مورد ارزیابی قرار گرفته‌اند. در این فصل مشخص می‌شود که کدام یک از روش‌های واکسینه کردن راس‌ها، تاثیر بیشتری در کم کردن سرعت پخش شایعه دارند.

[4] Infected
[5] Recovered
[6] interface

۲

فصل دوم

پیش نیاز

پیش‌نیاز

در این فصل به بررسی مفاهیمی که در این پروژه از آن‌ها استفاده شده است، می‌پردازیم. اصلی‌ترین این مفاهیم، شبکه‌های پیچیده و مدلی از پخش شایعه هستند که در این پروژه استفاده شده است.

۲,۱ شبکه‌های پیچیده

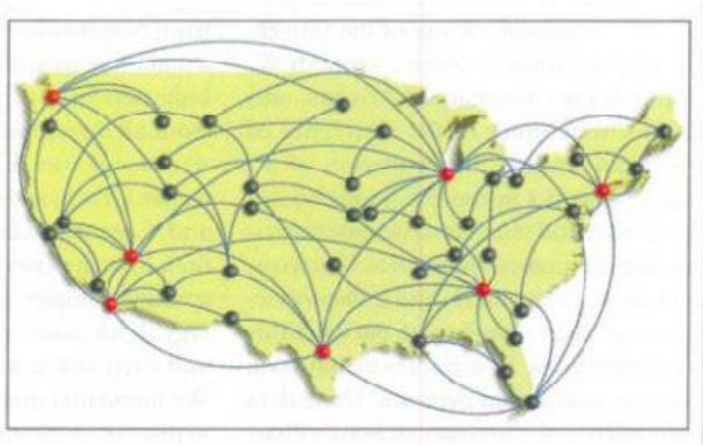
ما در زندگی روزمره با شبکه‌های بسیاری سر و کار داریم. مغز انسان شبکه‌ای از ملیون‌ها سلول‌های عصبی است که با آکسون‌ها با هم مرتبط می‌شوند. جوامعی که هم که در آن‌ها زندگی می‌کنیم شبکه‌ای از افراد هستند که توسط روابط خانوادگی، دوستی و شغلی با هم مرتبط می‌شوند. در دنیای تکنولوژی نیز شبکه‌ها حضور پررنگی دارند. اینترنت، شبکه‌های برق‌رسانی و سامانه‌های حمل و نقل، مثال‌هایی از این نوع شبکه‌ها هستند.

تحقیقات سال‌های اخیر، نشان داده‌اند که بسیاری از این شبکه‌ها، از جمله اینترنت، شبکه‌های اجتماعی و شبکه‌های سلولی، ویژگی‌های مشترکی دارند. این شبکه‌ها در شاخص‌هایی از قبیل توزیع درجه، ضریب خوشه‌بندی و خاصیت شرکت پذیری دارای ویژگی‌های مشترکی هستند. همچنین ساختار آن‌ها، ساختاری بین شبکه‌های تصادفی و منتظم است. به چنین شبکه‌هایی، شبکه‌های پیچیده گفته می‌شود. از جمله مهمترین دسته‌های شبکه‌های پیچیده، شبکه‌های مقیاس آزاد[۷] و شبکه‌های دنیای کوچک[۸] هستند.

[7] مقیاس آزاد Network
[8] Small world Networks

۲٫۱٫۱ شبکه‌های مقیاس آزاد

در این شبکه‌ها درصد کمی از راس‌ها وجود دارند که به اکثر راس‌های موجود در شبکه متصل هستند. به چنین راس‌هایی قطب[9] گفته می‌شود. از جمله خاصیت‌های چنین شبکه‌هایی این است که در برابر خرابی‌های تصادفی به طور قابل ملاحظه‌ای مقاوم هستند. از طرف دیگر، این شبکه‌ها در برابر حمله‌های استراتژیک به شدت آسیب‌پذیرند.

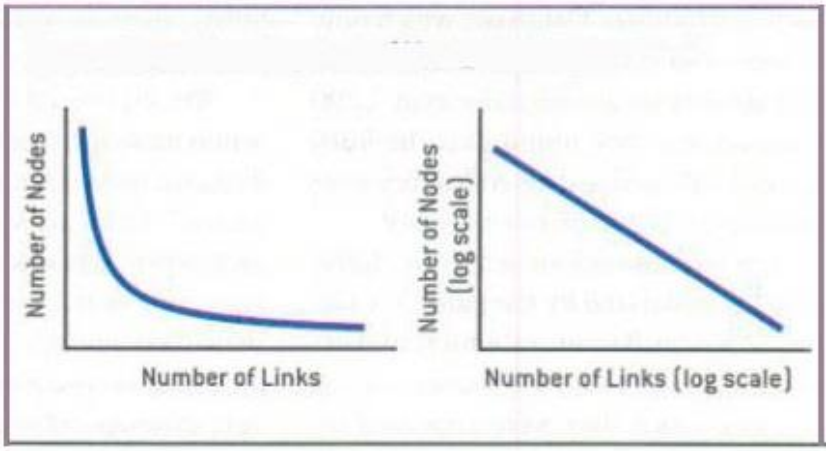


شکل ۱. نمونه‌ای از یک شبکه مقیاس آزاد

۲٫۱٫۱٫۱ ویژگی‌های اصلی

از جمله ویژگی‌های بارز این شبکه‌ها این است که توزیع درجه در آن‌ها از قاعده‌ی توزیع توانی پیروی می‌کنند. این قاعده بیان می‌کند که احتمال این که یک راس به k راس دیگر متصل باشد برابر است با $1/k^n$. مقدار n در این شبکه‌ها عددی بین ۲ و ۳ محاسبه شده است. توزیع توانی، بر خلاف توزیع نرمال، نقطه اوج ندارد. این توزیع در واقع یک تابع نزولی پیوسته است. توزیع توانی وقتی با مقیاس دوگانه‌ی لگاریتمی نمایش داده می‌شود، به یک خط راست تبدیل می‌شود. چنین توزیعی بیان می‌کند، که تعداد بسیار کمی از راس‌ها هستند که بیشترین درجه را در شبکه دارند. برای مثال، در شبکه‌ی اینترنت، گوگل و یاهو چنین راس‌هایی هستند که همان‌طور که قبلا گفته شد به آن‌ها قطب گفته می‌شود. در ادامه به مثال‌هایی از شبکه‌های مقیاس آزاد اشاره می‌کنیم.

[9] Hub nodes



شکل ۲. نمودارهای توزیع توانی درجه در شبکه‌های مقیاس آزاد

۲،۱،۱،۲ نمونه‌های شبکه‌های مقیاس آزاد

در بررسی که توسط Michalis Faloutsos از دانشگاه کالیفرنیا در ریورساید و همکارانش به روی روترهای اینترنت انجام داده‌اند، به این نتیجه رسیدند که شبکه‌ی روترهای اینترنت که با به طور نوری، یا توسط سایر اتصالات سیمی به هم وصل می‌شوند، این شبکه ساختار مقیاس آزاد دارد. محققان همچنین نشان داده‌اند که بعضی از شبکه‌های اجتماعی مقیاس آزاد هستند. برای مثال، در تحقیقی که دانشمندان دانشگاه‌های بستون و استوکهولم با همکاری یکدیگر انجام دادند، نشان داده شده است که شبکه‌ی روابط جنسی افراد در کشور سوئد از یک قاعده‌ی توانی پیروی می‌کند، با وجود این که بیشتر افراد در طول عمرشان شرکای جنسی محدودی داشتند، تعداد کمی از این افراد که در واقع راس‌های قطب هستند، صدها شریک جنسی داشته‌اند. در تحقیقی دیگر که توسط Stefan Bornholdt در دانشگاه kiel در آلمان انجام شده است، این نتیجه به دست آمده است که شبکه‌ی افرادی که توسط پست الکترونیکی با یکدیگر در ارتباطند، ساختار مقیاس آزاد دارد. Sidney Redner از دانشگاه بوستون، نشان داده است که شبکه‌ی مقالات علمی، که توسط ارجاعات به یکدیگر متصل شده‌اند، از قاعده‌ی توزیع توانی پیروی می‌کند. Mark Newman از دانشگاه میشیگان در بررسی که به روی همکاری علمی محققان در رشته‌های فیزیک و علوم کامپیوتر، انجام داده است، به این تنیجه دست یافته است که شبکه‌ای که از همکاری این محققان وجود دارد، ساختار مقیاس آزاد دارد. همچنین در تحقیق دیگری که توسط Alberto Barabasi انجام شده، همچنین این نتیجه به دست آمده که شبکه‌ی حاصل از همکاری دانشمندان حوزه‌ی ریاضی و عصب‌شناسی، نیز چنین ساختاری دارد. یکی از نکات جالب به دست آمده، این است که اردوش، ریاضی‌دان معروف، یکی از بزرگترین راس‌های قطب در ریاضی است، که بیش از هزار و چهارصد مقاله را منتشر کرده است که در آن‌ها با بیش از پانصد ریاضی‌دان همکاری داشته است.

شبکه‌های مقیاس آزاد در حوزه‌های اقتصادی نیز وجود دارند. Walter W. Powell از دانشگاه استنفورد به همراه همکارانش، در تحقیقی به مطالعه ساختار شبکه‌های شرکت‌های بیوتکنولوژی در کشور آمریکا پرداختند، و تعدادی از راس‌های قطب را پیدا کردند، برای مثال شرکت‌هایی مثل Genzyme، Chiron و Genatech با تعداد بسیار زیادی از شرکت‌های دیگر در این حوزه همکاری دارند.

حتی شبکه‌ی بازیگران در هالیوود هم ساختار مقیاس آزاد دارد. طبق بررسی‌های انجام شده، اکثر بازیگران حاضر، تنها با تعداد محدودی از بازیگران مرتبط هستند، ولی تعداد کمی از آن‌ها از جمله Rod Steiger و Donald Pleasence هزاران ارتباط دارند.

شبکه‌های مقیاس آزاد همچنین در دنیای زیستی هم حاضرند.Alberto Barabasi با همکاری یک زیست‌شناس سلولی به نام Zoltan Oltvai در تحقیقات خود به این نتیجه رسیده‌اند که ساختار سلولی حدود ۴۳ نوع مختلف از ارگانیسم‌های موجودات زنده، شبکه‌ای مقیاس آزاد را تشکیل می‌دهند. در چنین شبکه‌هایی، سلول‌ها برای سوزاندن غذا، مولکول‌های پیچیده را جدا می‌کنند، که در طی این فرآیند انرژی آزاد می‌شود. هر راس در این شبکه، یک مولکول است و هر یال، یک واکنش زیست‌شیمیایی است. این محققان به این نتیجه دست یافته‌اند که، بیشتر مولکول‌ها در یک یا دو واکنش شرکت می‌کنند، در حالی که تعداد کمی از مولکول‌ها مانند آب و آدنوزین تری فسفات، که در واقع همان راس‌های قطب هستند، در بسیاری از واکنش‌ها شرکت دارند.

این محققان همچنین به این نکته دست یافتند که شبکه‌ای که از تعامل پروتئینی سلول‌ها به وجود می‌ایند نیز مقیاس آزاد است. در چنین شبکه‌ای که پروتئین‌ها راس‌های آن هستند، دو پروتئین در صورتی به هم یال دارند که با یکدیگر تعامل داشته باشند.این محققان در طی تحقیقات خود مخمر شیرینی‌پزی را که یکی از ساده‌ترین ساختارهای سلولی را دارد شامل هزاران پروتئین می‌شود، بررسی کردند. آنها دریافتند که بیشتر پروتئین‌ها با یک یا دو پروتئین دیگر تعامل دارند، در حالی که تعداد کمی از آن‌ها وجود دارند که تعاملاتی با بسیاری از پروتئین‌های دیگر دارند، که این نشان می‌دهد این شبکه ساختار مقیاس آزاد دارند. همچنین در بررسی دیگری که به روی هلیپوباکتر انجام شده است، این محققان نشان دادند که پروتئین‌های موجود در این ماده نیز شبکه مقیاس آزاد را تشکیل می‌دهند.

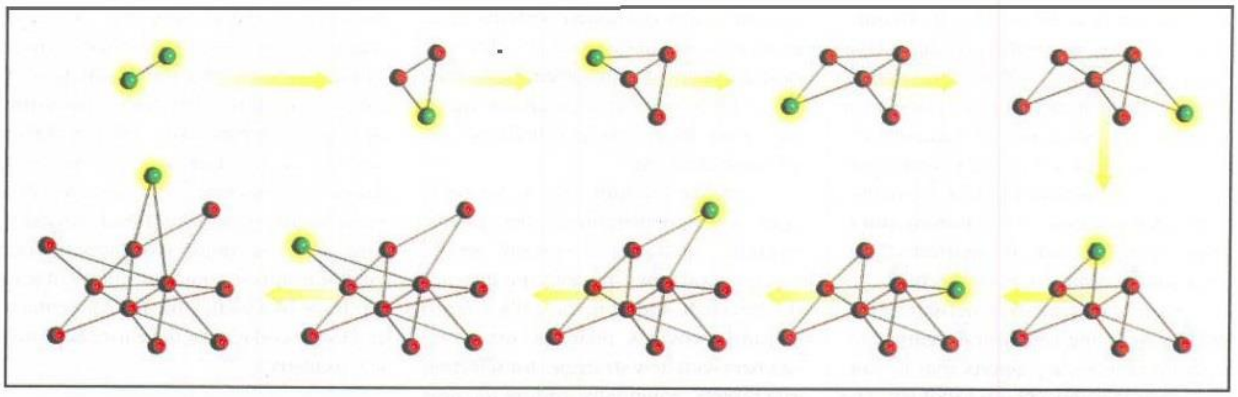
در نهایت، بررسی‌های متعدد انجام شده توسط محققان، این نتیجه را داشته است که بسیاری از شبکه‌هایی که در دنیای واقعی وجود دارند ساختار مقیاس آزاد دارند. در اینجا سوال مهمی که مطرح می‌شود این است که چطور ممکن است که طیف وسیعی از شبکه‌ها ساختار مشابهی داشته باشند و از قواعد یکسانی پیروی کنند؟ در بخش بعد به بررسی این سوال می‌پردازیم.

۲،۱،۱،۳ وجود داشتن راس‌های قطب

برای توضیح بهتر وجود راس‌های قطب، به توضیح نحوه گسترش بسیاری از شبکه‌ها در طول زمان می‌پردازیم. در بسیاری از شبکه‌ها راس‌ها ثابت نیستند. برای مثال شبکه صحفه‌های وب موجود در اینترنت، در سال ۱۹۹۰ تنها شامل یک صفحه می‌شد، ولی در حال حاضر، بیش از سه ملیارد صفحه موجود هستند. بسیاری از شبکه‌ها به طور مشابهی گسترش پیدا می‌کنند. برای مثال، هالیوود در سال ۱۸۹۰ تعداد کمی بازیگر داشت. و به مرور زمان که افراد بسیاری وارد هالیوود شدند، به طوری که در حال حاضر بیش از نیم میلیون نفر در آن حضور دارند. لازم به ذکر است که در ساختار شبکه‌ای هالیوود بازیگران باتجربه و تازه وارد، به بازیگران باتجربه متصل هستند.

در مثالی دیگر به اینترنت اشاره می‌کنیم. اینترنت هم در حدود سی سال پیش، تعداد کمی روتر داشت. اما شبکه اینترنت به مرور زمان گسترش یافت به طوری که اکنون میلیون‌ها روتر در این شبکه وجود دارند. در اینترنت هم، روترهای جدید، به روترهای قدیمی متصل می‌شوند.

در مجموع می‌توان گفت که در شبکه‌های دنیای واقعی، که همیشه راس‌ها در حال اضافه شدن به ساختار شبکه هستند، و راس‌های جدید با احتمال بیشتری به راس‌هایی که طول عمر بیشتری در شبکه دارند، وصل می‌شوند. در این شبکه‌ها، راس‌ها برابر نیستند. برای مثال در اینترنت، وقتی فردی می‌خواهد وب سایت جدیدی را ایجاد کند، در بیشتر موارد آن را به وب‌سایت‌های شناخته شده که تعدادشان هم محود است، پیوند می‌دهد. برای همین این وب‌سایت‌ها پیوندهای زیادی به وب‌های دیگر پیدا می‌کنند. افراد با انتخاب وب‌سایت‌های شناخته شده باعث می‌شوند که این راس‌ها در شبکه به قطب تبدیل شوند. این فرایند انتخاب در بقیه شبکه‌ها هم اتفاق می‌افتد. در هالیوود هر چه بازیگران در فیلم‌های بیشتری بازی کرده باشند، احتمال این که برای نقش‌های جدید انتخاب شوند بیشتر می‌شود. در اینترنت هرچه روترها اتصالات بیشتری داشته باشند، پهنای باند بیشتری را به خود اختصاص می‌دهند، در نتیجه کاربران جدید آن‌ها را برای اتصال ترجیح می‌دهند. در حوزه مقالات علمی نیز، هر چه مقالات از ارجاعات بیشتری برخوردار باشند، محققان بیشتری تشویق به مطالعه و ارجاع دادن به آن‌ها می‌شوند. دو عاملی که تا به حال در مورد آنها توضیح دادیم، یعنی رشد شبکه‌ها و امکان انتخاب راس‌ها برای برقراری پیوند، دلایل مناسبی برای توضیح وجود داشتن راس‌های قطب هستند. هنگامی که راس‌های جدیدی وارد شبکه می‌شوند، تمایل دارند که به راس‌هایی که یال‌های بیشتری دارند، متصل شوند. در نتیجه یال‌های این راس‌ها به مرور زمان بیشتر از یال‌های راس‌های دیگر می‌شود. در واقع هر چه راسی در زمان زودتری در شبکه وجود داشته باشد، احتمال آن که به یک راس قطب تبدیل شود، بیشتر می‌شود.



شکل ۳ . مراحل رشد یک شبکه مقیاس آزاد در اثر وارد شدن راس‌های جدید به شبکه

۲,۱,۱,۴ نقطه ضعف شبکه‌های مقیاس آزاد

با توجه به این که شبکه‌های مقیاس آزاد در بخش‌های مختلف زندگی ما حضور دارند، سوالی که مطرح می‌شود این است که شبکه‌ها تا چه اندازه قابل اطمینان هستند؟ خوشبختانه این شبکه‌ها در برابر بروز خطاهای تصادفی به شدت مقاوم هستند. برای مثال، در شبکه اینترنت با این که صدها روتر به در هر لحظه دچار ایراد فنی می‌شوند، شبکه اینترنت عملکرد خود را حفظ می‌کند و آسیب جدی به آن وارد نمی‌شود. در ادامه به پاسخ این سوال می‌پردازیم که منشا چنین مقاومتی در برابر خطاهای تصادفی چیست.

از لحاظ شهودی، هنگامی که چند راس در شبکه دچار مشکل می‌شوند، کارکرد کل شبکه باید مختل شود، چرا که شبکه بر اثر از دست دادن چند راس به بخش‌های مختلف تقسیم می‌شود. چنین چیزی در شبکه‌های تصادفی صحیح است. اگر درصد مهمی از راس‌های شبکه حذف شوند، کل سیستم به چند زیرشبکه غیرهمبند تقسیم می‌شود. ولی چنین چیزی در شبکه‌های مقیاس آزاد صحیح نیست. برای مثال در یک شبیه‌سازی نشان داده شد که در صورتی که حتی حدود ۸۰ درصد از روترهای اینترنت به طور تصادفی حذف شوند، روترهای باقیمانده همچنان شبکه‌ای همبند را تشکیل می‌دهند، به طوری که بین هر دو راس باقیمانده‌ای مسیری وجود دارد.

به طور کلی، شبکه‌های مقیاس آزاد به طرز شگفت‌اگیزی در برابر خرابی‌های تصادفی مقاوم هستند. دلیل این خاصیت هم ساختار ناهمگون آن‌ها یا همان توزیع درجه‌ای است که در این شبکه‌ها وجود دارد. در واقع، حذف تصادفی راس‌ها، معمول شامل راس‌هایی با درجه کمتر می‌شود، زیرا تعداد آن‌ها بسیار بیشتر از راس‌های قطب است. از طرفی حذف شدن راس‌هایی با درجه پایین، ساختار شبکه را به هم نمی‌زند، چون آن‌ها یال‌های کمتری در قیاس با راس‌های قطب دارند. اما، وابستگی شبکه‌های مقیاس آزاد به راس‌های قطب، یک مشکل اساسی به وجود می‌آورد و آن هم حساس بودن این شبکه‌ها به حمله‌های از پیش تعیین شده است. در چنین حمله‌هایی راس‌های خاصی از جمله راس‌های قطب مورد هدف قرار گرفته می‌شوند. شبیه‌سازی‌های انجام شده توسط Alberto Barabasi و همکارانش، نشان می‌دهند که حذف کردن تنها چند راس، از راس‌های قطب که نقش

کلیدی را در شبکه اینترنت ایفا می‌کنند، سیستم را به زیرشبکه‌های کوچک غیرهمبند که به قسمت‌های دیگر شبکه مسیری ندارند، تقسیم می‌کند.

۲,۱,۲ شبکه‌های دنیای کوچک

دسته‌ای از شبکه‌های دنیای واقعی هستند که ساختاری بین شبکه‌های تصادفی و منتظم دارند. در این شبکه‌ها، بین هر دو راس دلخواه در شبکه مسیر کوتاهی وجود دارد، که این ویژگی آن‌ها، مشابه شبکه‌های تصادفی است. از طرف دیگر در این شبکه‌ها ضریب خوشه‌بندی بالا است. یعنی اگر بین دو راس در شبکه، به راس دیگری یال داشته باشند، احتمال این که بین آن دو راس یالی وجود داشته باشد، بیشتر می‌شود. مثلاً در یک شبکه اجتماعی اگر دو فرد دوست مشترکی داشته باشند، احتمال این که با هم دوست باشند، به نسبت زیاد است. این ویژگی شبکه‌های دنیای کوچک، به شبکه‌های منتظم شبیه است.

۲,۲ پروتکل‌های مبتنی بر پخش شایعه

پروتکل‌های پخش شایعه، که به آن‌ها پروتکل‌های اپیدمیک هم گفته می‌شود، در شبکه‌های بسیار گسترده‌ای که در دنیای واقعی با آن‌ها مواجهیم، کاربرد زیادی دارند. انتقال اطلاعات بین راس‌های شبکه‌ای با مقیاس بزرگ و ساختار پویا، که در آن راس‌ها مدام در حال حذف و اضافه شدن هستند، با استفاده از این پروتکل‌ها امکان‌پذیر است. در چنین پروتکلی، هر راس، با راس‌های مجاور خود به طور مداوم در حال تبادل اطلاعات است. در ادامه به بررسی مدل خاصی که برای پخش شایعه در شبکه در نظر گرفتیم می‌پردازیم.

۲,۲,۱ معرفی مدل SIR

مدلی که ما برای پخش شایعه در نظر گرفتیم، مدل SIR است. این مدل بیشتر برای شبیه‌سازی پخش بیماری در شبکه‌ها در نظر گرفته می‌شود. اما به دلیل شباهت‌های کاربردی که در ادامه به توضیح آن‌ها می‌پردازیم، ما این مدل را برای پخش شایعه انتخاب کردیم.

برای درک بهتر مدل SIR توضیح مختصری در مورد بیماری‌های واگیردار می‌دهیم. بیماری‌های واگیردار به دو دسته اصلی حاد[10] و مزمن[11] تقسیم می‌شوند. در بیماری‌های حاد، سرعت پخش بیماری زیاد است. از طرفی با استفاده از داروهای موثر، پس از گذشت چند روز تا چند هفته، عامل بیماری در فرد مبتلا از بین می‌رود. بیماری‌هایی نظیر آنفولانزا، هاری و آبله مرغان در این دسته قرار می‌گیرند. از طرف دیگر، افراد مبتلا به بیماری‌های

[10] Acute
[11] Chronic

مزمن، بین چند ماه تا چند سال دچار بیماری هستند. بیماری‌هایی مثل تبخال و کلامیدیا در این دسته قرار می‌گیرند. در مدل SIR بیماری‌های حاد شبیه‌سازی می‌شوند، با این فرض که فرد مبتلا، بعد از گذشت دوره‌ای کوتاه، از بیماری مصون می‌شود و این مصونیت تا آخر عمر او ادامه دارد. برای مدل‌سازی چنین بیماری‌های مدل SIR مدل مناسبی است. که در آن افرادی که هنوز در معرض بیماری قرار نگرفته‌اند با S (افراد حساس به بیماری[۱۲])، افراد بیمار با I (کسانی که در حال حاضر در معرض بیماری قرار دارند[۱۳]) و افراد بهبودیافته[۱۴] (افرادی که نسبت به بیماری مصون هستند).

در این مدل، افراد از سه حالت مختلف تغییر وضعیت می‌دهند. به این صورت که آن‌ها از حالت S به حالت I، و از حالت I به حالت R منتقل می‌شوند. در این مدل فرض می‌شود، بعد از طی شدن مدت زمان ثابت، فرد از حالت بیمار به حالت بهبودیافته تغییر وضعیت دهد، و این به دلیل ماهیت بیماری‌های واگیردار حاد است. از طرف دیگر، برای انتقال بیماری از فرد بیمار به فرد حساس به بیماری، چند عامل تاثیرگذار هستند. عامل اول فراوانی افرادی است که دچار بیماری هستند، عامل دوم بعد ساختار شبکه افراد و عامل آخر احتمال انتقال بیماری در صورت تماس فرد بیمار با فرد حساس به بیماری است.

۲,۲,۲ دلیل انتخاب مدل SIR برای پخش شایعه

قبل از این که به توضیح بیشتر در مورد این مدل بپردازیم، دلیل انتخاب آن را برای پخش شایعه بیان می‌کنیم. در جامعه‌ای از افراد وقتی شایعه‌ای وارد جامعه می‌شود، ساختاری مانند بیماری واگیردار دارد. فردی که شایعه را باور کرده است، با دوستان خود در مورد آن صحبت می‌کند، دوستان این فرد هم ممکن است، حرف او را باور کنند و خود به ترویج شایعه بپردازند. از طرفی وقتی فردی دلایل عقلی و منطقی برای رد یک شایعه داشته باشد، آن شایعه را باور نمی‌کند، حتی اگر با افرادی که به شایعه باور دارند صحبت کند. علاوه بر این، هر شایعه بعد از گذشت مدت زمانی، رونق خود را از دست می‌دهد. در واقع افرادی که آن شایعه را باور کرده‌اند به مرور زمان دروغ بودن شایعه بر آن‌ها روشن می‌شود و در گروه افرادی قرار می‌گیرند که شایعه را قبول ندارند. در پخش شایعه، تغییر وضعیت افراد هم مشابه حالت پخش بیماری است: افرادی که هنوز در مورد شایعه چیزی نشنیده‌اند و در مورد آن صحبت نمی‌کنند، در صورت شنیدن شایعه، و ارتباط داشتن با افرادی که شایعه را قبول دارند، طبق احتمالی شایعه را باور می‌کنند. از طرفی افرادی که شایعه را باور دارند، با گذر زمان برای آن‌ها مشخص می‌شود که خبر مورد نظر شایعه بوده است. در نتیجه می‌توانیم در مدل پخش شایعه هم سه وضعیت S، I و R را برای افراد در نظر بگیریم که S نشان‌دهنده افرادی است که در مورد شایعه چیزی نمی‌دانند، I نشان‌دهنده‌ی افرادی

[12] Susceptible
[13] Infected
[14] Recovered

است که شایعه را باور کرده‌اند و در مورد آن صحبت می‌کنند و R نشان‌دهنده افرادی است که کذب بودن شایعه برای آن‌ها مسلم شده است.

۲,۲,۳ تحلیل مدل SIR بدون در نظر گرفتن نرخ مرگ و میر

حال به بررسی دقیق‌تر مدل SIR می‌پردازیم. برای سادگی، فرض می‌کنیم که تعداد افراد جمعیت ثابت است، و پدیده‌هایی مثل تولد، مرگ و یا مهاجرت که سبب تغییر تعداد افراد جمعیت می‌شود، را در نظر نمی‌گیریم. با توجه به این که ما بیماری را طوری در نظر می‌گیریم که در جامعه به سرعت شیوع پیدا می‌کند، عملاً پدیده‌هایی مثل تولد و مرگ افراد، تاثیری در مدل ما ندارند.

در این مدل نرخ بهبود افراد با γ نمایش داده می‌شود. γ در واقع معیاری است که بیان می‌کند افراد با چه سرعتی از حالت آلوده به حالت بهبودیافته تغییر وضعیت می‌دهند. علاوه بر این، در این مدل پدیده‌ای وجود دارد که به آن تحمیل آلودگی گفته می‌شود. این عامل را با λ نمایش می‌دهند. λ در واقع نرخ سرانه تبدیل راس‌ها از حالت حساس به حالت آلوده است. در نتیجه سرعت ورود افراد جدید به حالت آلوده برابر است با λX که در آن X تعداد افراد حاضر در وضعیت S است. به طور شهودی، تحمیل آلودگی، متناسب با تعداد افرادی است که در وضعیت آلوده وجود دارند. دو حالت برای انتقال بیماری، با توجه به تعداد افراد کلی موجود در شبکه وجود دارد: حالت اول این که $\lambda = \beta Y/N$ و حالت دوم این که $\lambda = \beta Y$ باشد، که در آن Y برابر تعداد افراد حاضر در وضعیت آلوده، N تعداد کل افراد حاضر در شبکه و β عاملی ترکیبی از سرعت ارتباط افراد با یکدیگر و احتمال انتقال بیماری است. به اولین رابطه، انتقال وابسته به تکثر[15]، و به دومین رابطه انتقال وابسته به چگالی [16] گفته می‌شود. این دو نوع انتقال با توجه به ساختار و نوع شبکه‌ای که با آن سر و کار داریم ممکن است کاربرد داشته باشند. انتقال مستقل از تکثر، در شرایطی اتفاق می‌افتد که تعداد ارتباط‌های میان افراد، مستقل از اندازه جمعیت است. برای مثال، در بیماری‌هایی که مربوط به انسان می‌شود، سرعت انتقال بیماری برای فردی در یک شهر پرجمعیت مثل لندن با هفت میلیون نفر جمعیت یا نیویورک با هشت میلیون نفر جمعیت زندگی می‌کند، با فردی که در شهر کم‌جمعیتی مثل کمبریج با صد هزار نفر جمعیت زندگی می‌کند، یکسان است. بر خلاف این حالت، در حالت انتقال وابسته به چگالی، هر چه تعداد افراد حاضر در شبکه بیشتر شود، یا به بیان دیگر، چگالی افراد بیشتر شود، تعداد ارتباط‌های بین افراد در جمعیت هم بیشتر می‌شود. طبق یک قاعده کلی، انتقال وابسته به تکثر بیشتر مناسب است که برای ویروس‌ها و بیماری‌های انسانی در نظر گرفته شود، در حالی که انتقال‌های وابسته به چگالی، بیشتر مناسب بیماری‌های گیاهی یا حیوانی است.

تفاوت بین دو نوع انتقال زمانی در نظر گرفته می‌شود که اندازه جمعیت تغییر کند، و در حالت‌های دیگر عامل $1/N$ را می‌توانیم برای تغییر پارامتری کردن β در رابطه انتقال وابسته به تکثر، حذف کنیم. از طرفی برای سادگی S را برابر با X/N و I را برابر با Y/N در نظر می‌گیریم که به ترتیب نشان‌دهنده نسبت افراد حساس و

[15] Frequency Dependent Transmission
[16] Density Dependent Transmission

آلوده به کل جمعیت باشند. با اعمال این تغییرات جدید رابطه وابسته به تکثر به صورت βSI تبدیل می‌شود. این رابطه نشان‌دهنده سرعت آلوده شدن افراد است. با این فرض که احتمال‌هایی که در این مدل وجود دارند ثابت هستند، معادلات زیر برای این مدل حاصل می‌شوند:

۱. $\frac{dS}{dt} = -\beta SI$

۲. $\frac{dI}{dt} = \beta SI - \gamma I$

۳. $\frac{dR}{dt} = \gamma R$

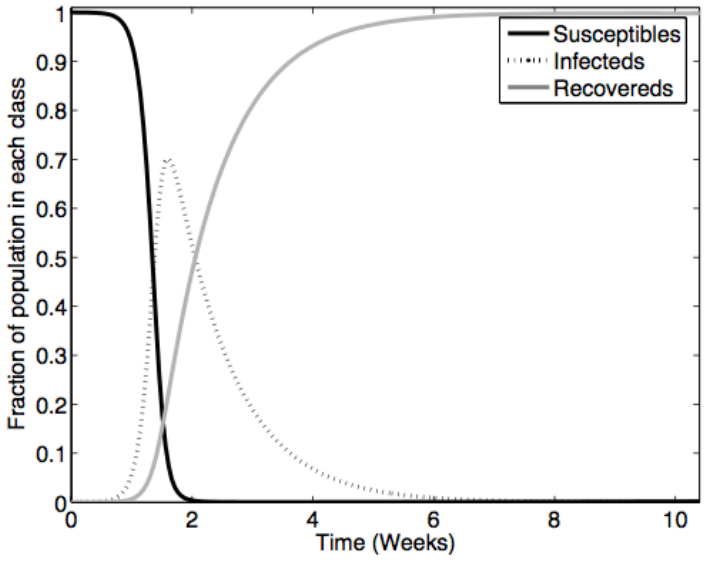
پارامتر γ سرعت بهبود را نشان می‌دهد. البته مقدار $1/\gamma$ برای ما اهمیت بیشتری دارد، چرا که این مقدار دوره بهبود بیماری را نشان می‌دهد. البته در این مدل پخش شایعه لازم نیست که ما معادله‌ای را به طور مستقیم برای گروه R داشته باشیم. زیرا می‌دانیم S+I+R=1. لازم به ذکر است که مقادیر S، I و R نرمال شده هستند و در واقع تعداد افرادی که در این گروه‌ها قرار دارند، بر تعداد کل افراد حاضر در جمعیت تقسیم شده‌اند. در نتیجه با دانستن S و I ما قادر به محاسبه R هستیم. در این معادلات همچنین شرایط اولیه ذیل در نظر گرفته می‌شود:

۴. S(0)>0

۵. I(0)>0

۶. R(0)=0

در شکل زیر، نمونه‌ای از رخ دادن یک شیوع بیماری که از این معادلات پیروی می‌کند، آمده است، این شکل مربوط به مدلی است که تعداد افراد بیمار اولیه یک نفر است و بقیه جمعیت در حالت حساس هستند. مقدار β برابر با ۵۲۰ نفر در سال است، که معادل با ۱,۴۲۸ نفر در روز می‌شود، و مقدار $1/\gamma$ برابر با ۷ روز بوده است.



شکل ۴ . مثالی از پدیده شیوع بیماری

علی‌رغم ساده بودن، این مدل که معادلات ۱، ۲ و ۳ برای آن صادق است، به طور ضمنی قابل حل نیست. یعنی ما نمی‌توانیم تحلیل دقیقی از رفتار S و I در حوزه زمانی داشته باشیم. و این مدل باید با روش عددی حل شود. با این همه، این مدل به دلیل این که دربرگیرنده دو اصل اساسی در شیوع بیماری است، به شدت ارزشمند است.

۲,۲,۴ پدیده آستانه[۱۷]

ابتدا به بررسی مراحل اولیه می‌پردازیم. در شروع، I(0) فرد آلوده و S(0) فرد حساس در سیستم وجود دارند. حال باید ببینیم چه عاملی باعث رخ دادن شیوع بیماری می‌شود. برای بررسی این عامل، معادله ۲را به صورت زیر بازنویسی می‌کنیم:

$$\frac{dI}{dt} = I(\beta S - \gamma) \qquad .۷$$

طبق این معادله تعداد افراد در وضعیت حساس در حالت اولیه (S(0)) کمتر از γ/β باشد، $\frac{dI}{dt} < 0$ و در شیوع بیماری متوقف می‌شود. این نتیجه در تحقیق Kermack و همکارش McKendrick در سال ۱۹۲۷ به دست آمده است که به آن پدیده آستانه گفته می‌شود. این پدیده را طور دیگری نیز می‌توانیم تفسیر کنیم: نسبت γ/β که نرخ بهبود افراد را نشان می‌دهد، باید به اندازه کافی کوچک باشد تا بیماری فرصت شیوع پیدا کند. به معکوس این کسر، نسبت اساسی بازتولیدکننده [۱۸] گفته می‌شود که با R0 آن را نمایش می‌دهند. این نسبت یکی از مهمترین مقادیر در پدیده پخش است که به صورت زیر تعریف می‌شود:

[17] Threshold Phenomenon
[18] Basic reproductive ratio

این نسبت میانگین تعداد افراد ثانویه‌ای است که از میانگین تعداد افراد اولیه حاضر در یک جمعیت حساس به بیماری به وجود می‌آیند.

با استفاده از $R0$ می‌توانیم بیان دیگری برای پدیده آستانه داشته باشیم؛ با فرض این که تمامی افراد جمعیت در حالت اولیه در وضعیت حساس باشند، در صورتی شیوع بیماری اتفاق می‌افتد که $R0 > 1$. چنین چیزی در دنیای واقعی هم برقرار است. به این معنا که یک بیماری اگر نتواند به طور میانگین به بیش از یک نفر منتقل شود، شیوع نمی‌یابد. در جدول ۲,۱ یک چند بیماری نمونه با مقادیر $R0$ تخمین زده شده، آمده است. این مقدار وابسته به بیماری و ساختار جمعیت است. از طریق ریاضی می‌توان مقدار $R0$ را به این صورت محاسبه کرد. ابتدا سرعت وارد شدن افراد به وضعیت آلوده را محاسبه می‌کنیم. سپس این مقدار را در مقدار عددی میانگین دوره بیماری، ضرب می‌کنیم.

مقدار تخمینی $R0$	ناقل	بیماری
۱۰-۱۲	انسان	آبله مرغان
۳-۴	انسان	آنفولانزا
۱۶-۱۸	انسان	سرخک
۱۶-۱۸	انسان	سیاه سرفه
۶-۷	انسان	سرخجه
۲-۶	گاو	سل

شکل ۵. جدول مربوط به اطلاعات بیماری‌های مسری مختلف

۲,۲,۵ پخش فراگیر

بررسی‌هایی که تا به اینجا انجام شده است، مربوط به مراحل اولیه پخش بیماری است. در ادامه ما وضعیت سیستم را در بلندمدت بررسی می‌کنیم. ابتدا معادلات ۱ و ۳ را بر یکدیگر تقسیم می‌کنیم:

$$\frac{dS}{dR} = -\frac{\beta S}{\gamma} = -SR_0 \qquad (۸)$$

با ادغام نسبت به R به رابطه زیر دست می‌یابیم:

$$S(t) = S(0)e^{-R(t)R_0} \quad (۹)$$

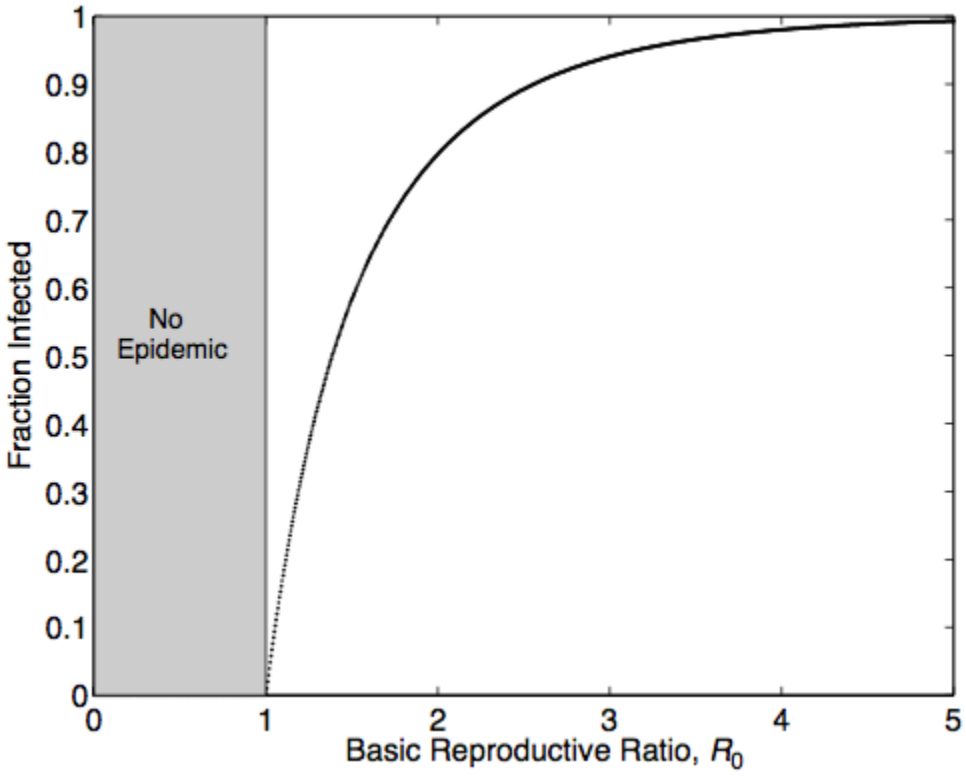
با فرض این که R(0)=0 باشد. در نتیجه با پیشروی شیوع بیماری، تعداد افراد حساس کمتر می‌شود، و تعداد افرای بهبودیافته بر اثر طی شدن دوره بیماری افزایش می‌یابد. با توجه به این که مقدار e^{-R_0} همواره مثبت است، تعداد افراد حساس به بیماری، همیشه بزرگتر از صفر است. در نتیجه، همیشه افرادی در جمعیت حاضر

هستند که دچار بیماری نمی‌شوند. این پدیده، نتیجه مهمی را در بر دارد و آن این که، شیوع بیماری به مرور زمان متوقف می‌شود، و این توقف به دلیل کاهش افراد آلوده است، و دلیل آن وجود نداشتن افراد حساس به بیماری نیست.

همان‌طوری که در مراحل به دست آوردن معادله (۶) نشان داده شد، می‌توان متغیر I را با تقسیم معادله (۳) بر (۱) حذف کرد. که بعد از ادغام، به معادله‌ای از S بر R می‌رسیم. با در نظر گرفتن رابطه S+I+R=1 و همچنین این که شیوع بیماری زمانی متوقف می‌شود که I = 0، معادله (۶) بعد از طی شدن زمان زیاد، به صورت زیر در می‌آید:

$$S(\infty) = 1 - R(\infty) = S(0)e^{-R(\infty)R_0}$$
$$\Rightarrow 1 - R_\infty - S(0)e^{-R(\infty)R_0} = 0 \qquad (۱۰)$$

که در آن R_∞ نسبت نهایی افراد بهبودیافته است، که برابر با جمعیت کلی افراد است که به بیماری آلوده می‌شوند.



شکل ۶. نمودار درصد افراد آلوده بر حسب نسبت R0

معادله (۱۰) غیرجبری[19] است و ریشه حقیقی ندارد. در نتیجه پیدا کردن جواب دقیق برای آن ممکن نیست. ولی از آنجایی که وقتی $R_\infty = 0$، مقدار معادله مثبت است، و در زمانی که $R_\infty = 1$ معادله منفی است، در نتیجه معادله باید ریشه‌ای بین صفر و یک داشته باشد. با استفاده از روش‌های استانداردی مثل روش نیوتون-رافسون، و یا حتی روش آزمون و خطا، می‌توان جواب تقریبی برای این معادله پیدا کرد. نتایج به دست آمده از حل معادله با این روش‌ها، با فرض این که S(0)=1 در نمودار شکل ۲ آمده است. این نمودار نشان می‌دهد، که اگر R0<1 پدیده پخش بیماری اتفاق نمی‌افتد.

همچنین این نمودار مشخص می‌کند که هر موقع یک بیماری دارای نسبت بازتولید اساسیِ به اندازه کافی بزرگ باشد (حدوداً بزرگتر از ۵)، تقریبا همه افراد موجود در جامعه (بیش از ۹۹ درصد) دچار بیماری می‌شوند. باید توجه داشته باشیم که نیتجه‌ای که از معادله (۷) به دست آمده است، به طور خاص وابسته به ساختار مدل SIR نیست. چنین نتیجه‌ای می‌تواند از هر مدل احتمالاتی که در ادامه در مورد آن توضیح می‌دهیم، گرفته شود. در مدلی که تنها یک فرد در وضعیت آلوده قرار داشته باشد، و بقیه افراد حاضر در وضعیت حساس باشند، به طور میانگین R0 نفر دیگر در هر مرحله در وضعیت آلوده قرار می‌گیرند. بنابراین، احتمال این که یک فرد مبتلا به بیماری نشود، برابر با $\exp(-\mathrm{R0/N})$ است. حال اگر Z فرد در وضعیت آلوده باشند، احتمال این که فردی به بیماری مبتلا نشود، در این حالت برابر با $\exp(-\mathrm{ZR0/N})$ است. اگر در انتهای شیوع بیماری، نسبت $R_\infty = Z/N$ نفر آلوده باشند، احتمال این که یک فرد درحالت حساس باقی بماند برابر با $S_\infty = \exp(-R_\infty R_0)$ می‌شود، که باید برابر با $1 - R_\infty$ شود. بنابراین طبق مباحث پیشین، معادله $1 - R_\infty = \exp(-R_\infty R_0)$ برقرار است، که مستقل از ساختار دقیق مدل پخش بیماری یا شایعه است.

همانطوری که گفته شد، پیدا کردن جواب دقیق برای مدل SIR (معادلات ۱ تا ۳) ممکن نیست. و این به دلیل غیرخطی بودن عامل انتقال بیماری، یعنی βSI است. با این حال، می‌توان جواب تقریبی برای نمودار شیوع بیماری پیدا کرد، که این جواب تعداد افراد جدیدی است که در بازه‌های زمانی، وارد وضیعت آلوده می‌شوند. مثالی برای نمودار شیوع بیماری در شکل ۳ آمده است. این نمودار تعداد مرگ و میر را در هفته، بر اثر بیماری طاعون در بمبئی در سال‌های ۱۹۰۵ تا ۱۹۰۶ نشان می‌دهد. فرض می‌شود در زمانی که نشانه‌های آلودگی در یک فرد بروز داده می‌شود، این فرد وارد وضعیت آلوده می‌شود. با این فرض، با استفاده از معادله dR/dt نمودار شیوع بیماری قابل رسم است. با انجام محاسبات به معادله زیر می‌رسیم:

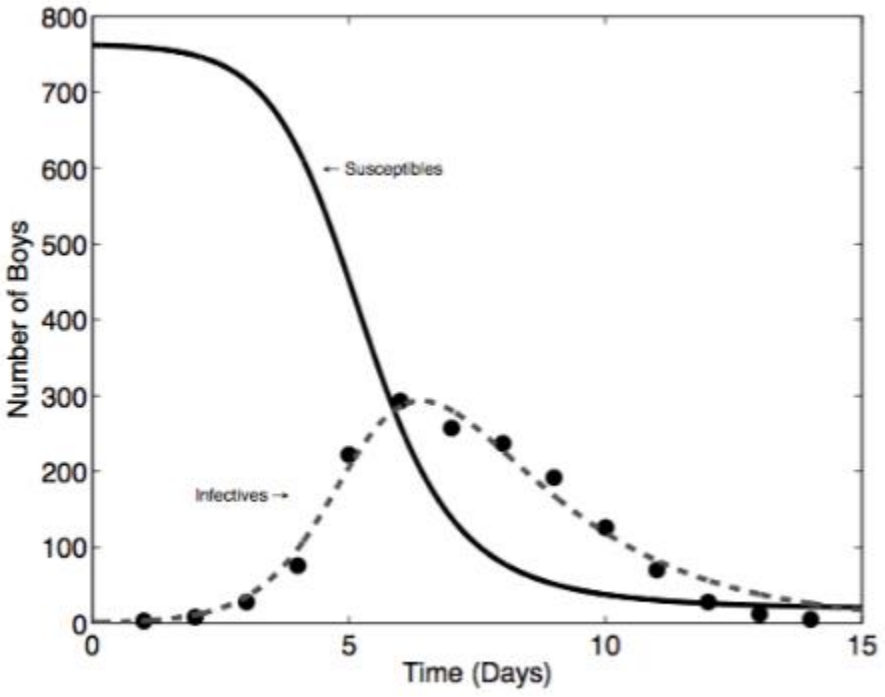
[19] Transcendental

$$\frac{dR}{dt} = \frac{\gamma\alpha^2}{2S(0)R_0^2} sech^2(\frac{1}{2}\alpha\gamma t - \emptyset) \qquad (۸)$$

مقادیر α و $\emptyset$ به شرایط اولیه و بقیه پارامترها وابسته‌اند، و برای هر شیوع خاصی، این مقادیر با توجه به داده‌های واقعی، قابل تخمین زدن هستند. باید توجه داشته داشته باشیم که برای این که این معادله برقرار باشد، مقدار R_0R بسیار کوچک باشد، که چنین چیزی در بسیاری از اپیدمی‌ها اتفاق نمی‌افتد، مخصوصاً در اواخر شیوع بیماری که مقدار R به ۱ نزدیک می‌شود. ولی برای مراحل اولیه پخش بیماری، تا زمانی که شیوع به اوج خود برسد، چنین فرضی معمولاً مناسب است. در هر حال، در بیشتر کاربردهای عملی، باید معادلات SIR را به صورت عددی حل کنیم، تا نحوه تغییر X و Y را در حوزه زمانی محاسبه کنیم. در ادامه به بررسی مثالی در حوزه شیوع بیماری می‌پردازیم.

۲,۲,۶ مثال کاربردی: آنفولانزا در مدرسه‌ای شبانه‌روزی

مثالی مناسب برای شیوع بیماری بدون نرخ مرگ و میر، مثالی است که در مورد پخش آنفولانزا در یک مدرسه شبانه‌روزی در انگلیس است که در سال ۱۹۷۸ اتفاق افتاده است. بعد از شروع ترم عید پاک، سه نفر از دانش‌آموزان پسر به درمانگاه مراجعه کردند و علائم آنفولانزا در آن‌ها دیده شد. بعد از گذشت چند روز، ۷۵۶ دانش‌آموز پسر، در مدرسه دچار بیماری شدند (این دانش‌آموزان با دایره در شکل ۷ نشان داده شده‌اند). بعد از گذشت دو هفته، شیوع بیماری متوقف شد، و این نتیجه با نتایج مدل ساده SIR بدون نرخ مرگ و میر مطابقت دارد.



شکل ۶. نمودار حاصل از شیوع بیماری در مدرسه شبانه‌روزی

با استفاده از تخمین پارامترها در مدل SIR می‌توان در مورد پدیده شیوع این نوع آنفولانزا درک مناسبی پیدا کرد. با استفاده از روش کمینه مربعات (در این روش اختلاف بین موارد پیش‌بینی شده با موارد مشاهده شده، را به کمترین مقدار می‌رسانند)، پارامترهای به دست آمده برای دوره فعال بیماری ($1/\gamma$) را برابر ۲,۲ روز و مقدار میانگین نرخ انتقال (β) ۱,۶۶ نفر در روز است. بنابراین مقدار پیش‌بینی شده برای R0 برای این بیماری از رابطه $\beta/\gamma = 1.66 \times 2.2$ که برابر با ۳,۶۵ می‌شود. همانطوری که در شکل ۷ نشان داده شده است، مدلی با این پارامترها با داده‌های واقعی به دست آمده مطابقت دارد.

۳
فصل سوم
تعریف مسئله

هدف ما در این پروژه این است پدیده پخش شایعه را در شبکه‌ها بررسی کنیم. از آنجایی که بسیاری از شبکه‌هایی که در دنیای واقعی وجود دارند، ساختار و خصوصیاتی مشابه با شبکه‌های پیچیده دارند. به خاطر همین ویژگی‌ها و شباهت‌ها برای شبیه‌سازی و پیاده‌سازی پروتکل‌های مورد نظر پروژه از آن‌ها استفاده می‌کنیم. همچنین به عنوان یک نمونه واقعی، از شبکه نمونه‌برداری شده از توییتر نیز استفاده می‌کنیم.

مدلی که در این پروژه برای پخش شایعه مورد آزمایش قرار گرفته است، مدل SIR[20] است که از جمله مدل‌های مبتنی بر پخش شایعه می‌باشد. برای پیاده‌سازی این مدل هر فرد حاضر در شبکه را با یک راس، و رابطه‌ی آشنایی بین دو فرد را با یال نشان می‌دهیم. هر کدام از افراد موجود، در یکی از سه وضعیت *حساس*[21]، *آلوده*[22]، و یا *بهبودیافته*[23] قرار دارند. در حالت اولیه، اکثر راس‌ها در وضعیت حساس هستند، و تنها درصد کمی از آن‌ها (یک تا ده درصد) در وضعیت آلوده قرار دارند. در هر مرحله، هر یک از راس‌های آلوده یکی از همسایه‌های خود را به طور تصادفی انتخاب می‌کند، و سپس با آن ارتباط برقرار می‌کند. اگر همسایه‌ی انتخاب شده، در وضعیت حساس قرار داشته باشد، با یک احتمال از پیش تعیین‌شده، بعد از برقراری ارتباط، وضعیت راس از حساس به وضعیت آلوده تغییر پیدا می‌کند. به این پخش‌شدن آلودگی در شبکه، در اثر ارتباط راس‌های آلوده با راس‌های حساس،

[20] Susceptible-Infected-Recovered
[21] susceptible
[22] infected
[23] recovered

پخش شایعه گفته می‌شود. نکته مهم این است که راس‌هایی که در وضعیت آلوده قرار دارند، بعد از گذشت دوره زمانی مشخص به وضعیت بهبودیافته تغییر پیدا می‌کنند. لازم به ذکر است که راس‌ها بعد از تغییر وضعیت به حالت بهبودیافته، به وضعیت حساس یا آلوده تغییر وضعیت نخواهند یافت.

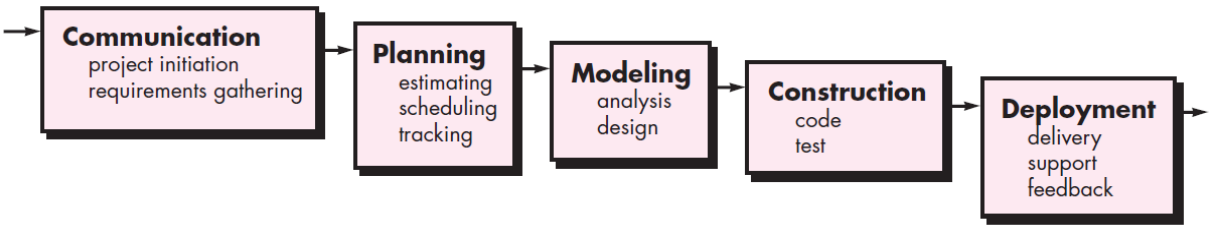
که در حالت اولیه، تعدادی از راس‌ها را در حالت بهبودیافته قرار دهیم، و آن‌ها را در برابر آلوده‌شدن مصون کنیم، و در نهایت تاثیر واکسینه‌کردن را در سرعت پخش شایعه (آلوده‌شدن راس‌ها) بررسی کنیم. لازم به ذکر است که واکسینه‌کردن می‌تواند دو حالت فعال[۲۴] و غیرفعال[۲۵] داشته باشد. در حالت غیرفعال، راس‌هایی که واکسینه شده اند، فقط خودشان تغییر وضعیت می‌دهند، و در تغییر وضعیت راس‌های دیگر تاثیری ندارند. اما، در حالت فعال، راس‌هایی که واکسینه شده‌اند، می‌توانند مانند راس‌های آلوده، در هر مرحله با یکی از همسایه‌های خود تعامل داشته باشند، و با احتمالی از پیش تعیین‌شده، وضعیت همسایه‌های خود را به حالت بهبودیافته تغییر دهند. نکته حائز اهمیت این است که دو روش برای واکسینه‌کردن راس‌ها در حالت اولیه به کار گرفته می‌شود، که در این در روش اول واکسینه کردن راس‌ها به طور تصادفی است، یعنی تعدادی از راس‌ها در حالت اولیه به طور تصادفی انتخاب می‌شوند، و وضعیت آن‌ها در حالت بهبودیافته قرار می‌گیرد. در روش دوم، واکسینه کردن راس‌ها بر اساس درجه آن‌هاست. یعنی در حالت اولیه، راس‌هایی که درجه بیشتری دارند انتخاب می‌شوند و در حالت بهبودیافته قرار می‌گیرند. و ین روش‌ها به روی ساختارهای مختلف شبکه‌ای مانند شبکه‌های مقیاس آزاد و مقیاس آزاد و همچنین شبکه نمونه‌برداری شده از توییتر اعمال می‌شوند.

[24] active
[25] passive

۴
فصل چهارم
پیاده سازی

۴,۱ مهندسی نرم‌افزار

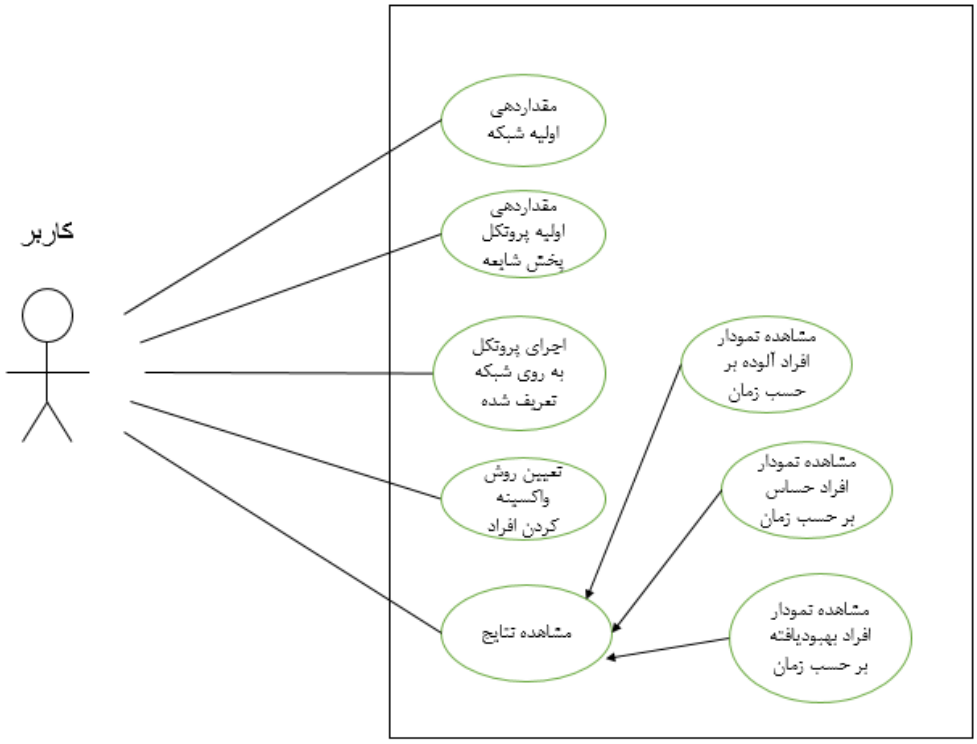
با توجه به این که نیازمندی‌ها در این پروژه، در ابتدای کار مشخص و ثابت بودند،از مدل فرآیندی آبشاری[۲۶] استفاده شده است. مدل فرآیند آبشاری یک مدل فرآیند خطی است و دارای پنج مرحله است که عبارتند از: ارتباط[۲۷]، برنامه‌ریزی[۲۸]، مدل‌سازی[۲۹]، ساخت[۳۰] و استقرار[۳۱]. در شکل ۸ نمودار این مدل آمده است.



شکل ۷. مدل فرآیند آبشاری

در مرحله اول، به تعریف نیازمندی‌های پروژه پرداخته شده است. در مرحله بعدی که برنامه‌ریزی است، برنامه‌ریزی زمانی انجام می‌شود. گام بعدی مربوط به مدل‌سازی نرم‌افزار است، که این تحلیل‌ها با استفاده از نمودار مورد کاربرد[۳۲] انجام می‌شود. این نمودار با توجه به نیازمندی‌های پروژه رسم می‌شود.

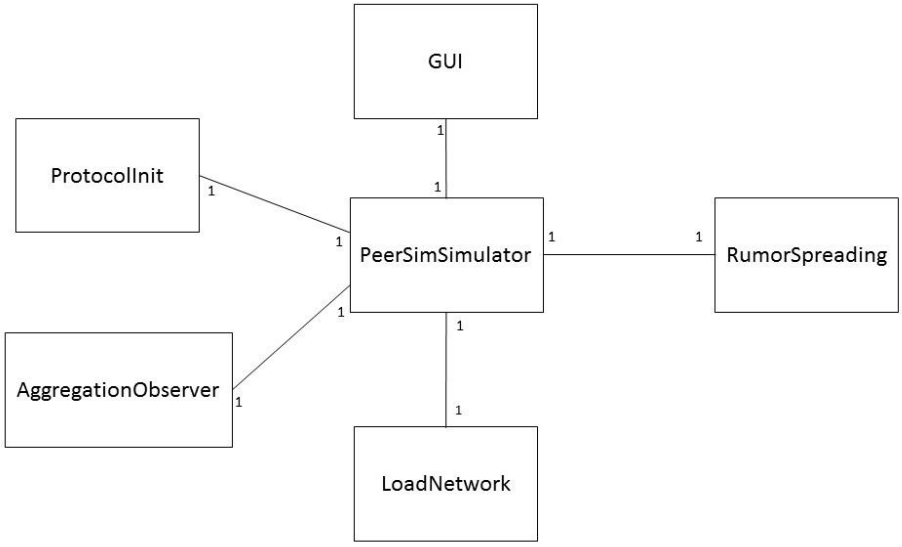
[26] Waterfall Process Model
[27] Communication
[28] Planning
[29] Modeling
[30] Construction
[31] Deployment
[32] ,Use-case diagram



شکل ۸. نمودار مورد کاربرد نرم‌افزار

در مرحله طراحی، مهم‌ترین نمودار که معماری نرم‌افزار را نیز مشخص می‌کند، نمودار کلاس[۳۳] است. این نمودار، کلاس‌های مورد استفاده در نرم‌افزار و نحوه ارتباط بین آن‌ها را مشخص می‌کند. در **Error! Reference source not found.**شکل ۹ نمودار کلاس نرم‌افزار این پروژه آورده شده است. در این نمودار نام کلاس‌ها و رابطه بین آن‌ها مشخص شده است.

[33] Class Diagram



شکل ۹ . نمودار کلاس نرم‌افزار

مرحله بعدی پیاده‌سازی است، که در ادامه به طور دقیق و جزئی به آن پرداخته شده است. نکته حائز اهمیت در مرحله پیاده‌سازی این است که برای راحتی کار، و تقسیم شدن بخش پیاده‌سازی به بخش‌های کوچکتر، این مرحله در دو فاز انجام شده است. در فاز اول به پیاده‌سازی مدل SIR با استفاده از PeerSim پرداخته شده است و در فاز دوم طراحی واسط کاربری انجام شده است.

۴,۲ پیاده‌سازی فاز اول با استفاده از PeerSim

همان‌طوری که گفته شد، در پیاده‌سازی مدل SIR در این پروژه از PeerSim استفاده شده است. در ادامه به معرفی این شبیه‌ساز مبتنی بر زبان جاوا می‌پردازیم.

۴,۲,۱ معرفی Peersim

همان‌طوری که قبلاً گفته شد، در بسیاری از شبکه‌های واقعی، تعداد راس‌های بسیار زیادی وجود دارد. از طرفی راس‌ها مدام در حال وارد شدن به شبکه و خارج شدن از آن هستند. در چنین شبکه‌هایی انجام آزمایش و پیاده‌سازی پروتکل‌های مختلف کار راحتی نیست.

شبیه‌ساز Peersim مبتنی بر زبان جاوا طراحی شده است. طراحی آن به گونه‌ای است که با این ویژگی‌های شبکه‌های پیچیده سازگاری دارد. به طوری در شبکه‌هایی با مقیاس بزرگ و ساختار پویا، عملکرد مناسبی دارد. علاوه بر این، ساختار شبیه‌ساز مبتنی بر مولفه[۳۴] است، و به همین دلیل، پیاده‌سازی یک پروتکل در محیط این شبیه‌ساز، با ترکیب مولفه‌های مختلف که در واقع کلاس‌های جاوا هستند، به راحتی امکان‌پذیر است. با توجه به ویژگی‌های peersim که در ادامه بیشتر به توضیح آن‌ها می‌پردازیم، برای پیاده‌سازی مدل پخش شایعه از آن استفاده شده است. در peersim دو مدل کلی شبیه‌سازی وجود دارد: مدل مبتی بر چرخه[۳۵]، و مدل مبتنی بر رویداد[۳۶].

این مدل ساختار بسیار ساده‌ای دارد، که این ساختار باعث مقیاس‌پذیری و کارایی بالای آن می‌شود. در عوض، نقطه ضعف آن، در این است که کمی از واقعیت فاصله دارد. البته در برخی از پروتکل‌های ساده، این نقطه ضعف، مشکل زیادی ایجاد نمی‌کند. اما به هر حال، باید در هنگام پیاده‌سازی پروتکل‌ها با این مدل، به این نقطه ضعف توجه داشت. برای توضیح بیشتر نحوه کار PeerSim به بررسی آن در یک مدل مبتنی بر چرخه می‌پردازیم.

۴,۲,۲ دوره حیات[۳۷] شبیه‌سازی Peersim

ایده طراحی این شبیه‌ساز، برنامه‌نویسی پیمانه‌ای[۳۸] مبتنی بر اشیا (بلوک‌های سازنده برنامه) بوده است، به طوری که هر بلوک قابل جایگزینی با بلوک دیگری باشد، که رابط[۳۹] مشابهی را پیاده‌سازی کرده است. طرح کلی در شبیه‌سازی مطابق مراحل زیر است:

۱. انتخاب اندازه شبکه (تعداد راس‌ها)
۲. انتخاب یک یا چند پروتکل برای آزمایش و مقداردهی اولیه آن‌ها

[34] Component based
[35] Cycle-based Model
[36] Event-based Model
[37] Life-Cycle
[38] Modular Programming
[39] Interface

۳. انتخاب یک یا چند شی Control برای کنترل ویژگی‌های مورد نظر برنامه‌نویس، و برای تغییر بعضی از پارامترهای شبیه‌سازی (برای مثال اندازه شبکه، وضعیت داخلی پروتکل‌ها)
۴. اجرای شبیه‌سازی با فراخوانی کلاس Simulator با یک فایل تنظیمات[40] که دارای اطلاعات ذکر شده در مراحل قبلی باشد.

همه اشیایی که در زمان شبیه‌سازی ساخته می‌شوند، نمونه‌هایی از کلاس‌هایی هستند که یک یا چند رابط را پیاده‌سازی کرده‌اند. رابط‌های اصلی در شکل ۱۱ آورده شده‌اند.

Node	شبکه مورد آزمایش از تعدادی راس ساخته شده است. هر راس شامل یک یا چند پروتکل می‌شود. Interface برای هر راس، دسترسی به پروتکل‌ها و شناسه آن راس را ممکن می‌سازد.
CDProtocol	این پروتکل برای اجرا در مدل مبتنی بر چرخه طراحی شده است. عملیاتی که در این پروتکل تعریف می‌شوند، در هر چرخه اجرا می‌شوند.
Linkable	یک interface که معمولاً در پروتکل‌ها پیاده‌سازی می‌شود. این interface به پروتکل‌ها این امکان را می‌دهد که به مجموعه‌ای از راس‌ها دسترسی داشته باشند. نمونه‌های پروتکل linkable در راس‌ها، ساختار کلی شبکه را می‌سازد.
Control	کلاس‌هایی که این interface را پیاده‌سازی می‌کنند، می‌توانند طوری تنظیم شوند که مقاطع زمانی خاصی از شبیه‌سازی اجرا شوند. این کلاس‌ها معمولاً برای نظارت و یا تغییر تنظیمات شبیه‌سازی به کار می‌روند.

شکل ۱۰ . توضیح مولفه‌های مختلف موجود در PeerSim

[40] Configuration File

چرخه حیات یک شبیه‌سازی مبتنی بر چرخه به این طریق می‌باشد: ابتدا، فایل تنظیمات خوانده می‌شود، که به عنوان ورودی به کلاس ***Simulator*** داده می‌شود. این فایل شامل تمامی پارامترهای شبیه‌سازی که در آزمایش نقش دارند، می‌شود. بعد از این مرحله، شبیه‌ساز شبکه را می‌سازد و راس‌ها را در آن قرار می‌دهد، و پروتکل‌هایی که هر راس دارد، را مقداردهی اولیه می‌کند. هر راس چند پروتکل دارد. در شبکه آرایه‌ای از نمونه‌های[41] پروتکل‌ها وجود دارد، و هر راس هم یک نمونه از هر کدام از پروتکل‌ها را دارد. نمونه‌های راس‌ها و پروتکل‌ها با clone کردن ساخته می‌شوند. یعنی، تنها یک نمونه از راس‌ها و پروتکل‌ها با استفاده از ***constructor*** شی ساخته می‌شود، که به عنوان نمونه اولیه[42] به حساب می‌آید. و بقیه راس‌ها در شبکه از این نمونه‌اولیه ***clone*** می‌شوند. به همین دلیل، باید به پیاده‌سازی متد clone در پروتکل‌ها توجه ویژه‌ای داشته باشیم.

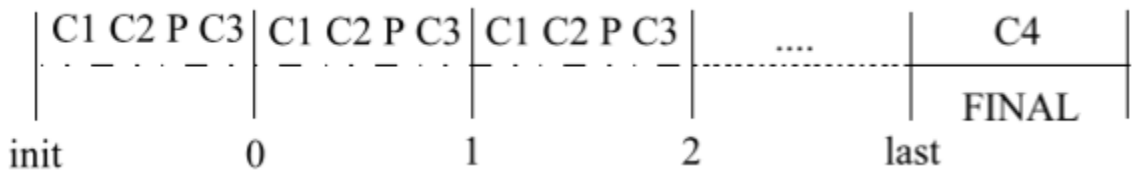
در این مرحله، مقداردهی اولیه انجام می‌شود، و حالت‌های اولیه هر پروتکل تنظیم می‌شود. فاز مقداردهی اولیه، توسط اشیا ***Control*** انجام می‌شود، که این فاز تنها در ابتدای هر آزمایش انجام می‌شود.

در فایل تنظیمات، مولفه‌های مربوط به مقداردهی اولیه با پیشوند init شناخته می‌شوند. نکته حائز اهمیت این است که اشیایی که مربوط به مقداردهی اولیه می‌شوند، هم خود اشیا Control هستند، با این تفاوت که طوری تنظیم شده‌اند که تنها در فاز مقداردهی اولیه اجرا شوند.

بعد از مقداردهی اولیه، موتور مبتنی بر چرخه، همه مولفه‌ها از قبیل پروتکل‌ها و مولفه‌های کنترلی را در هر چرخه، فراخوانی می‌کند. تا زمانی که تعداد چرخه‌ها به حد معینی برسد، یا این که یکی از مولفه‌ها شبیه‌سازی را پایان دهد.

برای هر شی در PeerSim یک شی Scheduler اختصاص داده می‌شود، که زمان اجرای دقیق آن را مشخص می‌کند. به طور پیش‌فرض، هر شی در هر چرخه اجرا می‌شود. اما این امکان هم وجود دارد که یک پروتکل یا شی کنترلی را طوری تنظیم کرد که فقط در چرخه‌های خاصی اجرا شوند، و همچنین می‌توان ترتیب اجرای مولفه‌ها را در هر چرخه کنترل کرد. در شکل ۱۲ نمونه‌ای از اجرای پروتکل‌ها نشان داده شده است.
هنگامی که از یک شی کنترلی برای جمع‌آوری داده‌ها استفاده می‌شود، این داده‌ها قالب‌بندی می‌شوند و به خروجی استاندارد منتقل می‌شوند، و به راحتی می‌توان آن‌ها را در یک فایل ذخیره کرد.

[41] Instance
[42] prototype



شکل ۱۱. مثالی از برنامه‌ریزی مولفه‌های کنترلی و پروتکل‌ها

در شکل ۱۱ برنامه‌ریزی اشیا کنترلی و پروتکل‌ها را می‌بینیم. حروف C نشان‌دهنده مولفه‌های کنترلی هستند. و حرف P یک پروتکل را مشخص می‌کند. عددهایی که در پایین شکل آمده‌اند، شماره چرخه‌ها را نشان می‌دهند. بعد از چرخه پایانی می‌توان یک شی کنترلی نهایی را اجرا کرد، تا اطلاعاتی از وضعیت سیستم در حالت نهایی دریافت شود.

۴,۲,۳ فایل تنظیمات

این فایل، یک فایل اسکی با فرمت **txt**. است، شامل جفت‌های کلید-مقدار[۴۳] می‌شود. خط‌هایی که با کارکتر # شروع می‌شوند، به عنوان کامنت در نظر گرفته می‌شوند. در ادامه، با ذکر مثالی در مورد این فایل بیشتر توضیح می‌دهیم.

در این مثال، شبکه مورد آزمایش، شبکه‌ای تصادفی است که دارای ۵۰۰۰۰۰ راس است. پروتکل مورد آزمایش از پروتکل‌های Aggregation برای میانگین گیری است، به این صورت که ابتدا مجموعه‌ای از اعداد در شبکه پخش می‌شوند. سپس هر راس، به طور مداوم، در هر دوره، یکی از همسایه‌های خود را انتخاب می‌کند، و آن‌ها به طور همزمان، تخمینی از مقدار میانگین می‌زنند، به طوری که این تخمین، بر اساس تخمین قبلی آن‌هاست. در این مثال، با استفاده از یک توزیع خطی، هر یک از راس‌ها مقداری بین ۰ تا ۱۰۰ می‌گیرند، و در نهایت یک شی کنترلی بر مقادیر میانگین پیش‌بینی شده نظارت می‌کند.

[43] Key-value

```
01 # PEERSIM EXAMPLE 1
02
03 random.seed 1234567890
04 simulation.cycles 30
05
06 control.shf Shuffle
07
08 network.size 50000
09
10 protocol.lnk IdleProtocol
11
12 protocol.avg example.aggregation.AverageFunction
13 protocol.avg.linkable lnk
14
15 init.rnd WireKOut
16 init.rnd.protocol lnk
17 init.rnd.k 20
18
19 init.peak example.aggregation.PeakDistributionInitializer
20 init.peak.value 10000
21 init.peak.protocol avg
22
23 init.lin LinearDistribution
24 init.lin.protocol avg
25 init.lin.max 100
26 init.lin.min 1
27
28 # you can change this to select the peak initializer instead
29 include.init rnd lin
30
31 control.avgo example.aggregation.AverageObserver
32 control.avgo.protocol avg
```

شکل ۱۲ . فایل تنظیمات برای مثال ذکر شده در رابطه با aggregation

شکل ۱۲ فایل تنظیمات مربوط به مثال فوق را نشان می‌دهد. اولین نکته حائز اهمیت، اسم کلیدهاست: بعضی از آن‌ها به خاصیت‌های سراسری مربوط می‌شوند، و بعضی دیگر مربوط به مولفه خاصی می‌شوند. برای مثال simulation.cycles سراسری است.

در حالی که protocol.lnk.xxx ویژگی xxx مربوط به پروتکل lnk را تعریف می‌کند. هر یک از مولفه‌های موجود یک اسم دارند، برای مثال lnk اسم یکی از مولفه‌هاست. این اسم اگر برای پروتکل‌ها به کار رود، به یک شاخص عددی نگاشت می‌شود که به آن protocol ID گفته می‌شود. این نگاشت توسط موتور

PeerSim انجام می‌گیرد. این شاخص در فایل تنظیمات نمی‌آید، ولی برای دسترسی به پروتکل‌ها در طول شبیه‌سازی لازم است. در مورد جزئیات بیشتر، بعداً توضیح می‌دهیم.

یک مولفه مانند یک پروتکل و یا کنترل، مطابق با قاعده زیر تعریف می‌شود:

$$< protocol|init|control >.stringID\ [fullPath]classname$$

البته آوردن مسیر کامل کلاس اختیاری است. زیرا PeerSim می‌تواند در مسیر کلاس‌هایش برای پیدا کردن یک کلاس جستجو کند. تنها در صورتی که چند کلاس اسم مشابهی را داشته باشند، آوردن آدرس مسیر کامل کلاس الزامی است. پیشوند init یک شی مقداردهنده اولیه[۴۴] را مشخص می‌کند. این شی باید رابط Control را پیاده‌سازی کند. پارامترهای مربوط به یک مولفه مقداردهنده اولیه مطابق زیر تعریف می‌شوند:

$$< protocol|init|control >.stringID.parameterName$$

برای مثال در خط شماره ۱۰ از شکل ۱,۳ که مربوط به فایل تنظیمات است، اولین پروتکل به وجود می‌آید. قسمت کلید، شامل نوع آن می‌شود که در اینجا protocol است، در قسمت بعدی اسم آن می‌آید که در اینجا lnk است، و در قسمت مقدار، اسم کلاس جاوایی که برای مولفه در نظر گرفته شده است، می‌آید، که در اینجا IdleProtocol است. که این کلاس در خود پکیج PeerSim قرار دارد.

برای هر مولفه پارامترها قابل تعریف هستند. برای مثال، خط ۱۳ شامل اسم و مقدار پارامتر می‌شود. از خط ۳ تا ۸ تعدادی از ویژگی‌های سراسری مربوط به شبیه‌سازی تعریف می‌شوند.

در اینجا تعداد کل چرخه‌های شبیه‌سازی و اندازه شبکه مورد نظر است. مولفه کنترلی Shuffle که در خط ۶ آمده است، ترتیب بررسی راس‌ها را در هر چرخه، تغییر می‌دهد. از خط ۱۰ تا ۱۳، دو پروتکل تعریف می‌شوند. اولی، IdleProtocol است، که کار خاصی به جز نگهداری، لیست مجاورت هر راس انجام نمی‌دهد. پروتکل دوم، کار میانگین‌گیری را انجام می‌دهد. پارامتر linkable که از پارامترهای این پروتکل است، نقش مهمی را ایفا می‌کند. پروتکل aggregation باید با راس‌ها در ارتباط باشد، ولی این پروتکل لیست مجاورت شبکه را به صورت مجزا ندارد. از آنجایی که ساختار PeerSim پیمانه‌ای است، پروتکل مربوط به میانگین‌گیری در هر ساختار شبکه‌ای قابل اجراست. پروتکلی که ساختار شبکه در آن تعریف شده، باید به عنوان پارامتر به پروتکل میانگین‌گیری داده شود. مقدار پارامتر linkable اسم پروتکلی است که رابط Likable را پیاده‌سازی می‌کند که در اینجا IdleProtocol است.

[44] Initializer Object

از خط ۱۵ تا ۲۶، پارامترهایی که تا به حال تعریف شده‌اند، مقداردهی اولیه می‌شوند. در اینجا سه مولفه مقداردهی اولیه تعریف می‌شود. که در واقع فقط از دو تای آن‌ها استفاده می‌شود (با تغییر در خط ۲۹ می‌توانیم پروتکلی‌های مورد نظر را برای مقداردهی اولیه تغییر داد). اولین مقداردهنده اولیه، که peersim.init.WireKOut است، ساختن شبکه را بر عهده دارد، بعد از اجرای این پروتکل، راس‌ها به طور تصادفی به یکدیگر متصل می‌شوند، تا یک گراف با ساختار تصادفی را به طوری که درجه هر راس k باشد به وجود بیاورند.

هر کدام از مولفه‌های دوم و سومی که برای مقداردهی اولیه تعریف شده‌اند، می‌توانند برای مقداردهی اولیه پروتکل aggregation به کار گرفته شوند، و مقادیر اولیه‌ای را که میانگین آن‌ها محاسبه خواهد شد، مشخص می‌کنند. در این مولفه‌ها، مقدارهای اولیه به دو صورت تعیین می‌شوند. یا به روش مقدار حداکثر[۴۵]، یا از روش توزیع خطی. در روش مقدار حداکثر، فقط یک راس مقدار غیرصفر دارد. در روش توزیع خطی، مقادیر راس‌ها به طور خطی افزایشی است. هر دو مولفه مقداردهی اولی یک اسم برای پروتکل نیاز دارند که پروتکلی را که باید مقداردهی اولیه شود، مشخص شود که در اینجا پروتکل lnk است. پارامترهای دیگری هم با توجه به مولفه باید تعیین شوند. این پارامترها شامل پارامترهای دامنه (min و max) برای پروتکل PeakDistributionInitializer و برای پروتکل LinearDistribution مقدار عددی است که باید تعیین شود. انتخاب بین این دو پروتکل برای مقداردهی اولیه با استفاده از خاصیت include.init است که در خط ۲۹ آمده است، که مشخص می‌کند کدامیک از مولفه‌های مقداردهی اولیه اجرا شوند. همچنین این خاصیت، ترتیب اجرای مولفه‌ها را مشخص می‌کند. لازم به ذکر است که ترتیب پیش‌فرض اجرای مولفه‌ها، طبق ترتیب الفبایی نام آن‌هاست. ویژگی include برای پروتکل‌ها و مولفه‌های کنترلی هم قابل استفاده است.

در نهایت در خط ۳۱ و ۳۲، آخرین مولفه تعریف می‌شود: aggregation.AverageObserver. تنها پارامتر این مولفه protocol است، که در اینجا به پروتکل aggregation.AverageFunction نسبت داده می‌شود. پس مقدار آن باید avg باشد.

در شکل زیر خروجی حاصل از اجرای پروتکل مورد نظر را می‌بینیم:

[45] peak

```
control.avgo: 0 1.0 100.0 50000 50.49999999999998 816.7990066335468 1 1
control.avgo: 1 1.2970059401188023 99.38519770395408 50000 50.50000000000005 249.40673287686545 1 1
control.avgo: 2 9.573571471429428 84.38874902498048 50000 50.500000000000085 77.89385877895182 1 1
control.avgo: 3 23.860361582231647 71.93627224106982 50000 50.49999999999967 24.131366707228402 1 1
control.avgo: 4 34.920915967147465 68.92828482118958 50000 50.49999999999994 7.702082905414273 1 1
control.avgo: 5 42.37228198409946 59.94511004870823 50000 50.49999999999987 2.431356211088775 1 1
control.avgo: 6 45.19621912151794 54.855516163070746 50000 50.499999999999844 0.7741451706754877 1 1
control.avgo: 7 47.68716274528092 53.11433934745646 50000 50.49999999999949 0.24515365729069857 1 1
control.avgo: 8 48.97706271318158 52.38916238021276 50000 50.50000000000026 0.07746523384731269 1 1
control.avgo: 9 49.59674440194668 51.46963472637451 50000 50.49999999999937 0.024689348817011823 1 1
control.avgo: 10 49.946490417215266 51.13343750384934 50000 50.50000000000048 0.007807022577928414 2 1
control.avgo: 11 50.18143472395333 50.858337267869565 50000 50.49999999999982 0.002493501256296898 2 1
control.avgo: 12 50.30454978101492 50.67203454827276 50000 50.500000000000206 7.90551008686205E-4 1 1
control.avgo: 13 50.3981394834783 50.60093898689035 50000 50.49999999999967 2.518940347803474E-4 1 1
control.avgo: 14 50.449347314832124 50.54962989951735 50000 50.5000000000003 8.071623184942779E-5 1 1
control.avgo: 15 50.47368195506415 50.52608817343459 50000 50.49999999999999 2.566284350168338E-5 1 1
control.avgo: 16 50.48510475374435 50.518871021756894 50000 50.50000000000012 8.191527862075119E-6 1 1
control.avgo: 17 50.49082426764112 50.51000681641142 50000 50.49999999999945 2.570199757692886E-6 1 1
control.avgo: 18 50.494810505765045 50.50556221303088 50000 50.5000000000003 8.197012224814065E-7 1 1
control.avgo: 19 50.496876367842034 50.50296444951085 50000 50.499999999999524 2.640584231868471E-7 1 1
control.avgo: 20 50.498457906558905 50.50182062146254 50000 50.500000000000334 8.565428611988968E-8 1 1
control.avgo: 21 50.49905541635283 50.50096466374638 50000 50.49999999999974 2.721171621666857E-8 1 1
control.avgo: 22 50.49946061473347 50.500553628252945 50000 50.49999999999975 8.590349265230611E-9 1 1
control.avgo: 23 50.49972602272376 50.500315571370415 50000 50.5000000000004 2.6248542064007986E-9 2 1
control.avgo: 24 50.4998450606816 50.50018053311878 50000 50.50000000000005 8.845012874999227E-10 1 1
control.avgo: 25 50.499894793874255 50.500096923965216 50000 50.50000000000079 1.864501428663076E-10 1 2
control.avgo: 26 50.4999267984512 50.500056126785694 50000 50.5000000000003 8.594896829690765E-11 1 1
control.avgo: 27 50.49996613170552 50.50003198608762 50000 50.50000000000017 1.9554527178661528E-11 1 1
control.avgo: 28 50.49997903068333 50.500019172164286 50000 50.499999999999766 3.274246411310768E-11 1 1
control.avgo: 29 50.49998958653935 50.5000099409645 50000 50.50000000000045 0.0 1 1
```

شکل ۱۳. خروجی استاندارد که نتیجه اجرای پروتکل aggregation است.

در شکل ۱۳ اعداد بسیاری را دیده می‌شود که کلاس ناظر آن‌ها را تولید می‌کند. مقادیر ستون سوم و چهارم داده‌ها در هر خط، مقدار کمینه و بیشنه مقادیر در شبکه را نشان می‌دهد. همانطوری که واضح است، اختلاف بین این دو مقدار در اثر طی شدن مراحل، کمتر می‌شود. در مرحله ۱۲، تقریباً تمامی راس‌ها، تخمین مناسبی از مقدار واقعی میانگین که ۵۰ است، دارند.

۴،۲،۴ مدل مبتنی بر رویداد

در مدل مبتنی بر رویداد، همه چیز مانند مدل مبتنی بر چرخه است، به جز مدیریت زمان، و نحوه فرستاده شدن اشیا کنترلی به پروتکل‌ها. پروتکل‌هایی که بخش اجرایی ندارند، مثل پروتکل‌هایی که برای ذخیره داده‌ها مورد استفاده قرار می‌گیرند، می‌توانند مانند مدل مبتنی بر چرخه مقداردهی اولیه شوند و به کار گرفته شوند. از جمله چنین پروتکل‌هایی می‌توان به پروتکل‌های linkable اشاره کرد که برای ذخیره‌سازی همسایه‌های رئوس در شبکه استفاده می‌شوند، و یا بردارها که برای نگهداری مقادیر عددی هستند. اشیا کنترلی هم در هر پکیجی غیر از peersim.cdsim می‌توانند مانند قبل مورد استفاده قرار بگیرند.

در مدل مبتنی در چرخه، مولفه‌های کنترلی به طور پیش‌فرض در هر چرخه اجرا می‌شوند، ولی در مدل مبتنی بر رویداد، زمان‌بندی آن‌ها باید به طور ضمنی و دقیق مشخص شود، زیرا چرخه‌ای در این مدل وجود ندارد. بنابراین مولفه‌های کنترلی باید به نحوی طراحی شوند که مختص مدل مبتنی بر رویداد باشند، یعنی بتوانند پیام‌هایی به پروتکل‌ها ارسال کنند.

در بسیاری از موارد چنین کاری لازم است، زیرا سیستم به طور جزئی یا کاملاً تحت تاثیر رویدادهای خارجی قرار می‌گیرد، مانند کوئری‌هایی که کاربر وارد می‌کند، و بهترین مدل‌سازی برای این سناریو، زمانی اتفاق می‌افتد که مولفه‌های کنترلی این رویدادها را تولید کنند و به واسطه این رویدادها، اجرای شبیه‌سازی را پیش ببرند. در این مدل مولفه‌هایی قابل استفاده نیستند، که شامل تمام مولفه‌هایی می‌شوند که وابسته به کلاس peersim.cdsim.CDState هستند.

در این کلاس یک interface قرار دارد که برای خواندن ویژگی‌های مختص به چرخه است. البته، بسیاری از مولفه‌هایی را که وابسته به این کلاس هستند، می‌توان به نحوی تغییر داد که این وابستگی از بین برود. از طرفی، نکته جالب توجه این است که تمامی پروتکل‌هایی که رابط مبتنی بر چرخه peersim .cdsim CDProtocol. را پیاده‌سازی می‌کنند، در مدل مبتنی بر رویداد هم قابل استفاده هستند. در ادامه بیشتر در مورد این توضیح می‌دهیم. البته در استفاده از چنین پروتکل‌هایی باید دقت کرد. زیرا در بسیاری از موارد استفاده از آن‌ها در مدل مبتنی بر رویداد، بی‌معنی است. اما به هر حال، این ویژگی کاربردهای مفیدی دارد. در واقع، این امکان را به سیستم می‌دهد که پروتکل‌ها را به طور دوره‌ای فراخوانی کند.

برای توضیح این مدل مانند قبل مثالی را بررسی می‌کنیم. این مثال هم مانند قبل، حساب کردن مقدار میانگین با روش مبتنی بر پخش شایعه است. در اینجا ساز و کار فرستادن پیام توسط راس‌ها به طور دقیق‌تری بررسی می‌شوند.
در این مثال، ابتدا به بررسی کلاس جاوایی که پروتکل میانگین‌گیری را پیاده‌سازی می‌کند، می‌پردازیم.

```
package example.edaggregation;

import peersim.vector.SingleValueHolder;
import peersim.config.*;
import peersim.core.*;
import peersim.transport.Transport;
import peersim.cdsim.CDProtocol;
import peersim.edsim.EDProtocol;

/**
* Event driven version of epidemic averaging.
*/
public class AverageED extends SingleValueHolder
implements CDProtocol, EDProtocol {
```

شکل ۱۴. قطعه کد مربوط به مثال میانگین‌گیری

اولین نکته قابل توجه این است که همان طوری که در شکل ۱۴ می‌بینیم، این کلاس هم رابط EDProtocol و هم CDProtocol را پیاده‌سازی می‌کند. EDProtocol برای پردازش پیام‌هاست. اما CDProtocol در مدل مبتنی بر چرخه، برای پروتکل‌هایی به کار می‌رود که می‌خواهند در برهه‌های زمانی مشخصی به مولفه کنترلی دسترسی داشته باشند. که این کار، با پیاده‌سازی این رابط و تنظیم CDScheduler در فایل تنظیمات امکان‌پذیر است. این کار مزایایی دارد. از جمله آن‌ها، این که کد تمیزتر می‌شود، مولفه اجرای دوره‌ای که جداگانه مدیریت می‌شود می‌تواند به طور جداگانه تنظیم شود، و این که پروتکل‌های مبتنی بر چرخه‌ای را که از قبل طراحی شده‌اند، با این روش راحت‌تر قابل استفاده در مدل مبتنی بر رویداد می‌شوند.

```
/**
 * @param prefix string prefix for config properties
 */
public AverageED(String prefix) { super(prefix); }
```

شکل ۱۵. constructor مربوط به پروتکل میانگین‌گیری

در این مثال این پروتکل پارامتری را از فایل تنظیمات نمی‌خواند. در ادامه پیاده‌سازی رابط مبتی بر چرخه را بررسی می‌کنیم. این متد فعالیتی را مشخص می‌کند که به طور دوره‌ای انجام می‌شود.

```
/**
 * This is the standard method the define periodic activity.
 * The frequency of execution of this method is defined by a
 * {@link peersim.edsim.CDScheduler} component in the configuration.
 */
public void nextCycle( Node node, int pid )
{
        Linkable linkable =
                (Linkable) node.getProtocol( FastConfig.getLinkable(pid) );
        if (linkable.degree() > 0)
        {
                Node peern = linkable.getNeighbor(
                                CommonState.r.nextInt(linkable.degree()));

                // XXX quick and dirty handling of failures
                // (message would be lost anyway, we save time)

                if(!peern.isUp()) return;

                AverageED peer = (AverageED) peern.getProtocol(pid);

                ((Transport)node.getProtocol(FastConfig.getTransport(pid))).
                        send(
                                node,
                                peern,
                                new AverageMessage(value,node),
                                pid);
        }
}
```

شکل ۱۶ . متد مربوط به انجام عملیات دوره‌ای در مدل مبتنی بر چرخه

در اینجا مواردی که خاص مدل مبتنی بر رویداد هستد، وجود دارد، که مربوط به لایه انتقال[۴۶] می‌شوند. کلاس FastConfig به ما امکان دسترسی به لایه انتقالی را که برای این پروتکل تنظیم شده می‌دهد. با استفاده از این لایه انتقال ما می‌توانیم به پروتکل‌های رئوس دیگر پیام بفرستیم. هر پیام می‌تواند یک شی دلخواه باشد. از آنجایی که شبیه‌ساز توزیع‌شده[۴۷] نیست، مشکلاتی از قبیل سریال کردن وجود ندارد و شی با رفرنس ذخیره می‌شود.

پروتکل مقصد برای فرستادن پیام، با راس مقصد peern تعریف می‌شود، و شاخصی که برای پروتکل مقصد به کار می‌رود، pid است. در این مثال، ما پیامی به پروتکل مشابه یک راس دیگر می‌فرستیم. مشخص است که پروتکل مقصد هم باید رابط EDProtocol را پیاده‌سازی کند.

[46] Transport layer
[47] Distributed

```
/**
* This is the standard method to define to process incoming messages.
*/
public void processEvent( Node node, int pid, Object event ) {

        AverageMessage aem = (AverageMessage)event;

        if( aem.sender!=null )
                ((Transport)node.getProtocol(FastConfig.getTransport(pid))).
                        send(
                                node,
                                aem.sender,
                                new AverageMessage(value,null),
                                pid);

        value = (value + aem.value) / 2;
}

}
```

شکل ۱۷ . متد مربوط به پردازش پیام‌ها

متد شکل ۱۲، در EDSimulator مشخص شده و وظیفه آن بررسی پیام‌هاست. در این مثال، تنها یک نوع پیام وجود دارد. تنها لازم است که بررسی کنیم که فرستنده null هست یا خیر. زیرا در این صورت، نیاز به پاسخ دادن به آن پیام نیست، چرا که خود پاسخی است که توسط پروتکل راس دیگری فرستاده شده است. اگر نیاز به جواب دادن به پیام باشد، این کار با استفاده از لایه انتقال انجام می‌شود.

```
/**
* The type of a message. It contains a value of type double and the
* sender node of type {@link peersim.core.Node}.
*/
class AverageMessage {

        final double value;
        /** If not null,
        this has to be answered, otherwise this is the answer. */
        final Node sender;
        public AverageMessage( double value, Node sender )
        {
                this.value = value;
                this.sender = sender;
        }
}
```

شکل ۱۸. کلاس مربوط به پیام

کلاس private که در شکل ۱۸ می‌بینید، برای تعریف نوع پیامی که پروتکل استفاده می‌کند، به کار گرفته شده است.

۴,۲,۴,۱ فایل تنظیمات

در اینجا به بررسی یک فایل تنظیمات برای اجرای شبیه‌سازی مبتنی بر رویداد می‌پردازیم. این فایل تفاوت کمی با حالت مبتنی بر چرخه دارد، که البته این تفاوت‌ها مهم هستند.

```
# network size
SIZE 1000

# parameters of periodic execution
CYCLES 100
CYCLE SIZE*10000

# parameters of message transfer
# delay values here are relative to cycle length, in percentage,
# eg 50 means half the cycle length, 200 twice the cycle length, etc.
MINDELAY 0
MAXDELAY 0
# drop is a probability, 0<=DROP<=1
DROP 0
```

شکل ۱۹. تعریف ثابت‌ها در فایل تنظیمات

مطابق با شکل ۱۹، تعدادی ثابت برای بیشتر شدن خوانایی فایل تعریف شده‌اند. برای مثال CYCLE طول یک چرخه را مشخص می‌کند.

```
random.seed 1234567890
network.size SIZE
simulation.endtime CYCLE*CYCLES
simulation.logtime CYCLE
```

شکل ۲۰. تنظیمات ویژگی‌های سراسری شبیه‌سازی

پارامتر simulation.endtime در شکل ۲۰ زمان پایان شبیه‌سازی را مشخص می‌کند. زمان در داخل برنامه به صورت یک عدد صحیح ۶۴ بیتی در نظر گرفته می‌شود. در شروع، مقدار آن صفر است، و طی رد و بدل شدن پیام‌ها مقدار آن افزایش می‌یابد. شبیه‌سازی زمانی پایان می‌یابد که صف رویدادها خالی شده باشد، و رویداد دیگری برای پردازش باقی نمانده باشد، و یا پایان شبیه‌سازی در صورت رسیدن زمان مقرر تعیین شده برای پایان برنامه انجام می‌گیرد.

شبیه‌ساز به واسطه خطای استاندارد[۴۸] پیام‌هایی را در مورد زمان پیشرفت شبیه‌سازی در اختیار ما قرار می‌دهد. پارامتر simulation.logtime تعداد این پیام‌ها را مشخص می‌کند.

```
################### protocols ===========================

protocol.link peersim.core.IdleProtocol

protocol.avg example.edaggregation.AverageED
protocol.avg.linkable link
protocol.avg.step CYCLE
protocol.avg.transport tr

protocol.urt UniformRandomTransport
protocol.urt.mindelay (CYCLE*MINDELAY)/100
protocol.urt.maxdelay (CYCLE*MAXDELAY)/100

protocol.tr UnreliableTransport
protocol.tr.transport urt
protocol.tr.drop DROP
```

شکل ۲۱. قطعه کد مربوط به تنظیم و تعریف پروتکل‌های موجود در شبیه‌سازی

مطابق با کد شکل ۲۱ ما تنظیمات پروتکل avg را انجام دهیم، و شبکه مورد نظر را که در اینجا link است، و همچنین لایه انتقال را که با tr نشان داده شده است، مشخص می‌کنیم.

از طرفی چون در این پروتکل رابط مبتنی بر چرخه را پیاده‌سازی کردیم، باید پارامتر step را که مربوط به زمان‌بندی است، تعیین کنیم، که این پارامتر مدت زمان هر چرخه را مشخص می‌کند.

[48] Standard error

ساختاری که برای شبکه تعیین می‌کنیم، پروتکلی است که لیست مجاورت راس‌ها را ذخیره می‌کند و در طول شبیه‌سازی تغییر نمی‌کند. نحوه مقداردهی اولیه این پرتوکل در شکل ۲۲ آمده است.

لایه انتقال هم به عنوان یک پروتکل در نظر گرفته می‌شود. این پروتکل قابلیت مدل‌سازی تاخیرهای تصادفی و از دست رفتن پیام‌ها را دارد. در اینجا ابتدا یک لایه انتقال را که تاخیر تصادفی دارد، تعریف می‌کنیم (urt)، و در نهایت آن را در یک غالب بسته‌بندی[۴۹] عمومی که هر نوع لایه انتقالی را دریافت می‌کند، قرار می‌دهیم، که قابلیت شبیه‌سازی گم شدن پیام‌ها را با یک احتمال از پیش تعریف شده، دارد، و نام آن در اینجا tr است. لایه‌های انتقال در بسته peersim.transport تعریف شده‌اند. این مولفه هم مانند بقیه مولفه‌ها ماژولار است، و برنامه‌نویس می‌تواند مطابق با نیاز خود، در آن‌ها تغییر دهد.

```
################### initialization ======================

init.rndlink WireKOut
init.rndlink.k 20
init.rndlink.protocol link

init.vals LinearDistribution
init.vals.protocol avg
init.vals.max SIZE
init.vals.min 1

init.sch CDScheduler
init.sch.protocol avg
init.sch.randstart
```

شکل ۲۲. بخش مربوط به مشخص کردن مولفه‌های مقداردهی اولیه

تنها مولفه‌ای که در این مثال مخصوص مدل مبتنی بر رویداد است، و آن را در قطعه کد شکل ۲۲ مشاهده می‌کنیم، sch است. این مولفه برای فراخوانی دوره‌ای متد nextCycle از رابط مبتنی بر چرخه، به کار می‌رود. در این تنظیمات، این مولفه ابتدا عددی تصادفی بین ۰ و Cycle به تمام راس‌ها نسبت می‌دهد. این مقدار در واقع نشان‌دهنده زمانی است که اولین بار متد nextCycle در پرتوکل avg فراخوانی می‌شود. فراخوانی‌های بعدی برای این متد، در بازه‌های زمانی به طول CYCLE اتفاق می‌افتند.

[49] Wrapper

```
################ control ==============================

control.0 SingleValueObserver
control.0.protocol avg
control.0.step CYCLE
```

شکل ۲۳. تنظیمات مربوط به تعریف مولفه کنترل

همان‌طوری که در قطعه کد شکل ۲۳ می‌بینیم، در اینجا باید پارامتر step را برای مولفه کنترلی هم تعریف کنیم. این پارامتر مشخص می‌کند که چند بار این مولفه فراخوانی شود. در ادامه نتایج حاصل از اجرای پروتکل را بررسی می‌کنیم. اگر ما با این تنظیمات، برنامه را اجرا کنیم، مطابق با شکل ۲۴ خروجی خطای استاندارد[۵۰] را می‌بینیم:

```
Simulator: loading configuration
ConfigProperties: File config-edexample.txt loaded.
Simulator: starting experiment 0 invoking peersim.edsim.EDSimulator
Random seed: 1234567890
EDSimulator: resetting
Network: no node defined, using GeneralNode
EDSimulator: running initializers
- Running initializer init.rndlink: class peersim.dynamics.WireKOut
- Running initializer init.sch: class peersim.edsim.CDScheduler
- Running initializer init.vals: class peersim.vector.LinearDistribution
EDSimulator: loaded controls [control.0]
Current time: 0
Current time: 10000000
Current time: 20000000
Current time: 30000000
Current time: 40000000
Current time: 50000000
.
.
.
Current time: 980000000
Current time: 990000000
EDSimulator: queue is empty, quitting at time 999980413
```

شکل ۲۴. خروجی خطای استاندارد

و شکل ۲۵ مربوط به خروجی استاندارد می‌شود:

[50] Standard Error

```
control.0: 1.0 1000.0 1000 500.5 83416.66666666667 1 1
control.0: 37.5 919.0 1000 500.5 25724.159091250687 1 1
control.0: 206.7109375 767.890625 1000 500.5 8096.807036889389 1 1
control.0: 352.373046875 695.453125 1000 500.5 2578.022573176135 1 1
control.0: 412.430419921875 625.474609375 1000 500.5 801.1082179446831 1 1
control.0: 436.43787479400635 570.459858417511 1000 500.5 243.53994072762902 1 1
control.0: 470.7608990445733 527.0359845032217 1000 500.49999999999994 74.13788674564383 1 2
control.0: 483.6040476858616 518.0301055684686 1000 500.49999999999903 23.428974301677556 1 1
control.0: 490.5196089811798 512.0301471857779 1000 500.4999999999993 7.285566419597019 1 1
control.0: 494.97216907397836 506.0375954180854 1000 500.4999999999999 2.1798299307442246 1 1
control.0: 497.18190345272336 503.5837144460532 1000 500.5000000000001 0.6073148838336206 1 1
control.0: 498.54320551492475 502.3533156558903 1000 500.5 0.1786794435445898 1 2
control.0: 499.4023441821402 501.4962048486104 1000 500.49999999999966 0.055257607540637785 1 1
control.0: 500.0032071191514 501.09832936709677 1000 500.4999999999995 0.017914865984002482 1 1
.
.
.
control.0: 500.5 500.5 1000 500.5 0.0 1000 1000
```

شکل ۲۵. اطلاعات خروجی استاندارد

این مقادیر در هر سطر به ترتیب مینیمم، ماکسیمم، تعداد نمونه‌ها، میانگین، واریانس، تعداد نمونه‌های با مقدار مینیمم، و در نهایت تعداد نمونه‌های با مقدار ماکسیمم هستند. این خروجی نشان می‌دهد که مقدار میانگین درست، یعنی ۵۰۰,۵ به دست آمده است، در حالی که مقدار واریانس صفر است، یعنی تمامی راس‌ها مقدار درست میانگین را در بر دارند.

۴,۳ نحوه استفاده از PeerSim در پیاده‌سازی پروژه

حال به توضیح پیاده‌سازی پروژه شبیه‌سازی پخش شایعه می‌پردازیم. ما برای پیاده‌سازی پروژه از مدل مبتنی بر رویداد استفاده کردیم. به این صورت که راس‌ها در شبکه، برای ارتباط با یکدیگر از ساز و کار فرستادن پیام استفاده می‌کنند. جزئیات پیاده‌سازی پروتکل‌ها در بخش‌های بعد آمده است. در ابتدا به توضیح فایل تنظیمات مربوط به پروژه می‌پردازیم.

۴,۳,۱ فایل تنظیمات پروژه

همان‌طوری که گفته شد، کلاسی که در PeerSim متد main را دارد، کلاس Simulator هست. این کلاس به عنوان ورودی فایل تنظیمات را می‌گیرد. ابتدا به توضیح فایل تنظیمات پروژه می‌پردازیم، بعد از آن به طور جزئی‌تر در مورد کلاس‌ها توضیح می‌دهیم.

در شکل ۲۶ بخشی که در رابطه با تعیین ثابت‌ها و همچنین تنظیمات مربوط به شبیه‌سازی است آمده است. در خط ۱ تا ۱۴ ثابت‌ها تعریف شده‌اند: SIZE تعداد راس‌های شبکه است، CYCLES تعداد چرخه‌ها را نشان می‌دهد، و CYCLE طول هر چرخه را مشخص می‌کند. خط ۱۱ تا ۱۴، ثابت‌های مربوط به تاخیر در ارسال پیام، و احتمال

گم شدن آن را نشان می‌دهند، که ما در اینجا آن‌ها را صفر در نظر می‌گیریم. در خط ۱۶ seed مربوط به تابع random تعریف شده است. خط‌های بعدی پارامترهای مربوط به شبیه‌سازی را نشان می‌دهند، endtime زمان پایان شبیه‌سازی، logtime زمان گزارش‌گیری از وضعیت سیستم، و experiments تعداد کل آزمایش‌ها را نشان می‌دهد.

```
1  # network size
2  SIZE 2731
3
4  # parameters of periodic execution
5  CYCLES 100
6  CYCLE SIZE
7
8  # parameters of message transfer
9  # delay values here are relative to cycle length, in percentage,
10 # eg 50 means half the cycle length, 200 twice the cycle length, etc.
11 MINDELAY 0
12 MAXDELAY 0
13 # drop is a probability, 0<=DROP<=1
14 DROP 0
15
16 random.seed 1234567890
17 network.size SIZE
18 simulation.endtime 80000
19 simulation.logtime  5000
20 simulation.experiments 10
```

شکل ۲۶. بخش اول فایل تنظیمات PeerSim در پروژه

مطابق با شکل ۲۷، خط ۲۴ تا ۴۱ مربوط به تعریف پروتکل‌ها می‌شود. در خط ۲۴ و ۲۵ ابتدا پروتکل مربوط به شبکه پوششی در سیستم مشخص می‌شود، بعد طول هر چرخه در آن تعیین می‌شود. در خط ۲۷ تا ۳۲ پروتکل مربوط به پخش شایعه را تعریف می‌کنیم. پارامترهای این پروتکل عبارتند از: احتمال انتقال شایعه (rumor_prob)، شبکه پوششی (linkable) که پروتکل در آن اجرا می‌شود، طول هر چرخه (step)، لایه انتقال (transport) که برای فرستادن پیام‌ها استفاده می‌شود. و در نهایت، در خط ۳۴ تا ۴۱ پروتکل مربوط به لایه انتقال تعریف می‌شوند، در مورد این پروتکل‌ها در بخش قبل، در مثال مربوط به میانگین‌گیری به تفصیل توضیح داده شده است.

```
21  ################### protocols =============
22
23
24  protocol.link NetworkOverlay
25  protocol.link.cycle 1000
26
27  protocol.avg NodeStateHolder
28  protocol.ave.rumor_prob 80
29  protocol.avg.linkable link
30  protocol.avg.step CYCLE
31  protocol.avg.transport tr
32  protocol.avg.overlay link
33
34  protocol.urt UniformRandomTransport
35  #protocol.urt.maxdelay (CYCLE*MAXDELAY)/100
36  protocol.urt.mindelay 0
37  protocol.urt.maxdelay 0
```

شکل ۲۷ . بخش دوم فایل تنظیمات PeerSim در پروژه

در شکل ۲۸ تنظیمات مربوط به مقداردهی اولیه پروتکل‌ها آمده است. خط‌ها ۴۵ تا ۵۶ مربوط به ساختن شبکه‌ها می‌شود. این شبکه‌ها، می‌توانند تصادفی، خوشه‌بندی شده، و یا شبکه‌ای از نمونه‌های واقعی باشند. در بخش‌های بعد در مورد الگوریتم‌های ساخت این شبکه‌ها توضیح می‌دهیم. خط‌های ۵۹ تا ۶۴ پروتکلی تعریف می‌شود که مقداردهی راس‌ها در حالت اولیه را بر عهده دارد. پارامترهای آن عبارتند از protocol که در اینجا avg است که مقداردهی اولیه avg را بر عهده دارد. از پارامترهای دیگر درصد اولیه افراد آلوده به بیماری و همچنین درصد افراد واکسینه شده هستند. پروتکل آخر مربوط وظیفه زمان‌بندی پروتکل avg را بر عهده دارد.

```
43  ################### initialization ======================
44
45  #init.sw WireScaleFreeDM
46  #init.sw.k 2
47  #init.sw.protocol link
48  ##init.rndlink WireKOut
49  ##init.rndlink.k 2
50  ##init.rndlink.protocol link
51  ###init.net GenerateModularNetwork
52  ###init.net.beta 0.2
53  ###init.net.cluster_size 100
54  ###init.net.k 4
55  init.net LoadNetwork
56  init.net.address /Users/amazloomian/Downloads/twitter.txt
57  init.net.overlay_protocol link
58
59  init.xnet InfectionInit
60  init.xnet.protocol avg
61  init.xnet.initial_infected_percentage 1
62  init.xnet.vaccinated_percentage 20
63
64  init.sch CDScheduler
65  init.sch.protocol avg
66  init.sch.randstart
```

شکل ۲۸. بخش سوم فایل تنظیمات PeerSim در پروژه

در آخر هم همان طور که در شکل ۲۹ آمده است، مولفه کنترلی این برنامه که وظیفه گزارش‌گیری از وضعیت سیستم را بر عهده دارد تعریف می‌شود. پارامتر پروتکل مشخص می‌کند که این مولفه کنترلی برای چه پروتکلی به کار رود و پارامتر step طول هر چرخه برای این مولفه مشخص می‌کند.

```
67  ################ control ==========
68
69  control.1 AggregationObserver
70  control.1.protocol avg
71  control.1.step CYCLE
```

شکل ۲۹. بخش مربوط به تعریف مولفه‌های کنترلی در فایل تنظیمات PeerSim

در ادامه در موردی چندی از کلاس‌های پیاده‌سازی شده که نقش مهمی در شبیه‌سازی دارند، می‌پردازیم.کلاس‌های اصلی RumorSpreading، AggreagtionObserver و InfectionInit هستند.

۴,۳,۲ کلاس RumorSpreading

این کلاس پیاده‌سازی مدل SIR و پخش شایعه در شبکه پیاده‌سازی شده است. در پیاده‌سازی، به هر یک از راس‌ها مقدار عددی صفر، یک یا دو نسبت داده می‌شود. مقدار صفر نشان‌دهنده راس‌هایی است که در حالت Susceptible یا حساس قرار دارند. مقدار یک، مربوط به راس‌هایی می‌شود که در حالت Infected یا آلوده قرار دارند، و مقدار دو، برای راس‌های استفاده می‌شود که در حالت Recovered یا بهبودیافته قرار دارند.

در این کلاس رابط‌های EDProtocol و CDProtocol پیاده‌سازی شده است. برای همین متدهای nextCycle و proccessEvent باید در این کلاس پیاده‌سازی شوند. همان‌طور که قبلاً هم اشاره کردیم، متد nextCycle برای انجام یک کار مشخص در بازه‌های زمانی از پیش تعیین شده است. در این کلاس، در هر چرخه، هر راس یکی از همسایه‌های خود را به طور تصادفی انتخاب می‌کند. و به آن پیامی را که دربردارنده وضعیت خود راس هست، می‌فرستد. برای این کار ابتدا به واسطه پروتکل linkable به لیست مجاورت راس‌ها دسترسی پیدا می‌کنیم. سپس یکی از آن‌ها را با استفاده از متد getNeighbor و به طور تصادفی انتخاب می‌کنیم. بعد با استفاده از لایه انتقال راس فعلی، پیام را به راس مقصد می‌فرستیم. بخش دیگری که در این متد وجود دارد، مربوط به بررسی وضعیت راس فعلی می‌شود. اگر این راس در وضعیت آلوده قرار داشته باشد، یک شمارنده را برای آن افزایش می‌دهد. اگر این شمارنده به مقدار مشخصی (MAX_DAYS) برسد، وضعیت راس از حالت آلوده به حالت بهبودیافته تغییر می‌کند، یعنی مقدار عددی آن از عدد یک به عدد دو تغییر پیدا می‌کند. در شکل ۳۰ کد مربوط به متد nextCycle آمده است.

```
public void nextCycle(Node node, int protocolID) {

    if(this.value==1)
        infectedDur++;
    if(infectedDur>= MAX_DAYS)
        value=2;

    NetworkOverlay linkable = (NetworkOverlay) node.getProtocol(FastConfig.getLinkable(protocolID));
    //  NetworkOverlay linkable = (NetworkOverlay) node.getProtocol(overlayPID);

    if(linkable.degree() > 0){

        Node peern = linkable.getNeighbor(CommonState.r.nextInt(linkable.degree()));
        if(!peern.isUp())
            return;
        ((Transport) node.getProtocol(FastConfig.getTransport(protocolID)))
                .send(node, peern, new AverageMessege(value, node), protocolID);;

    }

}
```

شکل ۳۰. متد مربوط به عملیات دوره‌ای در شبیه‌سازی مدل SIR

پردازش پیام‌ها با استفاده از متد processEvent انجام می‌شود. در این متد بررسی می‌شود که پیام ارسالی حاوی چه مقداری است. اگر مقدار پیام، عدد یک باشد، به این معناست که راس فرستنده پیام در وضعیت آلوده قرار داشته است. حال اگر وضعیت راس فعلی در حالت حساس باشد، یعنی مقدار عددی آن صفر باشد، با احتمالی که به عنوان پارامتر ورودی به برنامه داده شده است، راس فعلی به وضعیت آلوده تغییر وضعیت پیدا می‌کند. در حالت واکسیناسیون فعال، اگر راس فعلی در وضعیت حساس باشد، با احتمال از پیش تعیین شده، راس فعلی به وضعیت بهبودیافته تغییر وضعیت می‌دهد. در شکل ۳۱ پیاده‌سازی مربوط به processEvent آمده است.

```
public void processEvent(Node node, int pid, Object event) {
    Random infProb= new Random();

    AverageMessege messege= (AverageMessege) event;

    if(messege.val==1 && this.value!=2 && infProb.nextInt(100)<=INFECTION_PROB  )
        value=1;
    else if(messege.val==2 && value!=1 && infProb.nextInt(100)<=70)
        value=2;
    if(messege.sender!=null)
        ((Transport)node.getProtocol(FastConfig.getTransport(pid)))
        .send(node, messege.sender,new AverageMessege(value, null) , pid);;

}
```

شکل ۳۱. متد مربوط به پردازش پیام‌ها در شبیه‌سازی مدل SIR

۴,۳,۳ کلاس InfectionInit

این کلاس برای مقداردهی اولیه به راس‌ها در شبکه به کار می‌رود. از جمله متدهای این کلاس randomInfection، hubVaccination و randomVaccination هستند. در متد randomInfection درصدی از راس‌ها که توسط برنامه‌نویس تعیین شده است ()، انتخاب می‌شوند و در حالت آلوده قرار می‌گیرند، یعنی مقدار عددی آن‌ها برابر با یک می‌شود. دو متد دیگر برای واکسینه کردن راس‌ها به کار می‌روند. متد randomVaccination درصدی از راس‌ها را که کاربر تعیین کرده است در حالت بهبود یافته قرار می‌دهد، و مقدار عددی آن‌ها را یک می‌کند. در متد hubVaccination راس‌ها را بر اساس درجه مرتب‌سازی می‌کنیم. آن‌هایی که درجه بیشتری دارند در اولویت برای واکسینه کردن قرار می‌گیرند. در شکل‌های ۳۲ تا ۳۴ قطعه کدهای مربوط به این متدها آمده‌اند.

```
private void randomInfection() {

      for(int i=0;i<Network.size();i++){
           RumorSpreading ae= (RumorSpreading) Network.get(i).getProtocol(pid);;
           ae.setValue(0);
       }

       int numCurrentInfected = 0;
      int numInitalInfected = Network.size() * initialInfectedPercentage / 100;
      // int numInitalInfected =540;
       while (numCurrentInfected <= numInitalInfected) {

           int index = rnd.nextInt(Network.size());
           RumorSpreading ae = (RumorSpreading) Network.get(index).getProtocol(pid);

           if (ae.getValue() == 0) {
               ae.setValue(1);
               numCurrentInfected++;
           }

       }
   }
```

شکل ۳۲. متد مربوط به آلوده کردن تصادفی راس‌ها

```
private void hubVaccination(){

      for(int i=0;i<Network.size();i++){
          Linkable link=(Linkable)Network.get(i)
              .getProtocol(FastConfig.getLinkable(pid));
          nodes.add(new MyNode(i, link.degree()));
      }
      Collections.sort(nodes,Collections.reverseOrder());
      for(int i=0;i<540;i++){
          RumorSpreading ae=(RumorSpreading) Network.get
              (nodes.get(i).index).getProtocol(pid);
          ae.setValue(2);
      }
```

شکل ۳۳. متد مربوط با واکسینه کردن راس‌ها مبتنی بر درجه

```
private void randomVaccination(){

    //  int numToVaccinate=
          vaccinatedPercentage*Network.size()/100;
      int numToVaccinate=540;
      Random r= new Random();
      int numVaccinated=0;
      int numNodes=Network.size();
      while(numVaccinated<=numToVaccinate){
          int index=r.nextInt(numNodes);
          RumorSpreading nsh=(RumorSpreading)
              Network.get(index).getProtocol(pid);
          if(nsh.getValue()==0){
              nsh.setValue(2);
              numVaccinated++;
          }

      }
  }
```

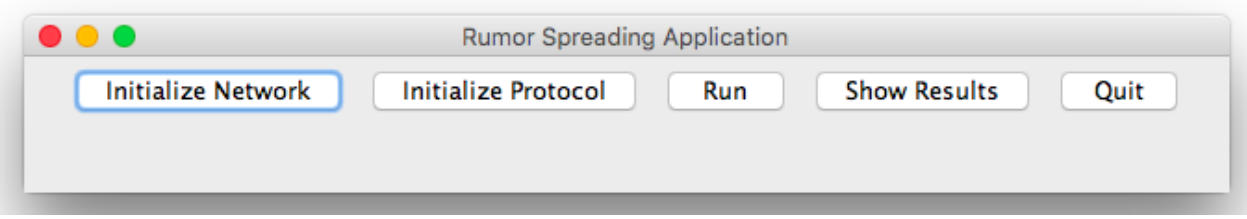
شکل ۳۴ . متد مربوط به واکسینه کردن تصادفی راس‌ها

۴,۳,۴ کلاس AggregationObserver

این کلاس مولفه کنترلی برای شبیه‌سازی محسوب می‌شود. از این کلاس برای گرفتن گزارش از سیستم در زمان‌های مشخص شده توسط برنامه‌نویس استفاده می‌شود. زمان‌های گرفتن گزارش از سیستم در فایل تنظیمات با استفاده پارامتر simulation.logtime آمده است. در گزارش‌گیری، در سه فایل متفاوت، نسبت راس‌هایی که در حالت حساس، آلوده و بهبودیافته قرار دارند، نوشته می‌شود.

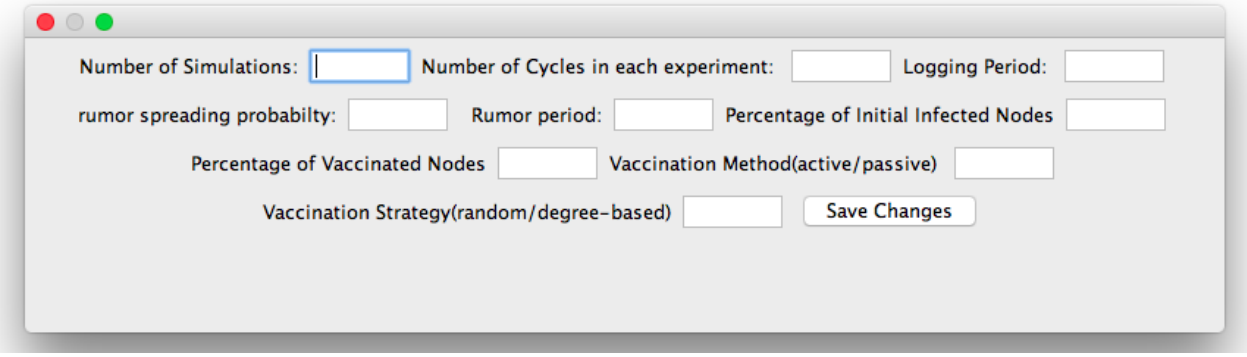
۴,۴ واسط کاربری

واسط کاربری، به نحوی طراحی شده است که کابر، بتواند به راحتی پارامترهای مربوط به شبیه‌سازی را مقداردهی اولیه کند. در صفحه اول، کاربر دکمه‌های Initialize Network، Initialize Protocol، Run، Show Results و Quit را می‌بیند.



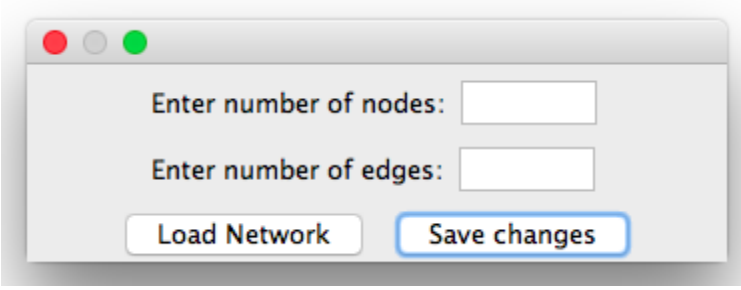
شکل ۳۵. صفحه اول برنامه

بخش Initialize Network برای تعریف شبکه مورد آزمایش طراحی شده است. در این بخش کاربر تعداد راس‌ها و یال‌های شبکه مورد نظر را در فیلدهای طراحی شده وارد می‌کند، همچنین باید فایلی را در سیستم بارگذاری کند که شامل لیست مجاورت شبکه است. فرمت این فایل باید به این صورت باشد، که در هر خط لیست مجاورت یکی از راس‌های شبکه آورده شود.



شکل ۳۶ . بخش مربوط به مقداردهی پارامترهای پروتکل شبیه‌سازی مدل SIR

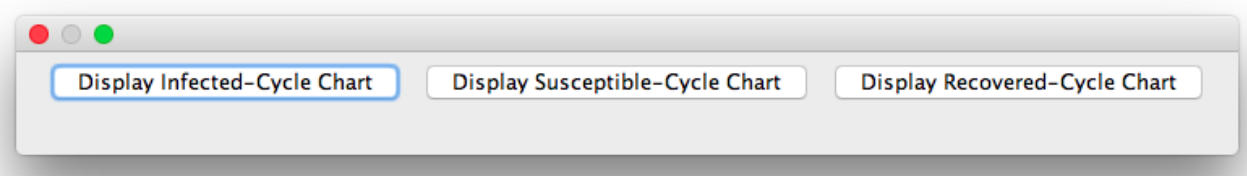
بخش Initialize Protocol برای مشخص کردن پارامترهای پروتکل پخش شایعه طراحی شده است. این بخش شامل فیلدهای مختلفی می‌شود. فیلد Number of Simulations تعداد کل آزمایش‌ها را نشان می‌دهد. فیلد Number of Cycles in each experiment تعداد چرخه‌های آزمایش را نشان می‌دهد. فیلد Log Time زمان گزارش گرفتن از سیستم را مشخص می‌کند. فیلد rumor spreading probability برای تعیین کردن احتمال پخش شایعه است. فیلد rumor period مدت زمانی را مشخص می‌کند که بعد از گذشتن این مدت زمان، یک راس آلوده در حالت بهبودیافته قرار می‌گیرد. فیلد Percentage of Initial Infected Nodes درصد راس‌های آلوده در حالت اولیه را مشخص می‌کند. فیلد Percentage of Vaccinated Nodes درصد راس‌هایی را که واکسینه می‌شوند، نشان می‌دهد. Vaccination Method حالت کلی واکسینه کردن که فعال یا غیرفعال است را مشخص می‌کند، و در نهایت Vaccination Strategy روش واکسینه کردن را که تصادفی یا مبتنی بر درجه است، مشخص می‌کند.



شکل ۳۷ . بخش مربوط به وارد کردن فایل شبکه مورد آزمایش

با کلیک کردن به روی دکمه Run، پروتکل پخش شایعه با ویژگی‌هایی که کاربر تعیین کرده است، به روی شبکه تعریف شده توسط کاربر، اجرا می‌شود.

در قسمت Show Results، کاربر امکان مشاهده سه نمودار را دارد. این سه نمودار عبارتند از: نمودار نسبت افراد آلوده به کل افراد بر حسب زمان، نسبت افراد حساس به کل افراد بر حسب زمان، و در نهایت نسبت افراد بهبودیافته به کل افراد در واحد زمان.



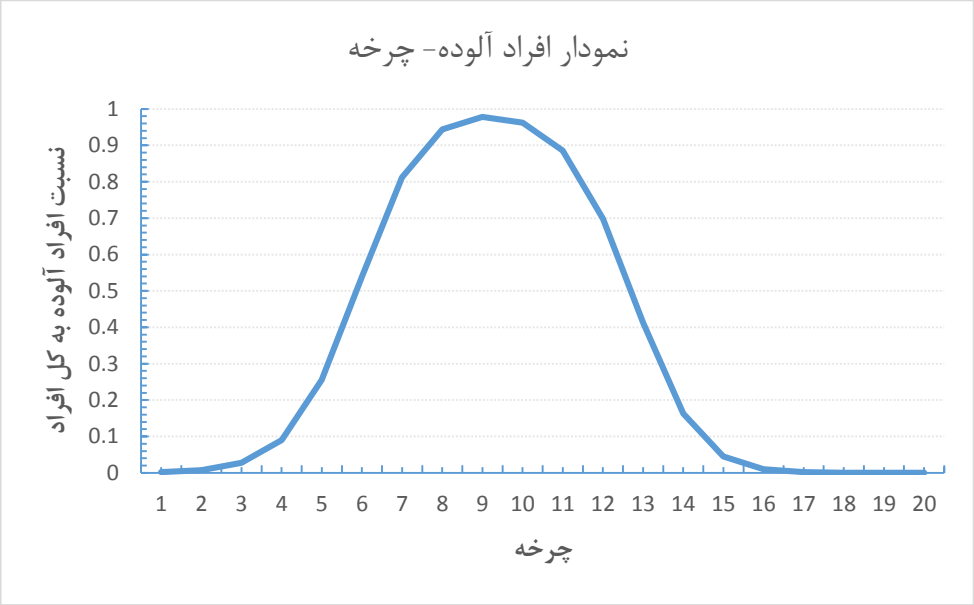
شکل ۳۸ . بخش مربوط به نمایش نتایج

۵
فصل پنجم
آزمایش و ارزیابی

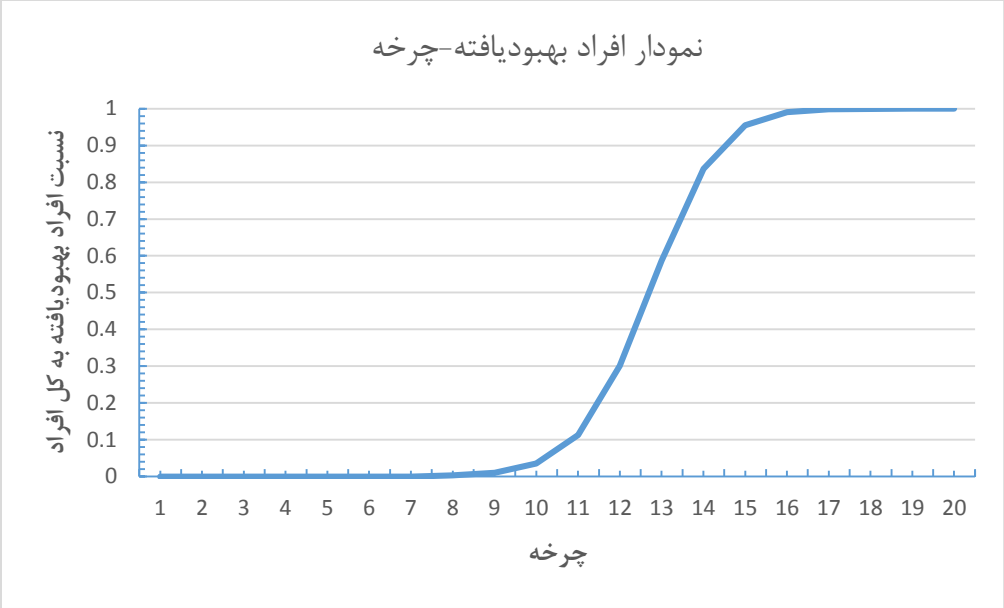
در این پروژه آزمایش‌های مختلفی را با استفاده از تغییر پارامترهای پروتکل پخش شایعه، و شبکه‌های مختلف انجام دادیم. در ابتدا، بدون واکسینه کردن راس‌ها، آزمایش‌ها را انجام دادیم. به طوری که درصد راس‌های آلوده در حالت اولیه متفاوت باشد. در ادامه نتایج به دست آمده را بررسی می‌کنیم.

۵,۱ بررسی پدیده پخش شایعه بدون واکسینه کردن راس‌ها

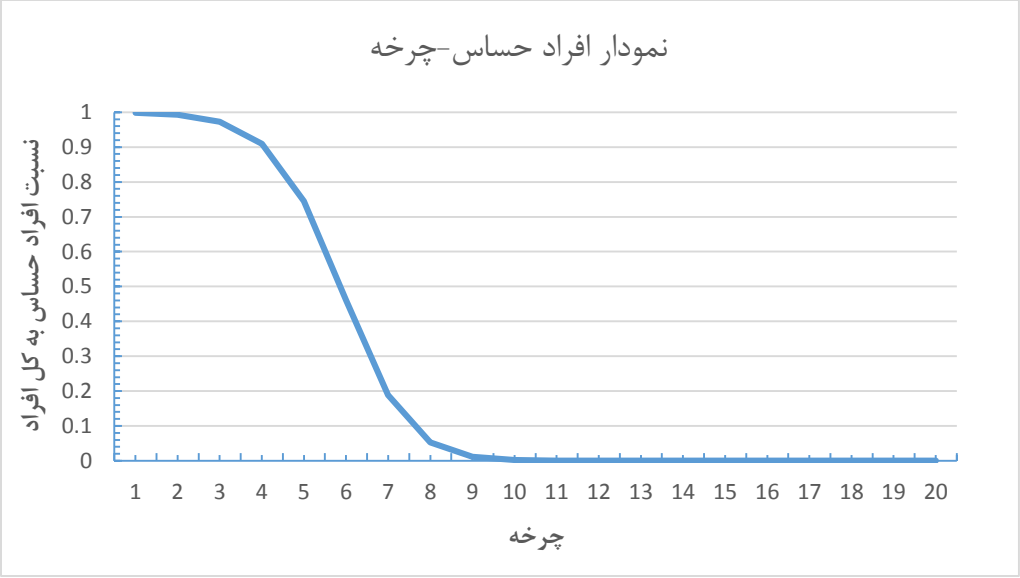
در این آزمایش‌ها، در حالت اولیه، هیچ راسی در حالت بهبودیافته قرار ندارد. و درصد کمی (بین یک تا ده درصد) از راس‌ها در حالت آلوده هستند. این آزمایش ها به روی شبکه‌های مختلفی مانند مقیاس آزاد، دنیای کوچک و شبکه نمونه‌برداری شده از twitter انجام شده است. لازم به ذکر است در آزمایش‌ها، احتمال انتقال شایعه ۸۰ درصد، مدت زمان لازم برای بهبود یافتن افراد آلوده ۷ چرخه است. در ادامه نمودارهای مربوط به این آزمایش‌های این بخش را می‌بینیم.



شکل ۳۹ . نمودار ا نمودار افراد آلوده بر حسب چرخه در شبکه مقیاس آزاد با ۱۰۰۰ راس



شکل ۴۰ . نمودار افراد بهبودیافته بر حسب چرخه در شبکه مقیاس آزاد با ۱۰۰۰ راس

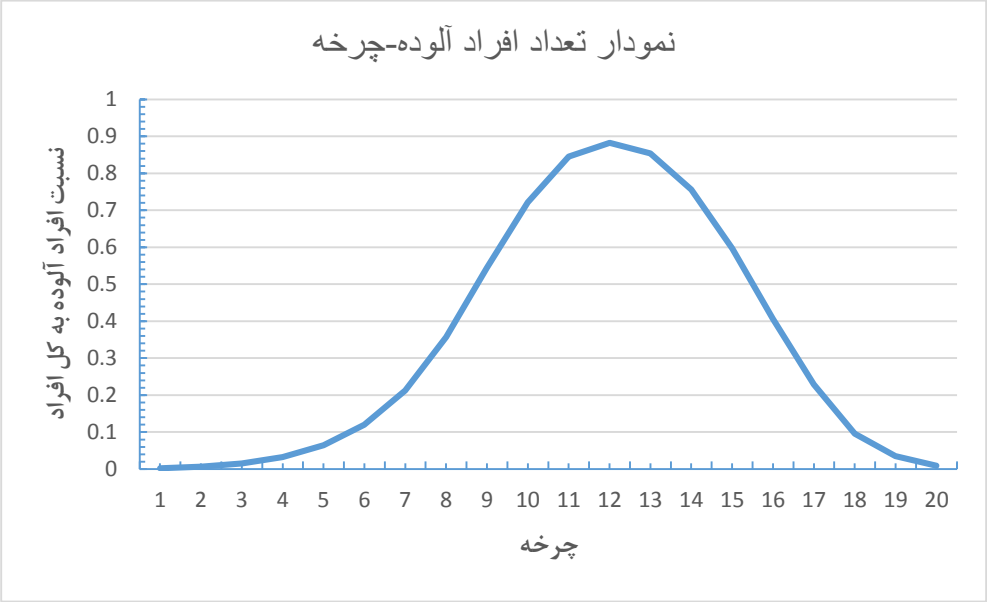


شکل ۴۱ . نمودار افراد حساس بر حسب چرخه در شبکه مقیاس آزاد با ۱۰۰۰ راس

طبق مشاهدات انجام شده در شبکه مقیاس آزاد با ۱۰۰۰ راس، و پارامترهای گفته شده در قبل، بعد از طی شدن ۹ چرخه، شایعه به حالت فراگیر پخش می‌شود، و در این چرخه ۹۷٫۸ درصد از راس‌ها در حالت آلوده قرار

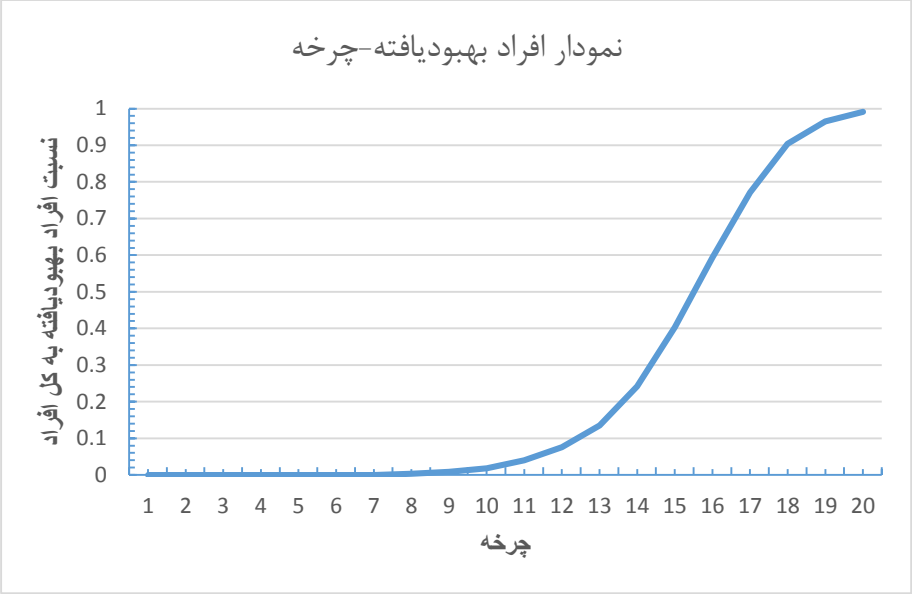
می‌گیرند. در ادامه شبیه‌سازی، تعداد راس‌های بهبودیافته افزایش می‌یابد تا در نهایت، تمامی آن‌ها در حالت بهبودیافته قرار می‌گیرند.

نمودارهای شکل‌های ۴۲ تا ۴۵ مربوط به شبیه‌سازی پروتکل به روی یک شبکه دنیای کوچک با ۱۰۰۰ راس هستند.



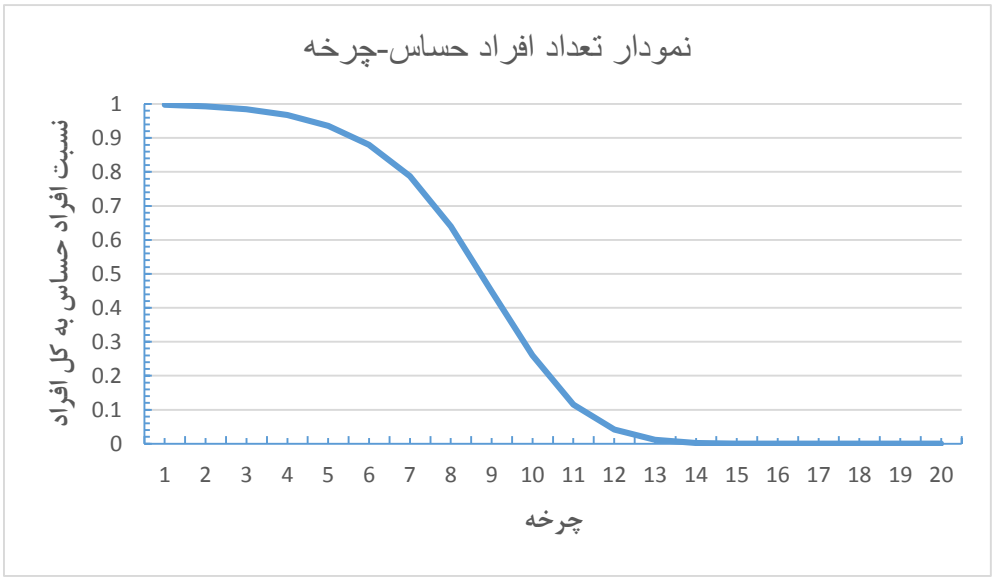
شکل ۴۲. نمودار مربوط به تعداد افراد آلوده بر حسب چرخه در شبکه دنیای کوچک با ۱۰۰۰ راس

در این آزمایش، در چرخه ۱۲، شاهد فراگیر شدن شیوع شایعه هستیم. در این چرخه ۸۸ درصد از کل راس‌ها به شایعه آلوده می‌شوند.



شکل ۴۳ . نمودار مربوط به تعداد افراد بهبودیافته بر حسب چرخه در شبکه دنیای کوچک با ۱۰۰۰ راس

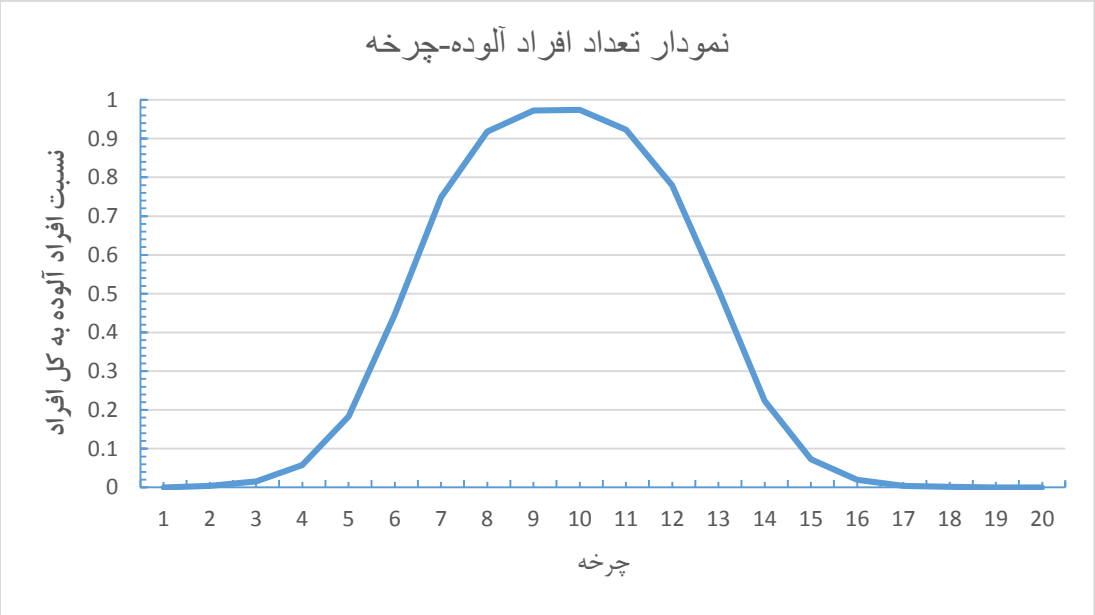
نکته حائز اهمیت این است که تا چرخه ۷، تعداد راس‌های در حالت بهبودیافته صفر بوده است. دلیل این اتفاق، این است که دوره بهبودیافتن راس‌ها ۷ چرخه تعیین شده است، و در حالت اولیه هم هیچ راسی در حالت بهبودیافته قرار ندارد. در نتیجه تا چرخه هفتم، هیچ راسی در حالت بهبودیافته قرار نمی‌گیرد، در واقع راس‌هایی که در مرحله اولیه آلوده شده‌اند، از چرخه هفتم به بعد، در حالت بهبودیافته قرار می‌گیرند.



شکل ۴۴. نمودار تعداد افراد حساس بر حسب چرخه در شبکه دنیای کوچک با ۱۰۰۰ راس

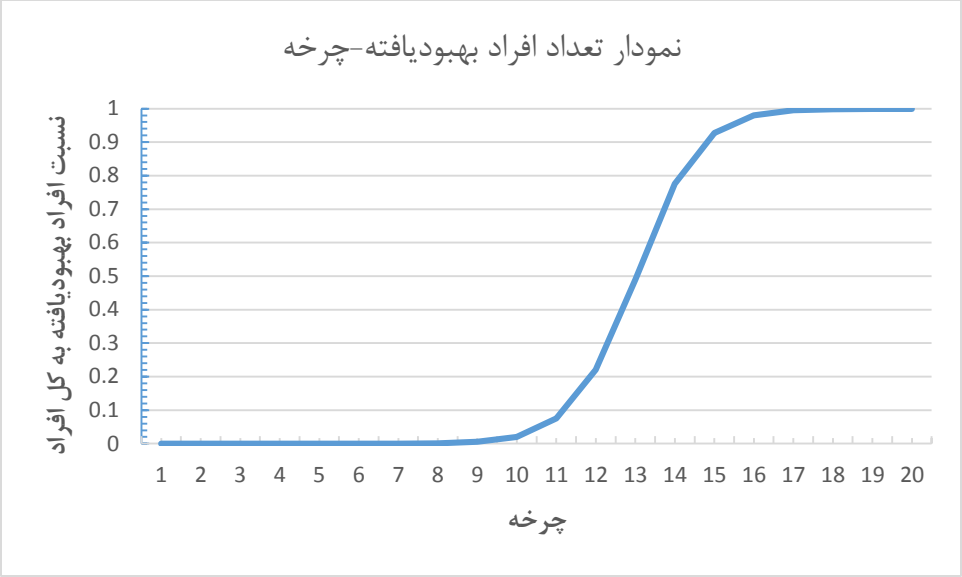
مطابق با نمودار شکل ۴۴، تعداد افراد حساس، در اثر گذر زمان، به طور نزولی اکید کاهش می‌یابد. دلیل آن این که، در ابتدای شبیه‌سازی، همانطور که در ابتدای بخش گفته شد، تنها یک راس در حالت آلوده قرار دارد، و هیچی راسی درحالت بهبودیافته قرار ندارد. از طرفی با گذشتن زمان، تعداد افراد آلوده افزایش می‌یابد، یعنی افراد از وضعیت حساس، به وضعیت آلوده تغییر وضعیت می‌دهند.

نمودارهای بعد، مربوط به آزمایش به روی شبکه نمونه‌برداری شده از توییتر می‌باشند. تعداد راس‌های این نمونه ۲۷۳۱ راس، و تعداد یال‌های آن ۱۶۴۶۲۹ یال هستند.



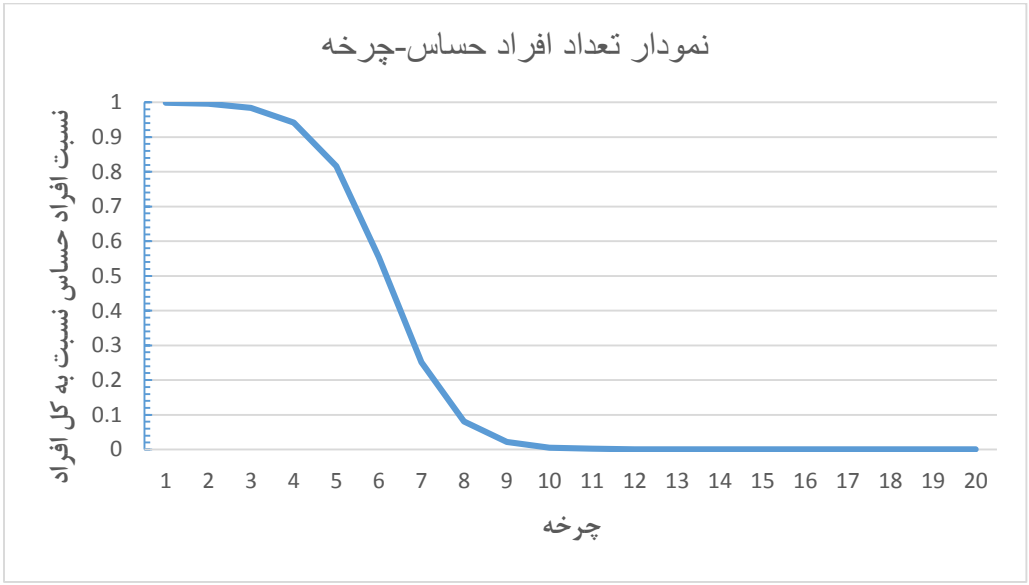
شکل ۴۵. نمودار تعداد افراد آلوده بر حسب چرخه، برای شبکه نمونه‌برداری شده از توییتر

در شبیه‌سازی پخش شایعه به روی شبکه نمونه‌برداری شده از توییتر، در چرخه دهم، شاهد فراگیر شدن پخش شایعه هستیم. که در این چرخه ۹۷ درصد از راس‌ها به شایعه آلوده شده‌اند. نکته جالب این که، در این شبیه‌سازی هم تعداد افراد آلوده در وضعیت اولیه برابر با یک راس بوده است. همان‌طوری که شاهد هستیم فراگیری شایعه مانند حالت‌های قبل با یک راس هم اتفاق می‌افتد.



شکل ٤٦. نمودار افراد بهبودیافته بر حسب چرخه در شبکه نمونه‌برداری شده توییتر

نمودار افراد بهبودیافته بر حسب چرخه، مانند شبیه‌سازی در شبکه‌های قبلی، در شبکه توییتر هم روند صعودی دارد، به طوری که در نهایت تمامی راس‌ها در حالت بهبودیافته قرار می‌گیرند.

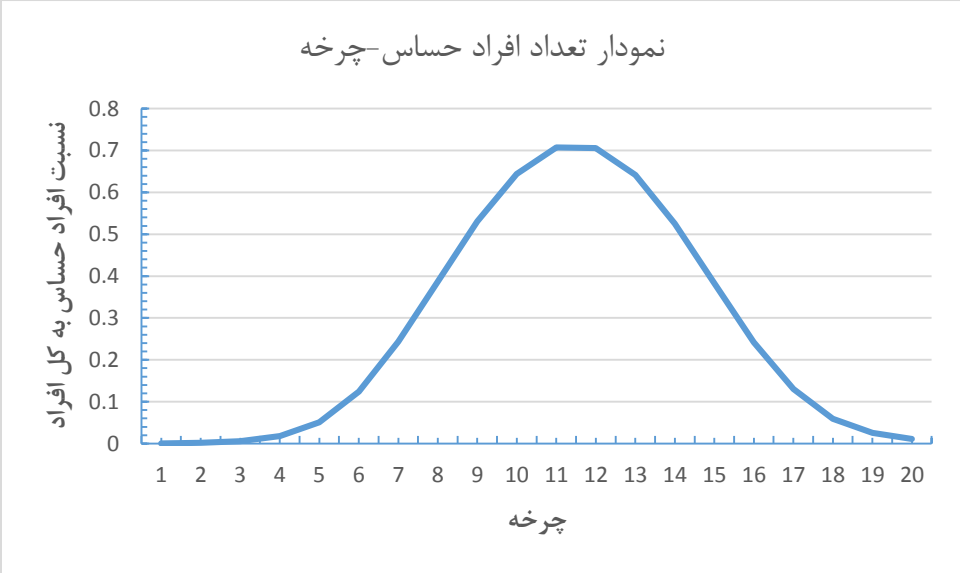


شکل ۴۷. نمودار تعداد افراد حساس بر حسب چرخه در شبکه نمونه‌برداری شده توییتر

در نمودار شکل ۴۷، روند نزولی راس‌های حساس را شاهد هستیم. مانند آزمایش‌های قبلی، راس‌های حساس، به دلیل تغییر وضعیت به حالت آلوده، این روند نزولی را در پیش می‌گیرند، تا در نهایت تعداد آن‌ها به صفر می‌رسد. زیرا در پایان تمامی راس‌ها در حالت بهبودیافته قرار می‌گیرند.

۵,۲ آزمایش با روش واکسینه کردن تصادفی در حالت غیرفعال

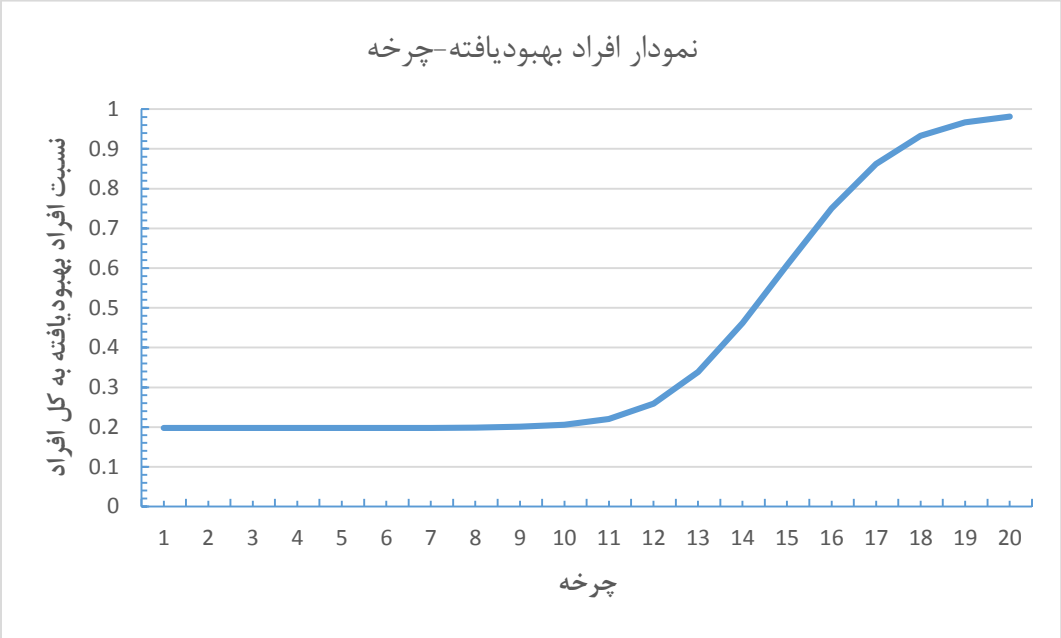
این آزمایش به روی شبکه نمونه‌برداری شده از توییتر انجام شده است. تعداد راس‌های آلوده در حالت اولیه برابر با یک راس، احتمال انتقال شایعه برابر با ۸۰ درصد، زمان مورد نیاز برای قرار گرفتن در حالت بهبود یافته ۷ چرخه، و درصد راس‌های واکسینه شده برابر با ۲۰ درصد از کل راس‌ها بوده است. این ۲۰ درصد به طور تصادفی از راس‌ها انتخاب شده‌اند. لازم به ذکر است، همان طور که قبلاً گفته شد، در حالت غیرفعال، راس‌های واکسینه شده، تاثیری در وضعیت راس‌های دیگر در صورت ارتباط با آن‌ها نمی‌گذارند. نتایج را در نمودارهای شکل‌های ۴,۱۰ تا ۴,۱۲ می‌بینیم.



شکل ۴۸ . نمودار مربوط به تعداد افراد آلوده به کل افراد در شبکه توییتر در حالت واکسینه کردن تصادفی غیرفعال

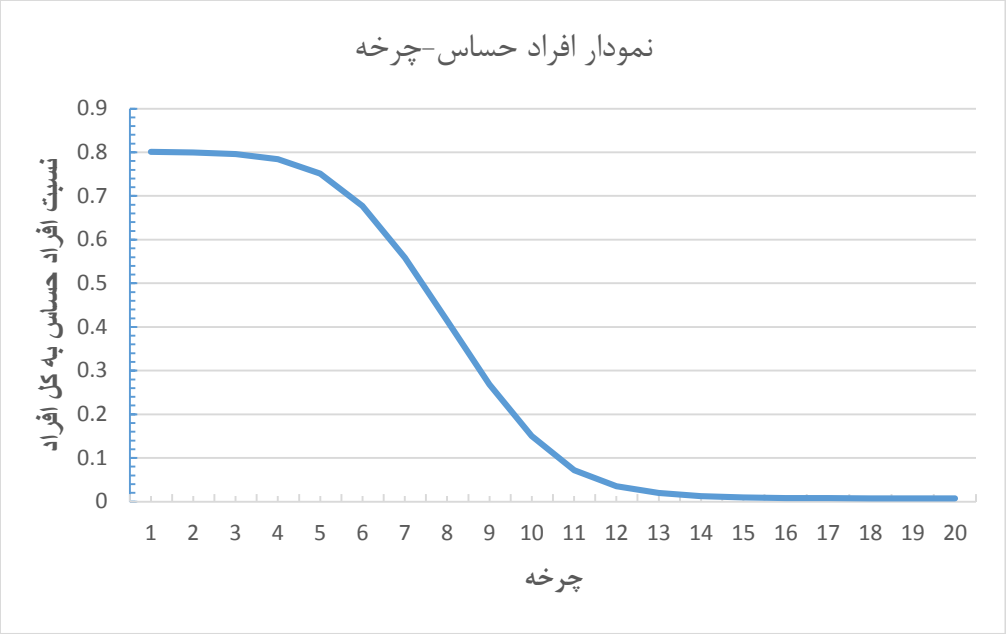
در این آزمایش نیز پدیده فراگیر شدن شایعه اتفاق افتاده است. در چرخه یازدهم شاهد ماکزیمم شدن راس‌های آلوده هستیم. نسبت به حالتی که راس‌ها را واکسینه نکردیم، زمان به اوج رسیدن شایعه یک چرخه افزایش پیدا کرده است.. از طرفی، حداکثر تعداد راس‌های آلوده در این حالت برابر با ۷۰ درصد از کل راس‌ها هستند. این مقدار نسبت به حالت بدون واکسینه کردن راس‌ها، ۲۷ درصد کاهش داشته است. البته، ۲۰ درصد از راس‌ها در

حالت اولیه در وضعیت بهبودیافته قرار دارند، که در نتیجه هیچوقت به وضعیت آلوده منتقل نمی‌شوند. با این حال، حداکثر ۸۰ درصد از راس‌ها امکان انتقال به وضعیت آلوده را دارند، که در اینجا به خاطر واکسینه کردن راس‌ها، این میزان به ۷۰ درصد کاهش یافته است.



شکل ۴۹. نمودار تعداد افراد بهبودیافته بر حسب چرخه در شبکه توییتر، با روش واکسینه کردن تصادفی راس‌ها در حالت غیرفعال.

در این آزمایش، مطابق با شکل ۴۹، تعداد راس‌های بهبودیافته تا اتمام چرخه هفتم ثابت است، که این تعداد برابر با تعداد راس‌های اولیه در حالت بهبودیافته است. از آنجایی که حالت واکسینه کردن غیرفعال است، و دوره بهبود، برای راس‌های آلوده هفت چرخه است، تعداد این راس‌ها تا چرخه هفتم ثابت می‌ماند، و از آن زمان به بعد، تعداد آن‌ها افزایش می‌یابد تا در نهایت تمامی راس‌های شبکه در وضعیت بهبودیافته قرار می‌گیرند.

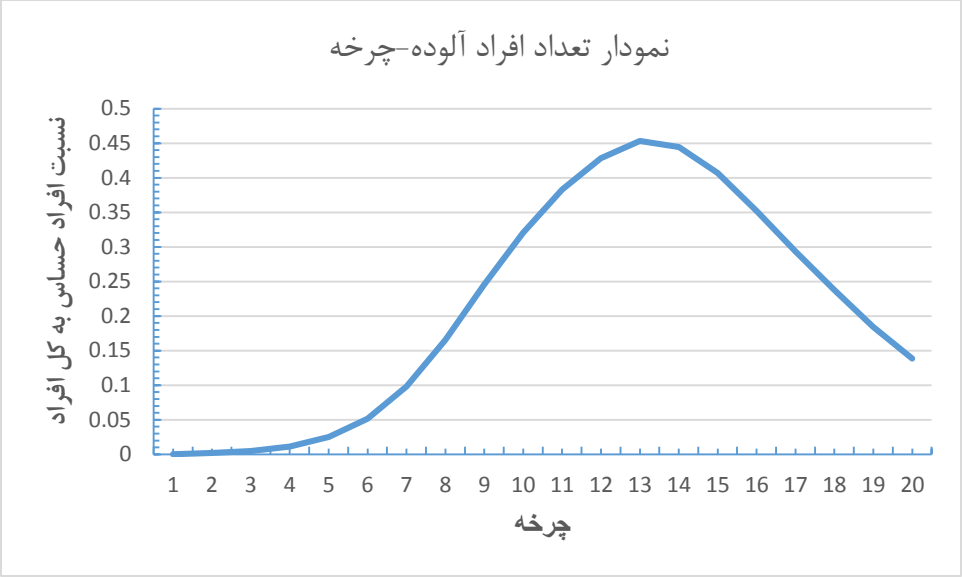


شکل ۵۰. نمودار تعداد افراد حساس بر حسب چرخه در شبکه توییتر، با روش واکسینه کردن تصادفی راس‌ها در حالت غیرفعال.

تفاوت نمودار افراد حساس بر حسب چرخه نسبت به حالت‌های قبل در این است که در ابتدا تمامی افراد در حالت حساس قرار ندارند، زیرا ۲۰ درصد از افراد جامعه برای واکسینه شدن در حالت بهبودیافته قرار می‌گیرند. سیر نزولی این نمودار مانند آزمایش‌های قبل است.

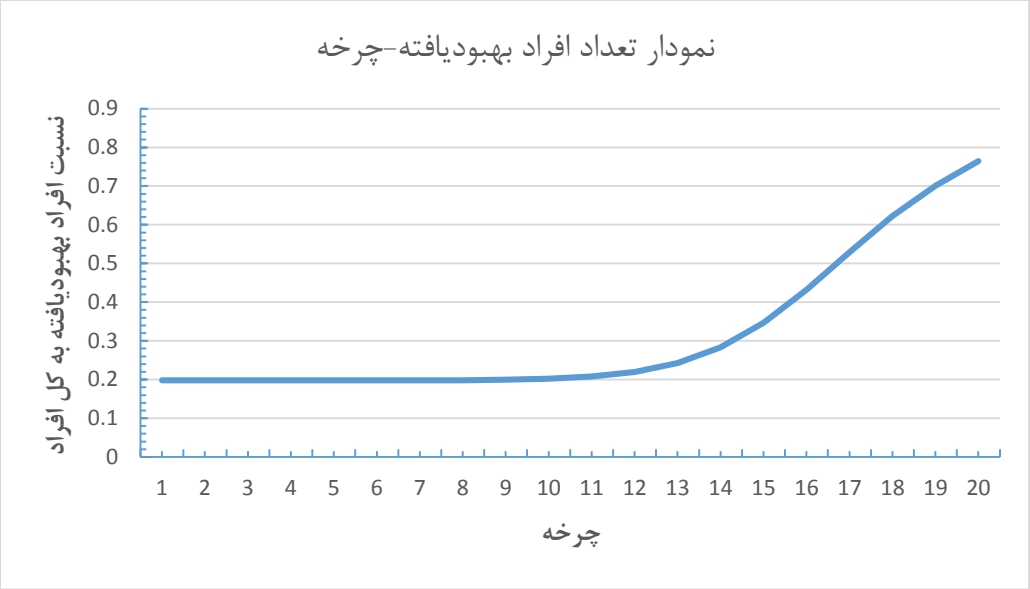
۵,۳ آزمایش با روش واکسینه کردن مبتنی بر درجه در حالت غیرفعال

در این آزمایش که به روی شبکه نمونه‌برداری شده از توییتر انجام شده است، احتمال انتقال شایعه ۸۰ درصد، دوره بهبود یافتن راس‌های آلوده برابر با ۷ چرخه، تعداد راس‌های آلوده در حالت اولیه یک راس، تعداد راس‌های واکسینه شده برابر با ۲۰ درصد از کل راس‌ها بوده است. واکسینه کردن راس‌ها بر اساس درجه بوده است. به این معنی که، راس‌های واکسینه شده بیشترین درجه را در بین راس‌ها داشته‌اند. در ادامه با توجه به نمودارهای شکل‌های ۵۱ تا ۵۳ به بررسی نتایج می‌پردازیم.



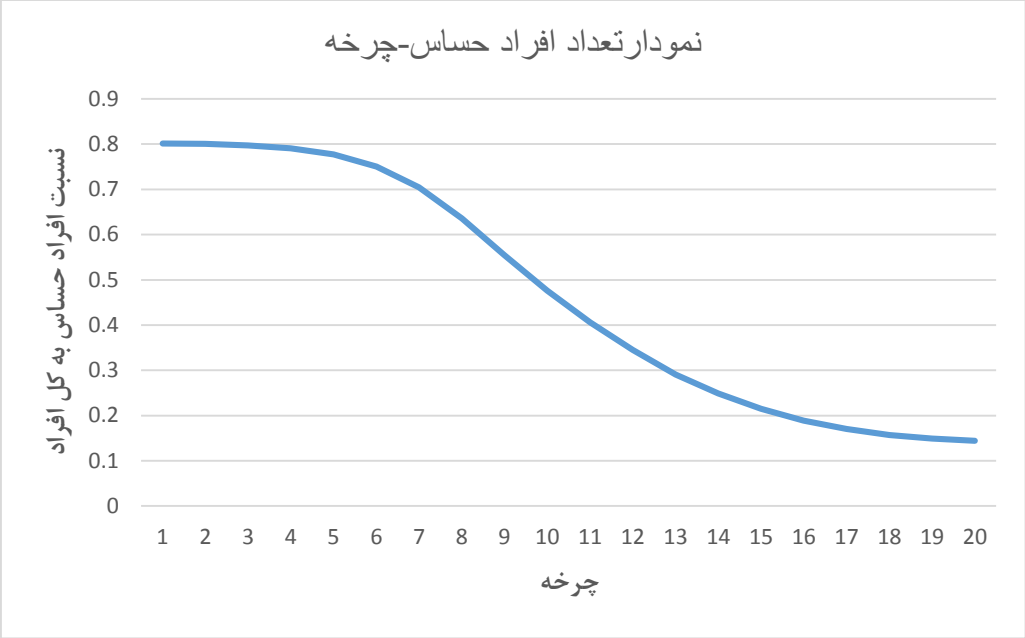
شکل ۵۱. نمودار تعداد افراد حساس بر حسب چرخه در شبکه توییتر، با روش واکسینه کردن مبتنی بر درجه در حالت غیرفعال

همان‌طوری که نمودار شکل ۵۱ نشان می‌دهد، تعداد حداکثر افرادی که در این حالت به شایعه آلوده می‌شوند، برابر با ۴۵ درصد از کل جمعیت است. این در حالی است که زمان فراگیر شدن شایعه، در چرخه سیزدهم اتفاق می‌افتد. در نتیجه نسبت به حالت تصادفی هم حداکثر افراد آلوده کاهش ۲۵ درصدی داشته است، هم زمان فراگیر شدن شایعه، دو چرخه دیرتر اتفاق افتاده است.



شکل ۵۲. مودار تعداد افراد بهبودیافته بر حسب چرخه در شبکه توییتر با روش واکسینه کردن مبتنی بر درجه در حالت غیرفعال.

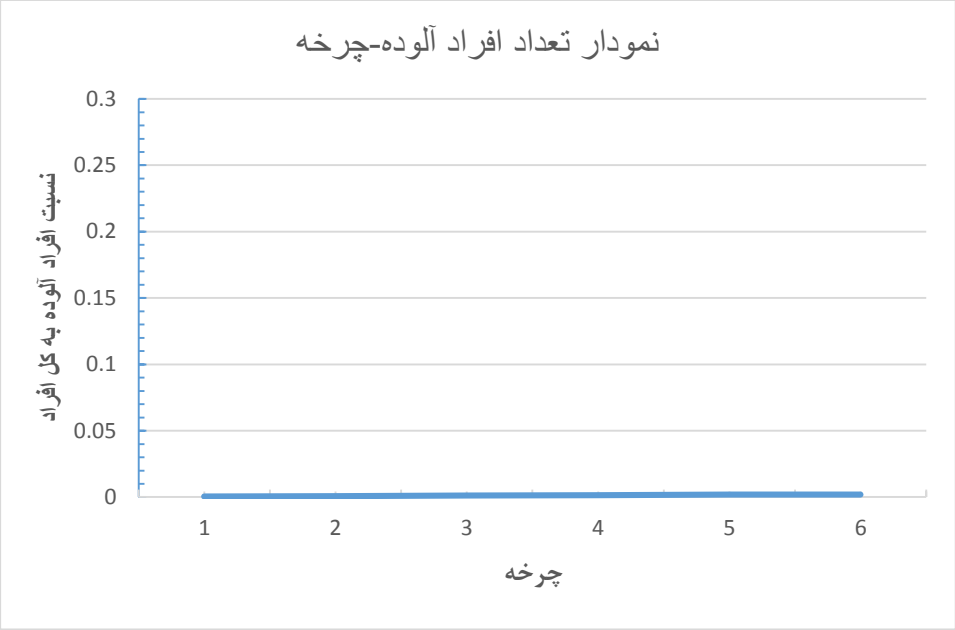
در این حالت نیز مطابق با انتظار این نمودار سیر صعودی را طی می‌کند. در ابتدا ۲۰ درصد اولیه راس‌ها در حالت بهبودیافته قرار دارند، این تعداد تا چرخه هفتم ثابت باقی می‌مانند، و از زمان به بعد، تعداد آن‌ها افرایش می‌یابد. دلیل این اتفاق هم این است که دوره زمانی لازم برای بهبود هر راس آلوده ۷ چرخه است، و اولین راس‌هایی که در حالت آلوده قرار گرفتند از چرخه هفتم به بعد وارد حالت بهبودیافته می‌شوند.



شکل ۵۳. نمودار تعداد افراد حساس بر حسب چرخه در شبکه توییتر با روش واکسینه کردن بر حسب درجه در حالت غیر فعال

۵,۴ آزمایش با روش واکسینه کردن در حالت فعال

این آزمایش به روی شبکه نمونه‌برداری شده از توییتر انجام شده است. پارامترهای اولیه مانند حالت قبل بودند. با این تفاوت که در حالت، راس‌های حالت بهبودیافته در صورت ارتباط با راس‌های حساس، با احتمال ۷۰ درصد، وضعیت آن‌ها را تغییر می‌دهند. طبق نتایج به دست آمده، در این حالت فراگیری شایعه اتفاق نمی‌افتد و کمتر از یک درصد از کل راس‌ها در حالت آلوده قرار می‌گیرند. البته چنین نتیجه‌ای دور از انتظار نیست. با توجه به این که تعداد راس‌های آلوده در حالت اولیه، یک راس است، که چیزی کمتر از یک درصد از کل راس‌ها را تشکیل می‌دهد، و درصد راس‌های واکسینه شده ۲۰ درصد کل راس‌هاست، راس‌های آلوده فرصت پخش شایعه و آلوده کردن بقیه راس‌ها را پیدا نمی‌کنند.



شکل ۵۴. . نمودار تعداد افراد آلوده بر حسب چرخه در روش واکسینه کردن فعال در توییتر

۵,۵ ارزیابی

همان‌طوری که در فصل پیش نیاز در مورد مدل SIR گفته شد، این مدل از معادلات دیفرانسیل زیر تبعیت می‌کند:

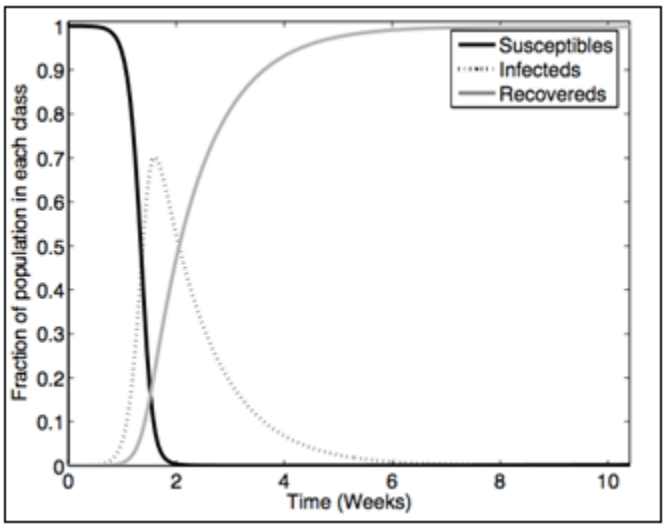
$$\frac{dS}{dt} = -\beta SI \quad .۴,۱$$

$$\frac{dI}{dt} = \beta SI - \gamma R \quad .۴,۲$$

$$\frac{dR}{dt} = \gamma R \quad .۴,۳$$

در تعیین پارامتر ثابت β احتمال انتقال شایعه، و تعداد ارتباط‌های افراد آلوده با افراد حساس نقش دارند. S، I و R در این رابطه‌ها، به ترتیب نشان‌دهنده نسبت افرادی هستند که در وضعیت حساس، آلوده و بهبودیافته قرار دارند. $1/\gamma$ بیانگر دوره زمانی لازم برای بهبود یافتن افراد آلوده است.

با حل این معادلات، نمودارهای مشابه شکل ۵۵ برای تعداد افراد بهبودیافته، آلوده و حساس، بر حسب زمان به دست می‌آید. سرعت فراگیرشدن شایعه و مدت زمان آن، با توجه به پارامترهای β و γ متفاوت هستند.



شکل ۵۵. نمودارهای مربوط به حل معادلات دیفرانسیل
۴,۱ ، ۴,۲ و ۴,۳

۶

فصل ششم

نتیجه گیری

مطابق با آزمایش‌های انجام شده، فرآیند پخش شایعه در حالتی که هیچ یک از راس‌ها واکسینه نشوند، سریع‌تر از حالت‌های قبل انجام می‌پذیرد. این فرآیند حتی با وجود آلوده شدن درصد بسیار کمی از راس‌ها در حالت اولیه، به سرعت انجام می‌پذیرد. برای کاهش سرعت این فرآیند، می‌توان راس‌ها را در حالت اولیه واکسینه کرد. این فرآیند طبق دو حالت انجام می‌گیرد. در حالت اول واکسینه کردن راس‌ها به صورت غیر فعال است، که راس‌های بهبودیافته تاثیری روی تغییر حالتِ بقیه راس‌ها ندارند. در این حالت، دو روش برای واکسینه کردن راس‌ها به کار گرفته شده است. در روش تصادفی، انتخاب راس‌ها برای واکسینه کردن به صورت تصادفی انجام می‌گیرد، سرعت پخش شایعه کمتر شده، و حداکثر راس‌هایی که در حالت آلوده قرار می‌گیرند هم کاهش یافته است. در روش دوم، که راس‌های با درجه بیشتر در اولویت واکسینه کردن قرار دارند، نتایج به مراتب از روش تصادفی بهتر می‌شود، یعنی هم سرعت فراگیر شدن شایعه، و هم حداکثر راس‌های آلوده، نسبت به حالت تصادفی کمتر می‌شود. در نهایت اگر حالت واکسینه کردن راس‌ها، فعال باشد، و راس‌های بهبودیافته بتوانند در صورت ارتباط با راس‌های حساس، وضعیت آن‌ها را به حالت بهبودیافته تغییر دهند، واکسینه کردن تاثیر بسیار قابل ملاحظه‌ای دارد، و پخش شایعه به طور کلی اتفاق نمی‌افتد.

منابع و مراجع

[1] BARABÁSI, BY ALBERT-LÁSZLÓ, and Eric Bonabeau. "*Scale-free.*" Scientific American (2003).

[2] Keeling, Matt J., and Pejman Rohani. "*Modeling infectious diseases in humans and animals*." Princeton University Press, 2008.

[3] Jelasity, Márk, and Alberto Montresor. "*Epidemic-style proactive aggregation in large overlay networks.*" Distributed Computing Systems, 2004. Proceedings. 24th International Conference on. IEEE, 2004.

[4] Jelasity, Márk, et al. "*Gossip-based peer sampling*." ACM Transactions on Computer Systems (TOCS) 25.3 (2007): 8

[5] Jelasity, Márk, and Ozalp Babaoglu. "*T-Man: Gossip-based overlay topology management.*" Engineering Self-Organising Systems.Springer Berlin Heidelberg, 2005. 1-15.